



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

OASIS Standard **incorporating Approved Errata** 12 July 2017

Specification URIs

This version:

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

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

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

Previous 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>

Latest version:

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

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

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

Technical Committee:

OASIS eXtensible Access Control Markup Language (XACML) TC

Chairs:

Bill Parducci (bill@parducci.net), Individual

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

Editor:

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

Additional artifacts:

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

- List of Errata: *eXtensible Access Control Markup Language (XACML) Version 3.0 Errata 01*. Edited by Richard C. Hill and Hal Lockhart. 12 July 2017. Approved Errata. <http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os.html>.
- *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01*. Edited by Erik Rissanen. 12 July 2017. OASIS Standard incorporating Approved Errata. <http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-complete.html>.
- XML schema – unmodified from OASIS Standard: <http://docs.oasis-open.org/xacml/3.0/errata01/os/schema/xacml-core-v3-schema-wd-17.xsd>.

Related work:

This specification provides Errata for:

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

Declared XML namespace:

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

Abstract:

This document represents the OASIS Standard *eXtensible Access Control Markup Language (XACML) Version 3.0* 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. 12 July 2017. OASIS Standard incorporating Approved Errata.

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

Latest version: <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html>.

Notices

Copyright © OASIS Open 2017. All Rights Reserved.

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

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

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

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

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

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

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

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

Table of Contents

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

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

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

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

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

1 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
94 a, or other custom attributes against those provided in the request context as a part of the
95 process of determining whether the **rule**, **policy**, or **policy set** is intended applicable to
96 evaluate the current 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

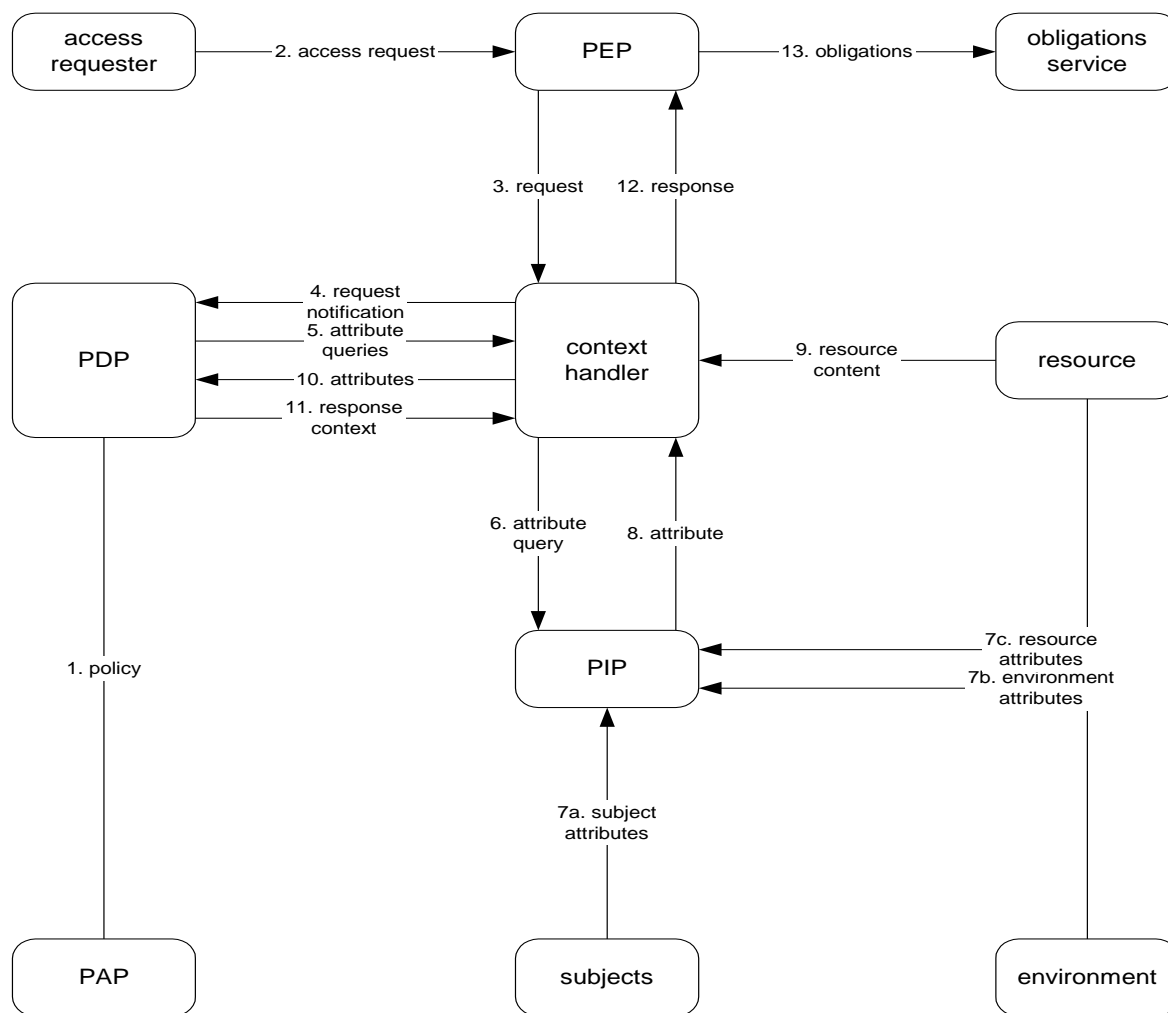
458 In some applications it is helpful to specify supplemental information about a decision. XACML provides
459 facilities to specify supplemental information about a decision with the <Advice> element. Such **advice**
460 may be safely ignored by the **PEP**.

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.

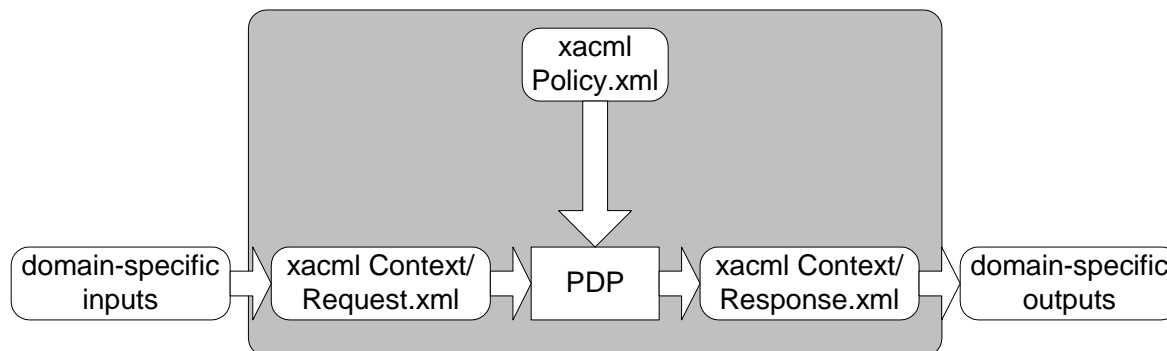
473 The model operates by the following steps.

- 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**.

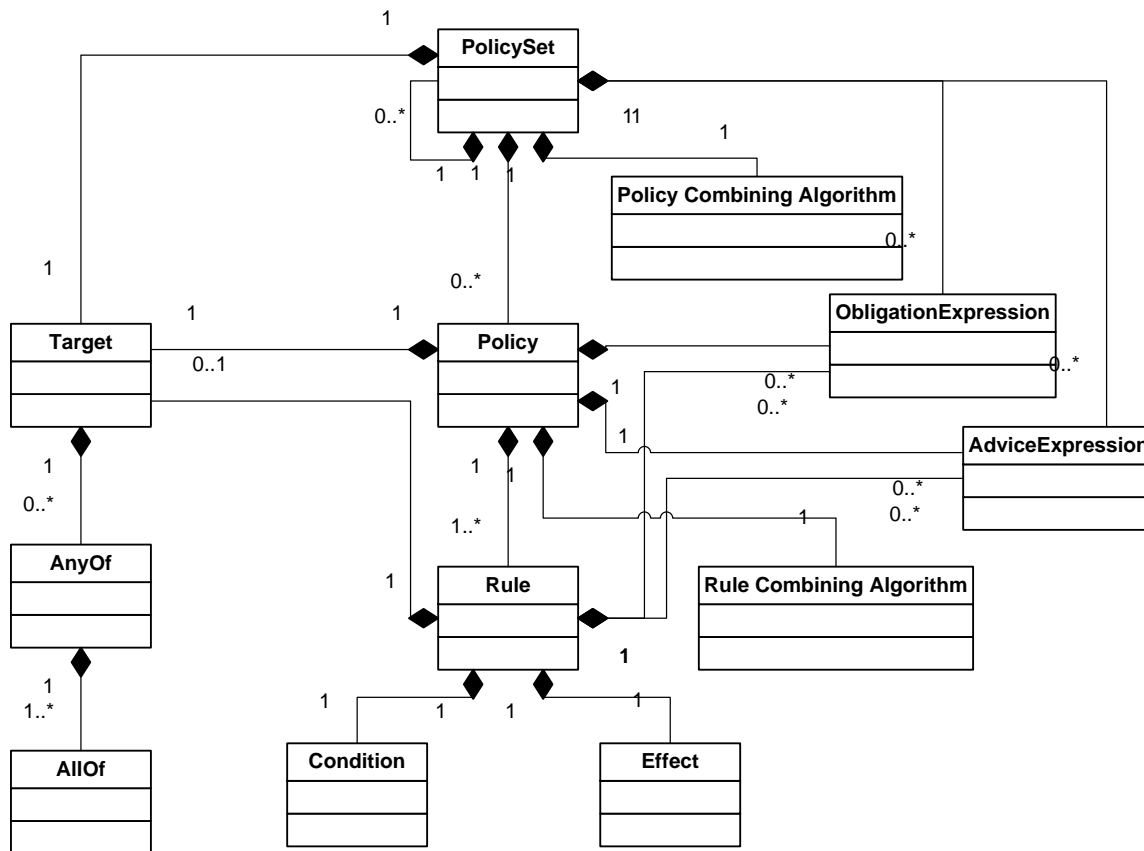
516 **3.3 Policy language model**

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

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

521 These are described in the following sub-sections.

522



523

524 *Figure 3 - Policy language model*

525 **3.3.1 Rule**

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

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

534 These are discussed in the following sub-sections.

535 3.3.1.1 Rule target

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

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

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

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

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

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

561 3.3.1.2 Effect

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

564 3.3.1.3 Condition

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

567 3.3.1.4 Obligation expressions

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

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

572 3.3.1.5 Advice

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

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

577 3.3.2 Policy

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

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

586 3.3.2.1 Policy target

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

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

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

606 3.3.2.2 Rule-combining algorithm

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

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

612 3.3.2.3 Obligation expressions

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

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

617 3.3.2.4 Advice

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

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

622 3.3.3 Policy set

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

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

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

631 3.3.3.1 Policy-combining algorithm

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

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

638 3.3.3.2 Obligation expressions

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

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

644 3.3.3.3 Advice expressions

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

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

650 4 Examples (non-normative)

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

655 4.1 Example one

656 4.1.1 Example policy

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

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

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

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

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

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

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

749 4.1.2 Example request context

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

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

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

```

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

```

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

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

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

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

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

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

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

813 4.1.3 Example response context

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

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

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

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

831 [c6] closes the response *context*.

832 4.2 Example two

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

836 4.2.1 Example medical record instance

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

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

```

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

```

921 4.2.2 Example request context

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

```

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

```

```

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

```

1000 [e2] - [e4] Standard namespace declarations.

1001 [e6] - [e27] **Access subject attributes** are placed in the urn:oasis:names:tc:xacml:1.0:subject-
1002 category:access-subject **attribute** category of the <Request> element. Each **attribute** consists of the
1003 **attribute** meta-data and the **attribute** value. There is only one **subject** involved in this request. This
1004 value of the **attribute** category denotes the identity for which the request was issued.

1005 [e8] - [e13] **Subject** subject-id **attribute**.

1006 [e14] - [e20] **Subject** role **attribute**.

1007 [e21] - [e26] **Subject** physician-id **attribute**.

1008 [e28] - [e56] **Resource attributes** are placed in the urn:oasis:names:tc:xacml:3.0:attribute-
 1009 category:resource **attribute** category of the <Request> element. Each **attribute** consists of **attribute**
 1010 meta-data and an **attribute** value.

1011 [e30] - [e42] **Resource** content. The XML **resource** instance, **access** to all or part of which may be
 1012 requested, is placed here.

1013 [e43] - [e49] The identifier of the **Resource** instance for which **access** is requested, which is an XPath
 1014 expression into the <Content> element that selects the data to be accessed.

1015 [e57] - [e64] **Action attributes** are placed in the urn:oasis:names:tc:xacml:3.0:attribute-category:action
 1016 **attribute** category of the <Request> element.

1017 [e59] - [e63] **Action** identifier.

1018 4.2.3 Example plain-language rules

1019 The following plain-language **rules** are to be enforced:

- 1020 Rule 1: A person, identified by his or her patient number, may read any record for which he or she is
 1021 the designated patient.
- 1022 Rule 2: A person may read any record for which he or she is the designated parent or guardian, and
 1023 for which the patient is under 16 years of age.
- 1024 Rule 3: A physician may write to any medical element for which he or she is the designated primary
 1025 care physician, provided an email is sent to the patient.
- 1026 Rule 4: An administrator shall not be permitted to read or write to medical elements of a patient
 1027 record.

1028 These **rules** may be written by different **PAPs** operating independently, or by a single **PAP**.

1029 4.2.4 Example XACML rule instances

1030 4.2.4.1 Rule 1

1031 **Rule 1** illustrates a simple **rule** with a single <Condition> element. It also illustrates the use of the
 1032 <VariableDefinition> element to define a function that may be used throughout the **policy**. The
 1033 following XACML <Rule> instance expresses **Rule 1**:

```

1034 [f1] <?xml version="1.0" encoding="UTF-8"?>
1035 [f2] <Policy
1036 [f3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1037 [f4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1038 [f5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1039 [f6]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1040 [f7]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:1"
1041 [f8]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1042     algorithm:deny-overrides"
1043 [f9]   Version="1.0">
1044 [f10]   <PolicyDefaults>
1045 [f11]     <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1046 [f12]   </PolicyDefaults>
1047 [f13]   <Target/>
1048 [f14]   <VariableDefinition VariableId="17590034">
1049 [f15]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1050 [f16]       <Apply
1051 [f17]         FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1052 [f18]           <AttributeDesignator
1053 [f19]             MustBePresent="false"
1054 [f20]             Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
1055             subject"
1056 [f21]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:patient-
1057             number"
1058 [f22]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1059 [f23]           </Apply>
1060 [f24]         <Apply
1061 [f25]           FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">

```

```

1062 [f26]         <AttributeSelector
1063 [f27]             MustBePresent="false"
1064 [f28]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1065 [f29]             Path="md:record/md:patient/md:patient-number/text()"
1066 [f30]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1067 [f31]         </Apply>
1068 [f32]     </Apply>
1069 [f33] </VariableDefinition>
1070 [f34] <Rule
1071 [f35]     RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
1072 [f36]     Effect="Permit">
1073 [f37]     <Description>
1074 [f38]         A person may read any medical record in the
1075 [f39]         http://www.med.example.com/schemas/record.xsd namespace
1076 [f40]         for which he or she is the designated patient
1077 [f41]     </Description>
1078 [f42]     <Target>
1079 [f43]         <AnyOf>
1080 [f44]             <AllOf>
1081 [f45]                 <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1082 [f46]                     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1083 [f47]                         >urn:example:med:schemas:record</AttributeValue>
1084 [f48]                     <AttributeDesignator
1085 [f49]                         MustBePresent="false"
1086 [f50]                         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1087 [f51]                         AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1088 [f52]                         DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1089 [f53]                     </Match>
1090 [f54]                 <Match
1091 [f55]                     MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1092 [f56]                         <AttributeValue
1093 [f57]                             DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1094 [f58]                             XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1095 [f59]                             >md:record</AttributeValue>
1096 [f60]                         <AttributeDesignator
1097 [f61]                             MustBePresent="false"
1098 [f62]                             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1099 [f63]                             AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1100 [f64]                             DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1101 [f65]                         </Match>
1102 [f66]                     </AllOf>
1103 [f67]                 </AnyOf>
1104 [f68]             <AnyOf>
1105 [f69]                 <AllOf>
1106 [f70]                     <Match
1107 [f71]                         MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1108 [f72]                             <AttributeValue
1109 [f73]                                 DataType="http://www.w3.org/2001/XMLSchema#string"
1110 [f74]                                 >read</AttributeValue>
1111 [f75]                             <AttributeDesignator
1112 [f76]                                 MustBePresent="false"
1113 [f77]                                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1114 [f78]                                 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1115 [f79]                                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1116 [f80]                             </Match>
1117 [f81]                     </AllOf>
1118 [f82]                 </AnyOf>
1119 [f83]             </Target>
1120 [f84]         <Condition>
1121 [f85]             <VariableReference VariableId="17590034"/>
1122 [f86]         </Condition>
1123 [f87]     </Rule>
1124 [f88] </Policy>

```

1125 [f3] - [f6] XML namespace declarations.

1126 [f11] XPath expressions in the **policy** are to be interpreted according to the 1.0 version of the XPath specification.

1128 [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**.

1130 [f15] The `FunctionId` attribute names the function to be used for comparison. In this case, comparison
1131 is done with the “urn:oasis:names:tc:xacml:1.0:function:string-equal” function; this function takes two
1132 arguments of type “http://www.w3.org/2001/XMLSchema#string”.

1133 [f17] The first argument of the variable definition is a function specified by the `FunctionId` attribute.
1134 Since urn:oasis:names:tc:xacml:1.0:function:string-equal takes arguments of type
1135 “http://www.w3.org/2001/XMLSchema#string” and `AttributeDesignator` selects a **bag** of type
1136 “http://www.w3.org/2001/XMLSchema#string”, “urn:oasis:names:tc:xacml:1.0:function:string-one-and-
1137 only” is used. This function guarantees that its argument evaluates to a **bag** containing exactly one
1138 value.

1139 [f18] The `AttributeDesignator` selects a **bag** of values for the patient-number **subject attribute** in
1140 the request **context**.

1141 [f25] The second argument of the variable definition is a function specified by the `FunctionId` attribute.
1142 Since “urn:oasis:names:tc:xacml:1.0:function:string-equal” takes arguments of type
1143 “http://www.w3.org/2001/XMLSchema#string” and the `AttributeSelector` selects a **bag** of type
1144 “http://www.w3.org/2001/XMLSchema#string”, “urn:oasis:names:tc:xacml:1.0:function:string-one-and-
1145 only” is used. This function guarantees that its argument evaluates to a **bag** containing exactly one
1146 value.

1147 [f26] The `<AttributeSelector>` element selects a **bag** of values from the **resource** content using a
1148 free-form XPath expression. In this case, it selects the value of the patient-number in the **resource**.
1149 Note that the namespace prefixes in the XPath expression are resolved with the standard XML
1150 namespace declarations.

1151 [f35] **Rule** identifier.

1152 [f36] **Rule effect** declaration. When a **rule** evaluates to ‘True’ it emits the value of the `Effect` attribute.
1153 This value is then combined with the `Effect` values of other **rules** according to the **rule-combining**
1154 **algorithm**.

1155 [f37] - [f41] Free form description of the **rule**.

1156 [f42] - [f83] A **rule target** defines a set of **decision requests** that the **rule** is intended to evaluate.

1157 [f43] - [f67] The `<AnyOf>` element contains a **disjunctive sequence** of `<AllOf>` elements. In this
1158 example, there is just one.

1159 [f44] - [f66] The `<AllOf>` element encloses the **conjunctive sequence** of `Match` elements. In this
1160 example, there are two.

1161 [f45] - [f53] The first `<Match>` element compares its first and second child elements according to the
1162 matching function. A match is positive if the value of the first argument matches any of the values
1163 selected by the second argument. This match compares the **target** namespace of the requested
1164 document with the value of “urn:example:med:schemas:record”.

1165 [f45] The `MatchId` attribute names the matching function.

1166 [f46] - [f47] Literal **attribute** value to match.

1167 [f48] - [f52] The `<AttributeDesignator>` element selects the **target** namespace from the **resource**
1168 contained in the request **context**. The **attribute** name is specified by the `AttributeId`.

1169 [f54] - [f65] The second `<Match>` element. This match compares the results of two XPath expressions
1170 applied to the `<Content>` element of the **resource** category. The second XPath expression is the
1171 location path to the requested XML element and the first XPath expression is the literal value “md:record”.
1172 The “xpath-node-match” function evaluates to “True” if the requested XML element is below the
1173 “md:record” element.

1174 [f68] - [f82] The `<AnyOf>` element contains a **disjunctive sequence** of `<AllOf>` elements. In this case,
1175 there is just one `<AllOf>` element.

1176 [f69] - [f81] The `<AllOf>` element contains a **conjunctive sequence** of `<Match>` elements. In this case,
1177 there is just one `<Match>` element.

1178 [f70] - [f80] The <Match> element compares its first and second child elements according to the matching
1179 function. The match is positive if the value of the first argument matches any of the values selected by
1180 the second argument. In this case, the value of the action-id **action attribute** in the request **context** is
1181 compared with the literal value "read".

1182 [f84] - [f86] The <Condition> element. A **condition** must evaluate to "True" for the **rule** to be
1183 applicable. This **condition** contains a reference to a variable definition defined elsewhere in the **policy**.

1184 4.2.4.2 Rule 2

1185 **Rule 2** illustrates the use of a mathematical function, i.e. the <Apply> element with functionId
1186 "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" to calculate the date of the patient's
1187 sixteenth birthday. It also illustrates the use of **predicate** expressions, with the functionId
1188 "urn:oasis:names:tc:xacml:1.0:function:and". This example has one function embedded in the
1189 <Condition> element and another one referenced in a <VariableDefinition> element.

```
1190 [g1] <?xml version="1.0" encoding="UTF-8"?>
1191 [g2] <Policy
1192 [g3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1193 [g4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1194 [g5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1195 [g6]   xmlns:xf="http://www.w3.org/2005/xpath-functions"
1196 [g7]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1197 [g8]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
1198 [g9]   Version="1.0"
1199 [g10]  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1200      algorithm:deny-overrides">
1201 [g11]  <PolicyDefaults>
1202 [g12]    <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1203 [g13]  </PolicyDefaults>
1204 [g14]  <Target/>
1205 [g15]  <VariableDefinition VariableId="17590035">
1206 [g16]    <Apply
1207 [g17]      FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal">
1208 [g18]      <Apply
1209 [g19]        FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
1210 [g20]          <AttributeDesignator
1211 [g21]            MustBePresent="false"
1212 [g22]            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
1213 [g23]            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
1214 [g24]            DataType="http://www.w3.org/2001/XMLSchema#date"/>
1215 [g25]          </Apply>
1216 [g26]        <Apply
1217 [g27]          FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration">
1218 [g28]            <Apply
1219 [g29]              FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
1220 [g30]                <AttributeSelector
1221 [g31]                  MustBePresent="false"
1222 [g32]                  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1223 [g33]                  Path="md:record/md:patient/md:patientDoB/text()"
1224 [g34]                  DataType="http://www.w3.org/2001/XMLSchema#date"/>
1225 [g35]                </Apply>
1226 [g36]              <AttributeValue
1227 [g37]                DataType="http://www.w3.org/2001/XMLSchema#yearMonthDuration"
1228 [g38]                >P16Y</AttributeValue>
1229 [g39]            </Apply>
1230 [g40]          </Apply>
1231 [g41]        </VariableDefinition>
1232 [g42]      <Rule
1233 [g43]        RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
1234 [g44]        Effect="Permit">
1235 [g45]        <Description>
1236 [g46]          A person may read any medical record in the
1237 [g47]          http://www.med.example.com/records.xsd namespace
1238 [g48]          for which he or she is the designated parent or guardian,
1239 [g49]          and for which the patient is under 16 years of age
1240 [g50]        </Description>
1241 [g51]        <Target>
1242 [g52]          <AnyOf>
1243 [g53]            <AllOf>
```

```

1244 [g54] <Match
1245 [g55] MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1246 [g56] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1247 [g57] >urn:example:med:schemas:record</AttributeValue>
1248 [g58] <AttributeDesignator
1249 [g59] MustBePresent="false"
1250 [g60] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1251 [g61] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1252 [g62] DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1253 [g63] </Match>
1254 [g64] <Match
1255 [g65] MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1256 [g66] <AttributeValue
1257 [g67] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1258 [g68] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1259 [g69] >md:record</AttributeValue>
1260 [g70] <AttributeDesignator
1261 [g71] MustBePresent="false"
1262 [g72] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1263 [g73] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1264 [g74] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1265 [g75] </Match>
1266 [g76] </AllOf>
1267 [g77] </AnyOf>
1268 [g78] <AnyOf>
1269 [g79] <AllOf>
1270 [g80] <Match
1271 [g81] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1272 [g82] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1273 [g83] >read</AttributeValue>
1274 [g84] <AttributeDesignator
1275 [g85] MustBePresent="false"
1276 [g86] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1277 [g87] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1278 [g88] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1279 [g89] </Match>
1280 [g90] </AllOf>
1281 [g91] </AnyOf>
1282 [g92] </Target>
1283 [g93] <Condition>
1284 [g94] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
1285 [g95] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1286 [g96] <Apply
1287 [g97] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1288 [g98] <AttributeDesignator
1289 [g99] MustBePresent="false"
1290 [g100] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1291 [g101] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:parent-
guardian-id"
1292 [g102] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1293 [g103] </Apply>
1294 [g104] <Apply
1295 [g105] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1296 [g106] <AttributeSelector
1297 [g107] MustBePresent="false"
1298 [g108] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1299 [g109] Path="md:record/md:parentGuardian/md:parentGuardianId/text()"
1300 [g110] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1301 [g111] </Apply>
1302 [g112] </Apply>
1303 [g113] <VariableReference VariableId="17590035"/>
1304 [g114] </Apply>
1305 [g115] </Condition>
1306 [g116] </Rule>
1307 [g117] </Policy>

```

1309 [g15] - [g41] The <VariableDefinition> element contains part of the **condition** (i.e. is the patient
1310 under 16 years of age?). The patient is under 16 years of age if the current date is less than the date
1311 computed by adding 16 to the patient's date of birth.

1312 [g16] - [g40] "urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal" is used to compare the two date
1313 arguments.

1314 [g18] - [g25] The first date argument uses “urn:oasis:names:tc:xacml:1.0:function:date-one-and-only” to
 1315 ensure that the **bag** of values selected by its argument contains exactly one value of type
 1316 “http://www.w3.org/2001/XMLSchema#date”.

1317 [g20] The current date is evaluated by selecting the “urn:oasis:names:tc:xacml:1.0:environment:current-
 1318 date” **environment attribute**.

1319 [g26] - [g39] The second date argument uses “urn:oasis:names:tc:xacml:1.0:function:date-add-
 1320 yearMonthDuration” to compute the date of the patient’s sixteenth birthday by adding 16 years to the
 1321 patient’s date of birth. The first of its arguments is of type “http://www.w3.org/2001/XMLSchema#date”
 1322 and the second is of type “http://www.w3.org/TR/2007/REC-xpath-functions-20070123/#dt-
 1323 yearMonthDuration”.

1324 [g30] The <AttributeSelector> element selects the patient’s date of birth by taking the XPath
 1325 expression over the **resource** content.

1326 [g36] - [g38] Year Month Duration of 16 years.

1327 [g51] - [g92] **Rule** declaration and **rule target**. See **Rule** 1 in Section 4.2.4.1 for the detailed explanation
 1328 of these elements.

1329 [g93] - [g115] The <Condition> element. The **condition** must evaluate to “True” for the **rule** to be
 1330 applicable. This **condition** evaluates the truth of the statement: the requestor is the designated parent or
 1331 guardian and the patient is under 16 years of age. It contains one embedded <Apply> element and one
 1332 referenced <VariableDefinition> element.

1333 [g94] The **condition** uses the “urn:oasis:names:tc:xacml:1.0:function:and” function. This is a Boolean
 1334 function that takes one or more Boolean arguments (2 in this case) and performs the logical “AND”
 1335 operation to compute the truth value of the expression.

1336 [g95] - [g112] The first part of the **condition** is evaluated (i.e. is the requestor the designated parent or
 1337 guardian?). The function is “urn:oasis:names:tc:xacml:1.0:function:string-equal” and it takes two
 1338 arguments of type “http://www.w3.org/2001/XMLSchema#string”.

1339 [g96] designates the first argument. Since “urn:oasis:names:tc:xacml:1.0:function:string-equal” takes
 1340 arguments of type “http://www.w3.org/2001/XMLSchema#string”,
 1341 “urn:oasis:names:tc:xacml:1.0:function:string-one-and-only” is used to ensure that the **subject attribute**
 1342 “urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id” in the request **context** contains
 1343 exactly one value.

1344 [g98] designates the first argument. The value of the **subject attribute**
 1345 “urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id” is selected from the request **context**
 1346 using the <AttributeDesignator> element.

1347 [g104] As above, the “urn:oasis:names:tc:xacml:1.0:function:string-one-and-only” is used to ensure that
 1348 the **bag** of values selected by its argument contains exactly one value of type
 1349 “http://www.w3.org/2001/XMLSchema#string”.

1350 [g106] The second argument selects the value of the <md:parentGuardianId> element from the
 1351 **resource** content using the <AttributeSelector> element. This element contains a free-form XPath
 1352 expression, pointing into the <Content> element of the resource category. Note that all namespace
 1353 prefixes in the XPath expression are resolved with standard namespace declarations. The
 1354 AttributeSelector evaluates to the **bag** of values of type
 1355 “http://www.w3.org/2001/XMLSchema#string”.

1356 [g113] references the <VariableDefinition> element, where the second part of the **condition** is
 1357 defined.

1358 4.2.4.3 Rule 3

1359 **Rule 3** illustrates the use of an **obligation** expression.

```

1360 [h1] <?xml version="1.0" encoding="UTF-8"?>
1361 [h2] <Policy
1362 [h3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1363 [h4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1364 [h5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```

1365 [h6]      xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
1366 http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
1367 [h7]      xmlns:md="http://www.med.example.com/schemas/record.xsd"
1368 [h8]      PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:3"
1369 [h9]      Version="1.0"
1370 [h10]     RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1371 algorithm:deny-overrides">
1372 [h11]     <Description>
1373 [h12]       Policy for any medical record in the
1374 [h13]       http://www.med.example.com/schemas/record.xsd namespace
1375 [h14]     </Description>
1376 [h15]     <PolicyDefaults>
1377 [h16]       <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1378 [h17]     </PolicyDefaults>
1379 [h18]     <Target>
1380 [h19]       <AnyOf>
1381 [h20]         <AllOf>
1382 [h21]           <Match
1383 [h22]             MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1384 [h23]               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1385 [h24]                 >urn:example:med:schemas:record</AttributeValue>
1386 [h25]               <AttributeDesignator
1387 [h26]                 MustBePresent="false"
1388 [h27]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1389 [h28]                 AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1390 [h29]                 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1391 [h30]             </Match>
1392 [h31]           </AllOf>
1393 [h32]         </AnyOf>
1394 [h33]     </Target>
1395 [h34]     <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:3"
1396 [h35]       Effect="Permit">
1397 [h36]       <Description>
1398 [h37]         A physician may write any medical element in a record
1399 [h38]         for which he or she is the designated primary care
1400 [h39]         physician, provided an email is sent to the patient
1401 [h40]       </Description>
1402 [h41]       <Target>
1403 [h42]         <AnyOf>
1404 [h43]           <AllOf>
1405 [h44]             <Match
1406 [h45]               MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1407 [h46]                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1408 [h47]                   >physician</AttributeValue>
1409 [h48]                 <AttributeDesignator
1410 [h49]                   MustBePresent="false"
1411 [h50]                   Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1412 [h51]                   AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
1413 [h52]                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
1414 [h53]                 </Match>
1415 [h54]             </AllOf>
1416 [h55]           </AnyOf>
1417 [h56]         <AnyOf>
1418 [h57]           <AllOf>
1419 [h58]             <Match
1420 [h59]               MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1421 [h60]                 <AttributeValue
1422 [h61]                   DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1423 [h62]                   XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1424 [h63]                   >md:record/md:medical</AttributeValue>
1425 [h64]                 <AttributeDesignator
1426 [h65]                   MustBePresent="false"
1427 [h66]                   Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1428 [h67]                   AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1429 [h68]                   DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1430 [h69]                 </Match>
1431 [h70]             </AllOf>
1432 [h71]           </AnyOf>
1433 [h72]         <AnyOf>
1434 [h73]           <AllOf>
1435 [h74]             <Match
1436 [h75]               MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1437 [h76]                 <AttributeValue

```

```

1438 [h77]         DataType="http://www.w3.org/2001/XMLSchema#string"
1439 [h78]         >write</AttributeValue>
1440 [h79]         <AttributeDesignator
1441 [h80]             MustBePresent="false"
1442 [h81]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1443 [h82]             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1444 [h83]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1445 [h84]         </Match>
1446 [h85]     </AllOf>
1447 [h86] </AnyOf>
1448 [h87] </Target>
1449 [h88] <Condition>
1450 [h89]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1451 [h90]         <Apply
1452 [h91]             FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1453 [h92]             <AttributeDesignator
1454 [h93]                 MustBePresent="false"
1455 [h94]                 Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1456 [h95]                 AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id"
1457 [h96]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1458 [h97]             </Apply>
1459 [h98]         <Apply
1460 [h99]             FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1461 [h100]            <AttributeSelector
1462 [h101]                MustBePresent="false"
1463 [h102]                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1464 [h103]                Path="md:record/md:primaryCarePhysician/md:registrationID/text()"
1465 [h104]                DataType="http://www.w3.org/2001/XMLSchema#string"/>
1466 [h105]            </Apply>
1467 [h106]        </Apply>
1468 [h107]    </Condition>
1469 [h108] </Rule>
1470 [h109] <ObligationExpressions>
1471 [h110]     <ObligationExpression
1472 [h111]         ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
1473 [h112]         FulfillOn="Permit">
1474 [h113]         <AttributeAssignmentExpression
1475 [h114]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:mailto">
1476 [h115]             <AttributeSelector
1477 [h116]                 MustBePresent="true"
1478 [h117]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1479 [h118]                 Path="md:record/md:patient/md:patientContact/md:email"
1480 [h119]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1481 [h120]             </AttributeAssignmentExpression>
1482 [h121]         <AttributeAssignmentExpression
1483 [h122]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
1484 [h123]             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1485 [h124]                 >Your medical record has been accessed by:</AttributeValue>
1486 [h125]             </AttributeAssignmentExpression>
1487 [h126]         <AttributeAssignmentExpression
1488 [h127]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
1489 [h128]             <AttributeDesignator
1490 [h129]                 MustBePresent="false"
1491 [h130]                 Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1492 [h131]                 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1493 [h132]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1494 [h133]             </AttributeAssignmentExpression>
1495 [h134]         </ObligationExpression>
1496 [h135]     </ObligationExpressions>
1497 [h136] </Policy>

```

1498 [h2] - [h10] The <Policy> element includes standard namespace declarations as well as **policy** specific
1499 parameters, such as PolicyId and RuleCombiningAlgId.

1500 [h8] **Policy** identifier. This parameter allows the **policy** to be referenced by a **policy set**.

1501 [h10] The **Rule-combining algorithm** identifies the algorithm for combining the outcomes of **rule**
1502 evaluation.

1503 [h11] - [h14] Free-form description of the **policy**.

1504 [h18] - [h33] **Policy target**. The **policy target** defines a set of applicable **decision requests**. The
1505 structure of the <Target> element in the <Policy> is identical to the structure of the <Target>

1506 element in the <Rule>. In this case, the **policy target** is the set of all XML **resources** that conform to
1507 the namespace “urn:example:med:schemas:record”.

1508 [h34] - [h108] The only <Rule> element included in this <Policy>. Two parameters are specified in the
1509 **rule** header: RuleId and Effect.

1510 [h41] - [h87] The **rule target** further constrains the **policy target**.

1511 [h44] - [h53] The <Match> element targets the **rule** at **subjects** whose
1512 “urn:oasis:names:tc:xacml:3.0:example:attribute:role” **subject attribute** is equal to “physician”.

1513 [h58] - [h69] The <Match> element targets the **rule** at **resources** that match the XPath expression
1514 “md:record/md:medical”.

1515 [h74] - [h84] The <Match> element targets the **rule** at **actions** whose
1516 “urn:oasis:names:tc:xacml:1.0:action:action-id” **action attribute** is equal to “write”.

1517 [h88] - [h107] The <Condition> element. For the **rule** to be applicable to the **decision request**, the
1518 **condition** must evaluate to “True”. This **condition** compares the value of the
1519 “urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id” **subject attribute** with the value of the
1520 <registrationId> element in the medical record that is being accessed.

1521 [h109] - [h134] The <ObligationExpressions> element. **Obligations** are a set of operations that
1522 must be performed by the **PEP** in conjunction with an **authorization decision**. An **obligation** may be
1523 associated with a “Permit” or “Deny” **authorization decision**. The element contains a single **obligation**
1524 expression, which will be evaluated into an obligation when the policy is evaluated.

1525 [h110] - [h133] The <ObligationExpression> element consists of the ObligationId attribute, the
1526 **authorization decision** value for which it must be fulfilled, and a set of **attribute** assignments.

1527 [h110] The ObligationId attribute identifies the **obligation**. In this case, the **PEP** is required to send
1528 email.

1529 [h111] The FulfillOn attribute defines the **authorization decision** value for which the **obligation**
1530 derived from the **obligation** expression must be fulfilled. In this case, the **obligation** must be fulfilled
1531 when **access** is permitted.

1532 [h112] - [h119] The first parameter indicates where the **PEP** will find the email address in the **resource**.
1533 The **PDP** will evaluate the <AttributeSelector> and return the result to the **PEP** inside the resulting
1534 **obligation**.

1535 [h120] - [h123] The second parameter contains literal text for the email body.

1536 [h125] - [h132] The third parameter indicates where the **PEP** will find further text for the email body in the
1537 **resource**. The **PDP** will evaluate the <AttributeDesignator> and return the result to the **PEP** inside
1538 the resulting **obligation**.

1539 4.2.4.4 Rule 4

1540 **Rule 4** illustrates the use of the “Deny” **Effect** value, and a <Rule> with no <Condition> element.

```

1541 [i1] <?xml version="1.0" encoding="UTF-8"?>
1542 [i2] <Policy
1543 [i3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1544 [i4]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1545 [i5]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1546 [i6]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:4"
1547 [i7]   Version="1.0"
1548 [i8]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1549 [i8]   algorithm:deny-overrides">
1550 [i9]   <PolicyDefaults>
1551 [i10]     <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1552 [i11]   </PolicyDefaults>
1553 [i12]   <Target/>
1554 [i13]   <Rule
1555 [i14]     RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
1556 [i15]     Effect="Deny">
1557 [i16]     <Description>
1558 [i17]       An Administrator shall not be permitted to read or write

```

```

1559 [i18] medical elements of a patient record in the
1560 [i19] http://www.med.example.com/records.xsd namespace.
1561 [i20] </Description>
1562 [i21] <Target>
1563 [i22] <AnyOf>
1564 [i23] <AllOf>
1565 [i24] <Match
1566 [i25] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1567 [i26] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1568 [i27] >administrator</AttributeValue>
1569 [i28] <AttributeDesignator
1570 [i29] MustBePresent="false"
1571 [i30] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1572 [i31] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
1573 [i32] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1574 [i33] </Match>
1575 [i34] </AllOf>
1576 [i35] </AnyOf>
1577 [i36] <AnyOf>
1578 [i37] <AllOf>
1579 [i38] <Match
1580 [i39] MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1581 [i40] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1582 [i41] >urn:example:med:schemas:record</AttributeValue>
1583 [i42] <AttributeDesignator
1584 [i43] MustBePresent="false"
1585 [i44] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1586 [i45] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1587 [i46] DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1588 [i47] </Match>
1589 [i48] <Match
1590 [i49] MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1591 [i50] <AttributeValue
1592 [i51] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1593 [i52] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1594 [i53] >md:record/md:medical</AttributeValue>
1595 [i54] <AttributeDesignator
1596 [i55] MustBePresent="false"
1597 [i56] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1598 [i57] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1599 [i58] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1600 [i59] </Match>
1601 [i60] </AllOf>
1602 [i61] </AnyOf>
1603 [i62] <AnyOf>
1604 [i63] <AllOf>
1605 [i64] <Match
1606 [i65] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1607 [i66] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1608 [i67] >read</AttributeValue>
1609 [i68] <AttributeDesignator
1610 [i69] MustBePresent="false"
1611 [i70] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1612 [i71] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1613 [i72] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1614 [i73] </Match>
1615 [i74] </AllOf>
1616 [i75] <AllOf>
1617 [i76] <Match
1618 [i77] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1619 [i78] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1620 [i79] >write</AttributeValue>
1621 [i80] <AttributeDesignator
1622 [i81] MustBePresent="false"
1623 [i82] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1624 [i83] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1625 [i84] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1626 [i85] </Match>
1627 [i86] </AllOf>
1628 [i87] </AnyOf>
1629 [i88] </Target>
1630 [i89] </Rule>
1631 [i90] </Policy>

```


1632 [i13] - [i15] The <Rule> element declaration.

1633 [i15] **Rule Effect.** Every **rule** that evaluates to “True” emits the **rule effect** as its value. This **rule**
 1634 **Effect** is “Deny” meaning that according to this **rule**, **access** must be denied when it evaluates to
 1635 “True”.

1636 [i16] - [i20] Free form description of the **rule**.

1637 [i21] - [i88] **Rule target.** The **Rule target** defines the set of **decision requests** that are applicable to the
 1638 **rule**.

1639 [i24] - [i33] The <Match> element targets the **rule** at **subjects** whose
 1640 “urn:oasis:names:tc:xacml:3.0:example:attribute:role” **subject attribute** is equal to “administrator”.

1641 [i36] - [i61] The <AnyOf> element contains one <AllOf> element, which (in turn) contains two <Match>
 1642 elements. The **target** matches if the **resource** identified by the request **context** matches both **resource**
 1643 match criteria.

1644 [i38] - [i47] The first <Match> element targets the **rule** at **resources** whose
 1645 “urn:oasis:names:tc:xacml:2.0:resource:target-namespace” **resource attribute** is equal to
 1646 “urn:example:med:schemas:record”.

1647 [i48] - [i59] The second <Match> element targets the **rule** at XML elements that match the XPath
 1648 expression “/md:record/md:medical”.

1649 [i62] - [i87] The <AnyOf> element contains two <AllOf> elements, each of which contains one <Match>
 1650 element. The **target** matches if the **action** identified in the request **context** matches either of the **action**
 1651 match criteria.

1652 [i64] - [i85] The <Match> elements **target** the **rule** at **actions** whose
 1653 “urn:oasis:names:tc:xacml:1.0:action:action-id” **action attribute** is equal to “read” or “write”.

1654 This **rule** does not have a <Condition> element.

1655 4.2.4.5 Example PolicySet

1656 This section uses the examples of the previous sections to illustrate the process of combining **policies**.
 1657 The **policy** governing read **access** to medical elements of a record is formed from each of the four **rules**
 1658 described in Section 4.2.3. In plain language, the combined **rule** is:

- 1659 • Either the requestor is the patient; or
- 1660 • the requestor is the parent or guardian and the patient is under 16; or
- 1661 • the requestor is the primary care physician and a notification is sent to the patient; and
- 1662 • the requestor is not an administrator.

1663 The following **policy set** illustrates the combined **policies**. **Policy 3** is included by reference and **policy**
 1664 **2** is explicitly included.

```

1665 [j1]    <?xml version="1.0" encoding="UTF-8"?>
1666 [j2]    <PolicySet
1667 [j3]      xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1668 [j4]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1669 [j5]      PolicySetId="urn:oasis:names:tc:xacml:3.0:example:policysetid:1"
1670 [j6]      Version="1.0"
1671 [j7]      PolicyCombiningAlgId=
1672 [j8]      "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides">
1673 [j9]    <Description>
1674 [j10]     Example policy set.
1675 [j11]    </Description>
1676 [j12]    <Target>
1677 [j13]     <AnyOf>
1678 [j14]       <AllOf>
1679 [j15]         <Match
1680 [j16]           MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1681 [j17]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1682 [j18]             >urn:example:med:schema:records</AttributeValue>
1683 [j19]           <AttributeDesignator
1684 [j20]             MustBePresent="false"

```

```

1685 [j21]         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1686 [j22]         AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1687 [j23]         DataType="http://www.w3.org/2001/XMLSchema#string"/>
1688 [j24]         </Match>
1689 [j25]         </AllOf>
1690 [j26]         </AnyOf>
1691 [j27]         </Target>
1692 [j28]         <PolicyIdReference>
1693 [j29]           urn:oasis:names:tc:xacml:3.0:example:policyid:3
1694 [j30]         </PolicyIdReference>
1695 [j31]         <Policy
1696 [j32]           PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
1697 [j33]           RuleCombiningAlgId=
1698 [j34]             "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides"
1699 [j35]           Version="1.0">
1700 [j36]           <Target/>
1701 [j37]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
1702 [j38]             Effect="Permit">
1703 [j39]           </Rule>
1704 [j40]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
1705 [j41]             Effect="Permit">
1706 [j42]           </Rule>
1707 [j43]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
1708 [j44]             Effect="Deny">
1709 [j45]           </Rule>
1710 [j46]         </Policy>
1711 [j47]       </PolicySet>

```

1712 [j2] - [j8] The `<PolicySet>` element declaration. Standard XML namespace declarations are included.

1713 [j5] The `PolicySetId` attribute is used for identifying this **policy set** for possible inclusion in another
1714 **policy set**.

1715 [j7] - [j8] The **policy-combining algorithm** identifier. **Policies** and **policy sets** in this **policy set** are
1716 combined according to the specified **policy-combining algorithm** when the **authorization decision** is
1717 computed.

1718 [j9] - [j11] Free form description of the **policy set**.

1719 [j12] - [j27] The **policy set** `<Target>` element defines the set of **decision requests** that are applicable to
1720 this `<PolicySet>` element.

1721 [j28] - [j30] `PolicyIdReference` includes a **policy** by id.

1722 [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>
```

```

1772 <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
1773 <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
1774 <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
1775 <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
1776 </xs:complexType>

```

1777 The <PolicySet> element is of PolicySetType complex type.

1778 The <PolicySet> element contains the following attributes and elements:

1779 PolicySetId [Required]

1780 **Policy set** identifier. It is the responsibility of the **PAP** to ensure that no two **policies** visible to
1781 the **PDP** have the same identifier. This MAY be achieved by following a predefined URN or URI
1782 scheme. If the **policy set** identifier is in the form of a URL, then it MAY be resolvable.

1783 Version [Required]

1784 The version number of the PolicySet.

1785 PolicyCombiningAlgId [Required]

1786 The identifier of the **policy-combining algorithm** by which the <PolicySet>,
1787 <CombinerParameters>, <PolicyCombinerParameters> and
1788 <PolicySetCombinerParameters> components MUST be combined. Standard **policy-**
1789 **combining algorithms** are listed in Appendix Appendix C. Standard **policy-combining**
1790 **algorithm** identifiers are listed in Section B.9.

1791 MaxDelegationDepth [Optional]

1792 If present, limits the depth of delegation which is authorized by this **policy set**. See the delegation
1793 profile [XACMLAdmin].

1794 <Description> [Optional]

1795 A free-form description of the **policy set**.

1796 <PolicyIssuer> [Optional]

1797 **Attributes** of the **issuer** of the **policy set**.

1798 <PolicySetDefaults> [Optional]

1799 A set of default values applicable to the **policy set**. The scope of the <PolicySetDefaults>
1800 element SHALL be the enclosing **policy set**.

1801 <Target> [Required]

1802 The <Target> element defines the applicability of a **policy set** to a set of **decision requests**.

1803 The <Target> element MAY be declared by the creator of the <PolicySet> or it MAY be computed
1804 from the <Target> elements of the referenced <Policy> elements, either as an intersection or
1805 as a union.

1806 <PolicySet> [Any Number]

1807 A **policy set** that is included in this **policy set**.

1808 <Policy> [Any Number]

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

1810 <PolicySetIdReference> [Any Number]

1811 A reference to a **policy set** that MUST be included in this **policy set**. If
1812 <PolicySetIdReference> is a URL, then it MAY be resolvable.

1813 <PolicyIdReference> [Any Number]

1814 A reference to a **policy** that MUST be included in this **policy set**. If the
1815 <PolicyIdReference> is a URL, then it MAY be resolvable.

- 1816 <ObligationExpressions> [Optional]
 1817 Contains the set of <ObligationExpression> elements. See Section 7.18 for a description of
 1818 how the set of **obligations** to be returned by the **PDP** shall be determined.
- 1819 <AdviceExpressions> [Optional]
 1820 Contains the set of <AdviceExpression> elements. See Section 7.18 for a description of how
 1821 the set of **advice** to be returned by the **PDP** shall be determined.
- 1822 <CombinerParameters> [Optional]
 1823 Contains a sequence of <CombinerParameter> elements. The parameters apply to the
 1824 combining algorithm as such and it is up to the specific combining algorithm to interpret them and
 1825 adjust its behavior accordingly.
- 1826 <PolicyCombinerParameters> [Optional]
 1827 Contains a sequence of <CombinerParameter> elements that are associated with a particular
 1828 <Policy> or <PolicyIdReference> element within the <PolicySet>. It is up to the specific
 1829 combining algorithm to interpret them and adjust its behavior accordingly.
- 1830 <PolicySetCombinerParameters> [Optional]
 1831 Contains a sequence of <CombinerParameter> elements that are associated with a particular
 1832 <PolicySet> or <PolicySetIdReference> element within the <PolicySet>. It is up to the
 1833 specific combining algorithm to interpret them and adjust its behavior accordingly.

1834 5.2 Element <Description>

- 1835 The <Description> element contains a free-form description of the <PolicySet>, <Policy>,
 1836 <Rule> or <Apply> element. The <Description> element is of xs:string simple type.

```
1837 <xs:element name="Description" type="xs:string"/>
```

1838 5.3 Element <PolicyIssuer>

- 1839 The <PolicyIssuer> element contains **attributes** describing the issuer of the **policy** or **policy set**.
 1840 The use of the **policy** issuer element is defined in a separate administration profile [**XACMLAdmin**]. A
 1841 PDP which does not implement the administration profile MUST report an error or return an Indeterminate
 1842 result if it encounters this element.

```
1843 <xs:element name="PolicyIssuer" type="xacml:PolicyIssuerType"/>
1844 <xs:complexType name="PolicyIssuerType">
1845   <xs:sequence>
1846     <xs:element ref="xacml:Content" minOccurs="0"/>
1847     <xs:element ref="xacml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
1848   </xs:sequence>
1849 </xs:complexType>
```

- 1850 The <PolicyIssuer> element is of PolicyIssuerType complex type.

- 1851 The <PolicyIssuer> element contains the following elements:

1852 <Content> [Optional]

- 1853 Free form XML describing the issuer. See Section 5.45.

1854 <Attribute> [Zero to many]

- 1855 An **attribute** of the issuer. See Section 5.46.

1856 5.4 Element <PolicySetDefaults>

- 1857 The <PolicySetDefaults> element SHALL specify default values that apply to the <PolicySet>
 1858 element.

```

1859 <xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
1860 <xs:complexType name="DefaultsType">
1861   <xs:sequence>
1862     <xs:choice>
1863       <xs:element ref="xacml:XPathVersion">
1864     </xs:choice>
1865   </xs:sequence>
1866 </xs:complexType>

```

1867 <PolicySetDefaults> element is of DefaultsType complex type.

1868 The <PolicySetDefaults> element contains the following elements:

1869 <XPathVersion> [Optional]

1870 Default XPath version.

1871 5.5 Element <XPathVersion>

1872 The <XPathVersion> element SHALL specify the version of the XPath specification to be used by
1873 <AttributeSelector> elements and XPath-based functions in the **policy set** or **policy**.

```

1874 <xs:element name="XPathVersion" type="xs:anyURI"/>

```

1875 The URI for the XPath 1.0 specification is "http://www.w3.org/TR/1999/REC-xpath-19991116".

1876 The URI for the XPath 2.0 specification is "http://www.w3.org/TR/2007/REC-xpath20-20070123".

1877 The <XPathVersion> element is REQUIRED if the XACML enclosing **policy set** or **policy** contains
1878 <AttributeSelector> elements or XPath-based functions.

1879 5.6 Element <Target>

1880 The <Target> element identifies the set of **decision requests** that the parent element is intended to
1881 evaluate. The <Target> element SHALL appear as a child of a <PolicySet> and <Policy> element
1882 and MAY appear as a child of a <Rule> element.

1883 The <Target> element SHALL contain a **conjunctive sequence** of <AnyOf> elements. For the parent
1884 of the <Target> element to be applicable to the **decision request**, there MUST be at least one positive
1885 match between each <AnyOf> element of the <Target> element and the corresponding section of the
1886 <Request> element.

```

1887 <xs:element name="Target" type="xacml:TargetType"/>
1888 <xs:complexType name="TargetType">
1889   <xs:sequence minOccurs="0" maxOccurs="unbounded">
1890     <xs:element ref="xacml:AnyOf"/>
1891   </xs:sequence>
1892 </xs:complexType>

```

1893 The <Target> element is of TargetType complex type.

1894 The <Target> element contains the following elements:

1895 <AnyOf> [Zero to Many]

1896 Matching specification for **attributes** in the **context**. If this element is missing, then the **target**
1897 SHALL match all **contexts**.

1898 5.7 Element <AnyOf>

1899 The <AnyOf> element SHALL contain a **disjunctive sequence** of <AllOf> elements.

```

1900 <xs:element name="AnyOf" type="xacml:AnyOfType"/>
1901 <xs:complexType name="AnyOfType">
1902   <xs:sequence minOccurs="1" maxOccurs="unbounded">
1903     <xs:element ref="xacml:AllOf"/>

```

1904 `</xs:sequence>`
1905 `</xs:complexType>`

1906 The `<AnyOf>` element is of `AnyOfType` complex type.

1907 The `<AnyOf>` element contains the following elements:

1908 `<AllOf>` [One to Many, Required]

1909 See Section 5.8.

1910 5.8 Element `<AllOf>`

1911 The `<AllOf>` element SHALL contain a **conjunctive sequence** of `<Match>` elements.

```
1912 <xs:element name="AllOf" type="xacml:AllOfType"/>
1913 <xs:complexType name="AllOfType">
1914   <xs:sequence minOccurs="1" maxOccurs="unbounded">
1915     <xs:element ref="xacml:Match"/>
1916   </xs:sequence>
1917 </xs:complexType>
```

1918 The `<AllOf>` element is of `AllOfType` complex type.

1919 The `<AllOf>` element contains the following elements:

1920 `<Match>` [One to Many]

1921 A **conjunctive sequence** of individual matches of the **attributes** in the request **context** and the
1922 embedded **attribute** values. See Section 5.9.

1923 5.9 Element `<Match>`

1924 The `<Match>` element SHALL identify a set of entities by matching **attribute** values in an

1925 `<Attributes>` element of the request **context** with the embedded **attribute** value.

```
1926 <xs:element name="Match" type="xacml:MatchType"/>
1927 <xs:complexType name="MatchType">
1928   <xs:sequence>
1929     <xs:element ref="xacml:AttributeValue"/>
1930     <xs:choice>
1931       <xs:element ref="xacml:AttributeDesignator"/>
1932       <xs:element ref="xacml:AttributeSelector"/>
1933     </xs:choice>
1934   </xs:sequence>
1935   <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
1936 </xs:complexType>
```

1937 The `<Match>` element is of `MatchType` complex type.

1938 The `<Match>` element contains the following attributes and elements:

1939 `MatchId` [Required]

1940 Specifies a matching function. The value of this attribute MUST be of type `xs:anyURI` with legal
1941 values documented in Section 7.6.

1942 `<AttributeValue>` [Required]

1943 Embedded **attribute** value.

1944 `<AttributeDesignator>` [Required choice]

1945 MAY be used to identify one or more **attribute** values in an `<Attributes>` element of the
1946 request **context**.

1947 `<AttributeSelector>` [Required choice]

1948 MAY be used to identify one or more **attribute** values in a <Content> element of the request
1949 **context**.

1950 5.10 Element <PolicySetIdReference>

1951 The <PolicySetIdReference> element SHALL be used to reference a <PolicySet> element by id.
1952 If <PolicySetIdReference> is a URL, then it MAY be resolvable to the <PolicySet> element.
1953 However, the mechanism for resolving a **policy set** reference to the corresponding **policy set** is outside
1954 the scope of this specification.

```
1955 <xs:element name="PolicySetIdReference" type="xacml:IdReferenceType"/>  
1956 <xs:complexType name="IdReferenceType">  
1957   <xs:simpleContent>  
1958     <xs:extension base="xs:anyURI">  
1959       <xs:attribute name="xacml:Version"  
1960         type="xacml:VersionMatchType" use="optional"/>  
1961       <xs:attribute name="xacml:EarliestVersion"  
1962         type="xacml:VersionMatchType" use="optional"/>  
1963       <xs:attribute name="xacml:LatestVersion"  
1964         type="xacml:VersionMatchType" use="optional"/>  
1965     </xs:extension>  
1966   </xs:simpleContent>  
1967 </xs:complexType>
```

1968 Element <PolicySetIdReference> is of `xacml:IdReferenceType` complex type.

1969 `IdReferenceType` extends the `xs:anyURI` type with the following attributes:

1970 `Version` [Optional]

1971 Specifies a matching expression for the version of the **policy set** referenced.

1972 `EarliestVersion` [Optional]

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

1974 `LatestVersion` [Optional]

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

1976 The matching operation is defined in Section 5.13. Any combination of these attributes MAY be present
1977 in a <PolicySetIdReference>. The referenced **policy set** MUST match all expressions. If none of
1978 these attributes is present, then any version of the **policy set** is acceptable. In the case that more than
1979 one matching version can be obtained, then the most recent one SHOULD be used.

1980 5.11 Element <PolicyIdReference>

1981 The <PolicyIdReference> element SHALL be used to reference a <Policy> element by id. If
1982 <PolicyIdReference> is a URL, then it MAY be resolvable to the <Policy> element. However, the
1983 mechanism for resolving a **policy** reference to the corresponding **policy** is outside the scope of this
1984 specification.

```
1985 <xs:element name="PolicyIdReference" type="xacml:IdReferenceType"/>
```

1986 Element <PolicyIdReference> is of `xacml:IdReferenceType` complex type (see Section 5.10) .

1987 5.12 Simple type VersionType

1988 Elements of this type SHALL contain the version number of the **policy** or **policy set**.

```
1989 <xs:simpleType name="VersionType">  
1990   <xs:restriction base="xs:string">  
1991     <xs:pattern value="(\d+\.)*\d+"/>  
1992   </xs:restriction>  
1993 </xs:simpleType>
```


1994 The version number is expressed as a sequence of decimal numbers, each separated by a period (.).
1995 'd+' represents a sequence of one or more decimal digits.

1996 5.13 Simple type VersionMatchType

1997 Elements of this type SHALL contain a restricted regular expression matching a version number (see
1998 Section 5.12). The expression SHALL match versions of a referenced **policy** or **policy set** that are
1999 acceptable for inclusion in the referencing **policy** or **policy set**.

```
2000 <xs:simpleType name="VersionMatchType">  
2001   <xs:restriction base="xs:string">  
2002     <xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)" />  
2003   </xs:restriction>  
2004 </xs:simpleType>
```

2005 A version match is '.'-separated, like a version string. A number represents a direct numeric match. A '*'
2006 means that any single number is valid. A '+' means that any number, and any subsequent numbers, are
2007 valid. In this manner, the following four patterns would all match the version string '1.2.3': '1.2.3', '1.*.3',
2008 '1.2.*' and '1.+'

2009 5.14 Element <Policy>

2010 The <Policy> element is the smallest entity that SHALL be presented to the **PDP** for evaluation.

2011 A <Policy> element may be evaluated, in which case the evaluation procedure defined in Section 7.12
2012 SHALL be used.

2013 The main components of this element are the <Target>, <Rule>, <CombinerParameters>,
2014 <RuleCombinerParameters>, <ObligationExpressions> and <AdviceExpressions>
2015 elements and the RuleCombiningAlgId attribute.

2016 A <Policy> element MAY contain a <PolicyIssuer> element. The interpretation of the
2017 <PolicyIssuer> element is explained in the separate administrative **policy** profile [XACMLAdmin].

2018 The <Target> element defines the applicability of the <Policy> element to a set of **decision requests**.
2019 If the <Target> element within the <Policy> element matches the request **context**, then the
2020 <Policy> element MAY be used by the **PDP** in making its **authorization decision**. See Section 7.12.

2021 The <Policy> element includes a sequence of choices between <VariableDefinition> and
2022 <Rule> elements.

2023 **Rules** included in the <Policy> element MUST be combined by the algorithm specified by the
2024 RuleCombiningAlgId attribute.

2025 The <ObligationExpressions> element contains a set of **obligation** expressions that MUST be
2026 evaluated into **obligations** by the **PDP** and the resulting **obligations** MUST be fulfilled by the **PEP** in
2027 conjunction with the **authorization decision**. If the **PEP** does not understand, or cannot fulfill, any of the
2028 **obligations**, then it MUST act according to the PEP bias. See Section 7.2 and 7.18.

2029 The <AdviceExpressions> element contains a set of **advice** expressions that MUST be evaluated into
2030 **advice** by the **PDP**. The resulting **advice** MAY be safely ignored by the **PEP** in conjunction with the
2031 **authorization decision**. See Section 7.18.

```
2032 <xs:element name="Policy" type="xacml:PolicyType"/>  
2033 <xs:complexType name="PolicyType">  
2034   <xs:sequence>  
2035     <xs:element ref="xacml:Description" minOccurs="0"/>  
2036     <xs:element ref="xacml:PolicyIssuer" minOccurs="0"/>  
2037     <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>  
2038     <xs:element ref="xacml:Target"/>  
2039     <xs:choice maxOccurs="unbounded">  
2040       <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>  
2041       <xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>  
2042       <xs:element ref="xacml:VariableDefinition"/>
```

```

2043         <xs:element ref="xacml:Rule"/>
2044     </xs:choice>
2045     <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
2046     <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
2047 </xs:sequence>
2048 <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
2049 <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
2050 <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
2051 <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
2052 </xs:complexType>

```

2053 The <Policy> element is of PolicyType complex type.

2054 The <Policy> element contains the following attributes and elements:

2055 PolicyId [Required]

2056 **Policy** identifier. It is the responsibility of the **PAP** to ensure that no two **policies** visible to the
2057 **PDP** have the same identifier. This MAY be achieved by following a predefined URN or URI
2058 scheme. If the **policy** identifier is in the form of a URL, then it MAY be resolvable.

2059 Version [Required]

2060 The version number of the **Policy**.

2061 RuleCombiningAlgId [Required]

2062 The identifier of the **rule-combining algorithm** by which the <Policy>,
2063 <CombinerParameters> and <RuleCombinerParameters> components MUST be
2064 combined. Standard **rule-combining algorithms** are listed in Appendix Appendix C. Standard
2065 **rule-combining algorithm** identifiers are listed in Section B.9.

2066 MaxDelegationDepth [Optional]

2067 If present, limits the depth of delegation which is authorized by this **policy**. See the delegation
2068 profile [**XACMLAdmin**].

2069 <Description> [Optional]

2070 A free-form description of the **policy**. See Section 5.2.

2071 <PolicyIssuer> [Optional]

2072 **Attributes** of the **issuer** of the **policy**.

2073 <PolicyDefaults> [Optional]

2074 Defines a set of default values applicable to the **policy**. The scope of the <PolicyDefaults>
2075 element SHALL be the enclosing **policy**.

2076 <CombinerParameters> [Optional]

2077 A sequence of parameters to be used by the **rule-combining algorithm**. The parameters apply
2078 to the combining algorithm as such and it is up to the specific combining algorithm to interpret
2079 them and adjust its behavior accordingly.

2080 <RuleCombinerParameters> [Optional]

2081 A sequence of <RuleCombinerParameter> elements that are associated with a particular
2082 <Rule> element within the <Policy>.. It is up to the specific combining algorithm to interpret
2083 them and adjust its behavior accordingly.

2084 <Target> [Required]

2085 The <Target> element defines the applicability of a <Policy> to a set of **decision requests**.

2086 The <Target> element MAY be declared by the creator of the <Policy> element, or it MAY be
2087 computed from the <Target> elements of the referenced <Rule> elements either as an
2088 intersection or as a union.

2089 <VariableDefinition> [Any Number]
 2090 Common function definitions that can be referenced from anywhere in a *rule* where an
 2091 expression can be found.

2092 <Rule> [Any Number]
 2093 A sequence of *rules* that MUST be combined according to the `RuleCombiningAlgId` attribute.
 2094 *Rules* whose <Target> elements and conditions match the *decision request* MUST be
 2095 considered. *Rules* whose <Target> elements or conditions do not match the *decision request*
 2096 SHALL be ignored.

2097 <ObligationExpressions> [Optional]
 2098 A *conjunctive sequence* of *obligation* expressions which MUST be evaluated into *obligations*
 2099 by the PDP. The corresponding *obligations* MUST be fulfilled by the *PEP* in conjunction with
 2100 the *authorization decision*. See Section 7.18 for a description of how the set of *obligations* to
 2101 be returned by the *PDP* SHALL be determined. See section 7.2 about enforcement of
 2102 *obligations*.

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

2108 5.15 Element <PolicyDefaults>

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

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

2118 <PolicyDefaults> element is of DefaultsType complex type.

2119 The <PolicyDefaults> element contains the following elements:

2120 <XPathVersion> [Optional]

2121 Default XPath version.

2122 5.16 Element <CombinerParameters>

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

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

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

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

2136 `</xs:complexType>`

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

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

2139 `<CombinerParameter>` [Any Number]

2140 A single parameter. See Section 5.17.

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

2142 5.17 Element `<CombinerParameter>`

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

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

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

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

2154 `ParameterName` [Required]

2155 The identifier of the parameter.

2156 `<AttributeValue>` [Required]

2157 The value of the parameter.

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

2159 5.18 Element `<RuleCombinerParameters>`

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

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

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

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

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

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

2185 **5.19 Element <PolicyCombinerParameters>**

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

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

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

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

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

2208 The <PolicyCombinerParameters> element contains the following attribute:

2209 PolicyIdRef [Required]

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

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

2214 **5.20 Element <PolicySetCombinerParameters>**

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

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

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

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

```

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

```

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

2237 The <PolicySetCombinerParameters> element contains the following attribute:

2238 PolicySetIdRef [Required]

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

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

2243 5.21 Element <Rule>

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

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

```

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

```

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

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

2263 RuleId [Required]

2264 A string identifying this **rule**.

2265 Effect [Required]

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

2267 <Description> [Optional]

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

2269 <Target> [Optional]

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

2273 <Condition> [Optional]

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

2275 <ObligationExpressions> [Optional]

2276 A **conjunctive sequence** of **obligation** expressions which MUST be evaluated into **obligations**
2277 by the PDP. The corresponding **obligations** MUST be fulfilled by the **PEP** in conjunction with
2278 the **authorization decision**. See Section 7.18 for a description of how the set of **obligations** to
2279 be returned by the **PDP** SHALL be determined. See section 7.2 about enforcement of
2280 **obligations**.

2281 <AdviceExpressions> [Optional]

2282 A **conjunctive sequence** of **advice** expressions which MUST evaluated into **advice** by the **PDP**.
2283 The corresponding **advice** provide supplementary information to the **PEP** in conjunction with the
2284 **authorization decision**. See Section 7.18 for a description of how the set of **advice** to be
2285 returned by the **PDP** SHALL be determined.

2286 5.22 Simple type EffectType

2287 The EffectType simple type defines the values allowed for the Effect attribute of the <Rule> element
2288 and for the FulfillOn attribute of the <ObligationExpression> and <AdviceExpression>
2289 elements.

```
2290 <xs:simpleType name="EffectType">  
2291   <xs:restriction base="xs:string">  
2292     <xs:enumeration value="Permit"/>  
2293     <xs:enumeration value="Deny"/>  
2294   </xs:restriction>  
2295 </xs:simpleType>
```

2296 5.23 Element <VariableDefinition>

2297 The <VariableDefinition> element SHALL be used to define a value that can be referenced by a
2298 <VariableReference> element. The name supplied for its VariableId attribute SHALL NOT occur
2299 in the VariableId attribute of any other <VariableDefinition> element within the encompassing
2300 **policy**. The <VariableDefinition> element MAY contain undefined <VariableReference>
2301 elements, but if it does, a corresponding <VariableDefinition> element MUST be defined later in
2302 the encompassing **policy**. <VariableDefinition> elements MAY be grouped together or MAY be
2303 placed close to the reference in the encompassing **policy**. There MAY be zero or more references to
2304 each <VariableDefinition> element.

```
2305 <xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>  
2306 <xs:complexType name="VariableDefinitionType">  
2307   <xs:sequence>  
2308     <xs:element ref="xacml:Expression"/>  
2309   </xs:sequence>  
2310   <xs:attribute name="VariableId" type="xs:string" use="required"/>  
2311 </xs:complexType>
```

2312 The <VariableDefinition> element is of VariableDefinitionType complex type. The
2313 <VariableDefinition> element has the following elements and attributes:

2314 <Expression> [Required]

2315 Any element of ExpressionType complex type.

2316 VariableId [Required]

2317 The name of the variable definition.

2318 5.24 Element <VariableReference>

2319 The <VariableReference> element is used to reference a value defined within the same
2320 encompassing <Policy> element. The <VariableReference> element SHALL refer to the

2321 <VariableDefinition> element by **identifier equality** on the value of their respective VariableId
2322 attributes. One and only one <VariableDefinition> MUST exist within the same encompassing
2323 <Policy> element to which the <VariableReference> refers. There MAY be zero or more
2324 <VariableReference> elements that refer to the same <VariableDefinition> element.

```
2325 <xs:element name="VariableReference" type="xacml:VariableReferenceType"  
2326 substitutionGroup="xacml:Expression"/>  
2327 <xs:complexType name="VariableReferenceType">  
2328 <xs:complexContent>  
2329 <xs:extension base="xacml:ExpressionType">  
2330 <xs:attribute name="VariableId" type="xs:string"  
2331 use="required"/>  
2332 </xs:extension>  
2333 </xs:complexContent>  
2334 </xs:complexType>
```

2335 The <VariableReference> element is of the VariableReferenceType complex type, which is of
2336 the ExpressionType complex type and is a member of the <Expression> element substitution group.
2337 The <VariableReference> element MAY appear any place where an <Expression> element occurs
2338 in the schema.

2339 The <VariableReference> element has the following attribute:

2340 VariableId [Required]

2341 The name used to refer to the value defined in a <VariableDefinition> element.

2342 5.25 Element <Expression>

2343 The <Expression> element is not used directly in a **policy**. The <Expression> element signifies that
2344 an element that extends the ExpressionType and is a member of the <Expression> element
2345 substitution group SHALL appear in its place.

```
2346 <xs:element name="Expression" type="xacml:ExpressionType" abstract="true"/>  
2347 <xs:complexType name="ExpressionType" abstract="true"/>
```

2348 The following elements are in the <Expression> element substitution group:

2349 <Apply>, <AttributeSelector>, <AttributeValue>, <Function>, <VariableReference> and
2350 <AttributeDesignator>.

2351 5.26 Element <Condition>

2352 The <Condition> element is a Boolean function over **attributes** or functions of **attributes**.

```
2353 <xs:element name="Condition" type="xacml:ConditionType"/>  
2354 <xs:complexType name="ConditionType">  
2355 <xs:sequence>  
2356 <xs:element ref="xacml:Expression"/>  
2357 </xs:sequence>  
2358 </xs:complexType>
```

2359 The <Condition> contains one <Expression> element, with the restriction that the <Expression>
2360 return data-type MUST be "http://www.w3.org/2001/XMLSchema#boolean". Evaluation of the
2361 <Condition> element is described in Section 7.9.

2362 5.27 Element <Apply>

2363 The <Apply> element denotes application of a function to its arguments, thus encoding a function call.

2364 The <Apply> element can be applied to any combination of the members of the <Expression>
2365 element substitution group. See Section 5.25.


```

2366 <xs:element name="Apply" type="xacml:ApplyType"
2367 substitutionGroup="xacml:Expression"/>
2368 <xs:complexType name="ApplyType">
2369   <xs:complexContent>
2370     <xs:extension base="xacml:ExpressionType">
2371       <xs:sequence>
2372         <xs:element ref="xacml:Description" minOccurs="0"/>
2373         <xs:element ref="xacml:Expression" minOccurs="0"
2374           maxOccurs="unbounded"/>
2375       </xs:sequence>
2376       <xs:attribute name="FunctionId" type="xs:anyURI"
2377         use="required"/>
2378     </xs:extension>
2379   </xs:complexContent>
2380 </xs:complexType>

```

2381 The <Apply> element is of ApplyType complex type.

2382 The <Apply> element contains the following attributes and elements:

2383 FunctionId [Required]

2384 The identifier of the function to be applied to the arguments. XACML-defined functions are
2385 described in Appendix A.3.

2386 <Description> [Optional]

2387 A free-form description of the <Apply> element.

2388 <Expression> [Optional]

2389 Arguments to the function, which may include other functions.

2390 5.28 Element <Function>

2391 The <Function> element SHALL be used to name a function as an argument to the function defined by
2392 the parent <Apply> element.

```

2393 <xs:element name="Function" type="xacml:FunctionType"
2394 substitutionGroup="xacml:Expression"/>
2395 <xs:complexType name="FunctionType">
2396   <xs:complexContent>
2397     <xs:extension base="xacml:ExpressionType">
2398       <xs:attribute name="FunctionId" type="xs:anyURI"
2399         use="required"/>
2400     </xs:extension>
2401   </xs:complexContent>
2402 </xs:complexType>

```

2403 The <Function> element is of FunctionType complex type.

2404 The <Function> element contains the following attribute:

2405 FunctionId [Required]

2406 The identifier of the function.

2407 5.29 Element <AttributeDesignator>

2408 The <AttributeDesignator> element retrieves a **bag** of values for a **named attribute** from the
2409 request **context**. A **named attribute** SHALL be considered present if there is at least one **attribute** that
2410 matches the criteria set out below.

2411 The <AttributeDesignator> element SHALL return a **bag** containing all the **attribute** values that are
2412 matched by the **named attribute**. In the event that no matching **attribute** is present in the **context**, the

2413 MustBePresent attribute governs whether this element returns an empty **bag** or “Indeterminate”. See
2414 Section 7.3.5.

2415 The <AttributeDesignator> MAY appear in the <Match> element and MAY be passed to the
2416 <Apply> element as an argument.

2417 The <AttributeDesignator> element is of the AttributeDesignatorType complex type.

```
2418 <xs:element name="AttributeDesignator" type="xacml:AttributeDesignatorType"  
2419 substitutionGroup="xacml:Expression"/>  
2420 <xs:complexType name="AttributeDesignatorType">  
2421 <xs:complexContent>  
2422 <xs:extension base="xacml:ExpressionType">  
2423 <xs:attribute name="Category" type="xs:anyURI"  
2424 use="required"/>  
2425 <xs:attribute name="AttributeId" type="xs:anyURI"  
2426 use="required"/>  
2427 <xs:attribute name="DataType" type="xs:anyURI"  
2428 use="required"/>  
2429 <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2430 <xs:attribute name="MustBePresent" type="xs:boolean"  
2431 use="required"/>  
2432 </xs:extension>  
2433 </xs:complexContent>  
2434 </xs:complexType>
```

2435 A **named attribute** SHALL match an **attribute** if the values of their respective Category,
2436 AttributeId, DataType and Issuer attributes match. The attribute designator’s Category MUST
2437 match, by **identifier equality**, the Category of the <Attributes> element in which the **attribute** is
2438 present. The attribute designator’s AttributeId MUST match, by **identifier equality**, the
2439 AttributeId of the attribute. The attribute designator’s DataType MUST match, by **identifier**
2440 **equality**, the DataType of the same **attribute**.

2441 If the Issuer attribute is present in the attribute designator, then it MUST match, using the
2442 “urn:oasis:names:tc:xacml:1.0:function:string-equal” function, the Issuer of the same **attribute**. If the
2443 Issuer is not present in the attribute designator, then the matching of the **attribute** to the **named**
2444 **attribute** SHALL be governed by AttributeId and DataType attributes alone.

2445 The <AttributeDesignatorType> contains the following attributes:

2446 Category [Required]

2447 This attribute SHALL specify the Category with which to match the **attribute**.

2448 AttributeId [Required]

2449 This attribute SHALL specify the AttributeId with which to match the **attribute**.

2450 DataType [Required]

2451 The **bag** returned by the <AttributeDesignator> element SHALL contain values of this data-
2452 type.

2453 Issuer [Optional]

2454 This attribute, if supplied, SHALL specify the Issuer with which to match the **attribute**.

2455 MustBePresent [Required]

2456 This attribute governs whether the element returns “Indeterminate” or an empty **bag** in the event
2457 the **named attribute** is absent from the request **context**. See Section 7.3.5. Also see Sections
2458 7.19.2 and 7.19.3.

2459 5.30 Element <AttributeSelector>

2460 The <AttributeSelector> element produces a **bag** of unnamed and uncategorized **attribute** values.
2461 The values shall be constructed from the node(s) selected by applying the XPath expression given by the
2462 element's Path attribute to the XML content indicated by the element's Category attribute. Support for
2463 the <AttributeSelector> element is OPTIONAL.

2464 See section 7.3.7 for details of <AttributeSelector> evaluation.

```
2465 <xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"  
2466 substitutionGroup="xacml:Expression"/>  
2467 <xs:complexType name="AttributeSelectorType">  
2468 <xs:complexContent>  
2469 <xs:extension base="xacml:ExpressionType">  
2470 <xs:attribute name="Category" type="xs:anyURI"  
2471 use="required"/>  
2472 <xs:attribute name="ContextSelectorId" type="xs:anyURI"  
2473 use="optional"/>  
2474 <xs:attribute name="Path" type="xs:string"  
2475 use="required"/>  
2476 <xs:attribute name="DataType" type="xs:anyURI"  
2477 use="required"/>  
2478 <xs:attribute name="MustBePresent" type="xs:boolean"  
2479 use="required"/>  
2480 </xs:extension>  
2481 </xs:complexContent>  
2482 </xs:complexType>
```

2483 The <AttributeSelector> element is of AttributeSelectorType complex type.

2484 The <AttributeSelector> element has the following attributes:

2485 Category [Required]

2486 This attribute SHALL specify the **attributes** category of the <Content> element containing the
2487 XML from which nodes will be selected. It also indicates the **attributes** category containing the
2488 applicable ContextSelectorId attribute, if the element includes a ContextSelectorId xml
2489 attribute.

2490 ContextSelectorId [Optional]

2491 This attribute refers to the **attribute** (by its AttributeId) in the request **context** in the category
2492 given by the Category attribute. The referenced **attribute** MUST have data type
2493 urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression, and must select a single node in the
2494 <Content> element. The XPathCategory attribute of the referenced **attribute** MUST be equal
2495 to the Category attribute of the **attribute selector**.

2496 Path [Required]

2497 This attribute SHALL contain an XPath expression to be evaluated against the specified XML
2498 content. See Section 7.3.7 for details of the XPath evaluation during <AttributeSelector>
2499 processing. The namespace context for the value of the Path attribute is given by the [in-scope
2500 namespaces] [INFOSET] of the <AttributeSelector> element.

2501 DataType [Required]

2502 The attribute specifies the datatype of the values returned from the evaluation of this
2503 <AttributeSelector> element.

2504 MustBePresent [Required]

2505 This attribute governs whether the element returns "Indeterminate" or an empty **bag** in the event
2506 the XPath expression selects no node, that the attributes category specified by the Category
2507 attribute does not exist in the request context, or the attributes category does exist but it does not
2508 have a <Content> child element, or the <Content> element does exist but the XPath
2509 expression selects no node. See Section 7.3.5. Also see Sections 7.19.2 and 7.19.3.

2510 5.31 Element <AttributeValue>

2511 The <AttributeValue> element SHALL contain a literal **attribute** value.

```
2512 <xs:element name="AttributeValue" type="xacml:AttributeValueType"  
2513 substitutionGroup="xacml:Expression"/>  
2514 <xs:complexType name="AttributeValueType" mixed="true">  
2515   <xs:complexContent mixed="true">  
2516     <xs:extension base="xacml:ExpressionType">  
2517       <xs:sequence>  
2518         <xs:any namespace="##any" processContents="lax"  
2519           minOccurs="0" maxOccurs="unbounded"/>  
2520       </xs:sequence>  
2521       <xs:attribute name="DataType" type="xs:anyURI"  
2522         use="required"/>  
2523       <xs:anyAttribute namespace="##any" processContents="lax"/>  
2524     </xs:extension>  
2525   </xs:complexContent>  
2526 </xs:complexType>
```

2527 The <AttributeValue> element is of AttributeValueType complex type.

2528 The <AttributeValue> element has the following attributes:

2529 DataType [Required]

2530 The data-type of the **attribute** value.

2531 5.32 Element <Obligations>

2532 The <Obligations> element SHALL contain a set of <Obligation> elements.

```
2533 <xs:element name="Obligations" type="xacml:ObligationsType"/>  
2534 <xs:complexType name="ObligationsType">  
2535   <xs:sequence>  
2536     <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>  
2537   </xs:sequence>  
2538 </xs:complexType>
```

2539 The <Obligations> element is of ObligationsType complexType.

2540 The <Obligations> element contains the following element:

2541 <Obligation> [One to Many]

2542 A sequence of **obligations**. See Section 5.34.

2543 5.33 Element <AssociatedAdvice>

2544 The <AssociatedAdvice> element SHALL contain a set of <Advice> elements.

```
2545 <xs:element name="AssociatedAdvice" type="xacml:AssociatedAdviceType"/>  
2546 <xs:complexType name="AssociatedAdviceType">  
2547   <xs:sequence>  
2548     <xs:element ref="xacml:Advice" maxOccurs="unbounded"/>  
2549   </xs:sequence>  
2550 </xs:complexType>
```

2551 The <AssociatedAdvice> element is of AssociatedAdviceType complexType.

2552 The <AssociatedAdvice> element contains the following element:

2553 <Advice> [One to Many]

2554 A sequence of **advice**. See Section 5.35.

2555 5.34 Element <Obligation>

2556 The <Obligation> element SHALL contain an identifier for the **obligation** and a set of **attributes** that
2557 form arguments of the action defined by the **obligation**.

```
2558 <xs:element name="Obligation" type="xacml:ObligationType"/>
2559 <xs:complexType name="ObligationType">
2560   <xs:sequence>
2561     <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
2562       maxOccurs="unbounded"/>
2563   </xs:sequence>
2564   <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
2565 </xs:complexType>
```

2566 The <Obligation> element is of ObligationType complexType. See Section 7.18 for a description
2567 of how the set of **obligations** to be returned by the **PDP** is determined.

2568 The <Obligation> element contains the following elements and attributes:

2569 ObligationId [Required]

2570 **Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

2571 <AttributeAssignment> [Optional]

2572 **Obligation** arguments assignment. The values of the **obligation** arguments SHALL be
2573 interpreted by the **PEP**.

2574 5.35 Element <Advice>

2575 The <Advice> element SHALL contain an identifier for the **advice** and a set of **attributes** that form
2576 arguments of the supplemental information defined by the **advice**.

```
2577 <xs:element name="Advice" type="xacml:AdviceType"/>
2578 <xs:complexType name="AdviceType">
2579   <xs:sequence>
2580     <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
2581       maxOccurs="unbounded"/>
2582   </xs:sequence>
2583   <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>
2584 </xs:complexType>
```

2585 The <Advice> element is of AdviceType complexType. See Section 7.18 for a description of how the
2586 set of **advice** to be returned by the **PDP** is determined.

2587 The <Advice> element contains the following elements and attributes:

2588 AdviceId [Required]

2589 **Advice** identifier. The value of the **advice** identifier MAY be interpreted by the **PEP**.

2590 <AttributeAssignment> [Optional]

2591 **Advice** arguments assignment. The values of the **advice** arguments MAY be interpreted by the
2592 **PEP**.

2593 5.36 Element <AttributeAssignment>

2594 The <AttributeAssignment> element is used for including arguments in **obligation** and **advice**
2595 expressions. It SHALL contain an AttributeId and the corresponding **attribute** value, by extending
2596 the AttributeValueType type definition. The <AttributeAssignment> element MAY be used in
2597 any way that is consistent with the schema syntax, which is a sequence of <xs:any> elements. The
2598 value specified SHALL be understood by the **PEP**, but it is not further specified by XACML. See Section
2599 7.18. Section 4.2.4.3 provides a number of examples of arguments included in **obligation** -expressions.

```
2600 <xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
```

```

2601 <xs:complexType name="AttributeAssignmentType" mixed="true">
2602   <xs:complexContent>
2603     <xs:extension base="xacml:AttributeValueType">
2604       <xs:attribute name="AttributeId" type="xs:anyURI"
2605         use="required"/>
2606       <xs:attribute name="Category" type="xs:anyURI"
2607         use="optional"/>
2608       <xs:attribute name="Issuer" type="xs:string" use="optional"/>
2609     </xs:extension>
2610   </xs:complexContent>
2611 </xs:complexType>

```

2612 The <AttributeAssignment> element is of AttributeAssignmentType complex type.

2613 The <AttributeAssignment> element contains the following attributes:

2614 AttributeId [Required]

2615 The **attribute** Identifier.

2616 Category [Optional]

2617 An optional category of the **attribute**. If this attribute is missing, the **attribute** has no category.

2618 The **PEP** SHALL interpret the significance and meaning of any Category attribute. Non-

2619 normative note: an expected use of the category is to disambiguate **attributes** which are relayed

2620 from the request.

2621 Issuer [Optional]

2622 An optional issuer of the **attribute**. If this attribute is missing, the **attribute** has no issuer. The

2623 **PEP** SHALL interpret the significance and meaning of any Issuer attribute. Non-normative note:

2624 an expected use of the issuer is to disambiguate **attributes** which are relayed from the request.

2625 5.37 Element <ObligationExpressions>

2626 The <ObligationExpressions> element SHALL contain a set of <ObligationExpression>

2627 elements.

```

2628 <xs:element name="ObligationExpressions"
2629   type="xacml:ObligationExpressionsType"/>
2630 <xs:complexType name="ObligationExpressionsType">
2631   <xs:sequence>
2632     <xs:element ref="xacml:ObligationExpression" maxOccurs="unbounded"/>
2633   </xs:sequence>
2634 </xs:complexType>

```

2635 The <ObligationExpressions> element is of ObligationExpressionsType complexType.

2636 The <ObligationExpressions> element contains the following element:

2637 <ObligationExpression> [One to Many]

2638 A sequence of **obligations** expressions. See Section 5.39.

2639 5.38 Element <AdviceExpressions>

2640 The <AdviceExpressions> element SHALL contain a set of <AdviceExpression> elements.

```

2641 <xs:element name="AdviceExpressions" type="xacml:AdviceExpressionsType"/>
2642 <xs:complexType name="AdviceExpressionsType">
2643   <xs:sequence>
2644     <xs:element ref="xacml:AdviceExpression" maxOccurs="unbounded"/>
2645   </xs:sequence>
2646 </xs:complexType>

```

2647 The <AdviceExpressions> element is of AdviceExpressionsType complexType.

2648 The <AdviceExpressions> element contains the following element:

2649 <AdviceExpression> [One to Many]

2650 A sequence of **advice** expressions. See Section 5.40.

2651 5.39 Element <ObligationExpression>

2652 The <ObligationExpression> element evaluates to an **obligation** and SHALL contain an identifier
2653 for an **obligation** and a set of expressions that form arguments of the action defined by the **obligation**.
2654 The FulfillOn attribute SHALL indicate the **effect** for which this **obligation** must be fulfilled by the
2655 **PEP**.

```
2656 <xs:element name="ObligationExpression"  
2657   type="xacml:ObligationExpressionType"/>  
2658 <xs:complexType name="ObligationExpressionType">  
2659   <xs:sequence>  
2660     <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"  
2661       maxOccurs="unbounded"/>  
2662   </xs:sequence>  
2663   <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>  
2664   <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>  
2665 </xs:complexType>
```

2666 The <ObligationExpression> element is of ObligationExpressionType complexType. See
2667 Section 7.18 for a description of how the set of **obligations** to be returned by the **PDP** is determined.

2668 The <ObligationExpression> element contains the following elements and attributes:

2669 ObligationId [Required]

2670 **Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

2671 FulfillOn [Required]

2672 The **effect** for which this **obligation** must be fulfilled by the **PEP**.

2673 <AttributeAssignmentExpression> [Optional]

2674 **Obligation** arguments in the form of expressions. The expressions SHALL be evaluated by the
2675 PDP to constant <AttributeValue> elements or **bags**, which shall be the attribute
2676 assignments in the <Obligation> returned to the PEP. If an
2677 <AttributeAssignmentExpression> evaluates to an atomic **attribute** value, then there
2678 MUST be one resulting <AttributeAssignment> which MUST contain this single **attribute**
2679 value. If the <AttributeAssignmentExpression> evaluates to a **bag**, then there MUST be a
2680 resulting <AttributeAssignment> for each of the values in the **bag**. If the **bag** is empty, there
2681 shall be no <AttributeAssignment> from this <AttributeAssignmentExpression>. The
2682 values of the **obligation** arguments SHALL be interpreted by the **PEP**.

2683 5.40 Element <AdviceExpression>

2684 The <AdviceExpression> element evaluates to an **advice** and SHALL contain an identifier for an
2685 **advice** and a set of expressions that form arguments of the supplemental information defined by the
2686 **advice**. The AppliesTo attribute SHALL indicate the **effect** for which this **advice** must be provided to
2687 the **PEP**.

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

2696 `</xs:complexType>`

2697 The `<AdviceExpression>` element is of `AdviceExpressionType` complexType. See Section 7.18
 2698 for a description of how the set of **advice** to be returned by the **PDP** is determined.

2699 The `<AdviceExpression>` element contains the following elements and attributes:

2700 `AdviceId` [Required]
 2701 **Advice** identifier. The value of the **advice** identifier MAY be interpreted by the **PEP**.

2702 `AppliesTo` [Required]
 2703 The **effect** for which this **advice** must be provided to the **PEP**.

2704 `<AttributeAssignmentExpression>` [Optional]
 2705 **Advice** arguments in the form of expressions. The expressions SHALL be evaluated by the PDP
 2706 to constant `<AttributeValue>` elements or **bags**, which shall be the attribute assignments in
 2707 the `<Advice>` returned to the PEP. If an `<AttributeAssignmentExpression>` evaluates to
 2708 an atomic **attribute** value, then there MUST be one resulting `<AttributeAssignment>` which
 2709 MUST contain this single **attribute** value. If the `<AttributeAssignmentExpression>`
 2710 evaluates to a **bag**, then there MUST be a resulting `<AttributeAssignment>` for each of the
 2711 values in the **bag**. If the **bag** is empty, there shall be no `<AttributeAssignment>` from this
 2712 `<AttributeAssignmentExpression>`. The values of the **advice** arguments MAY be
 2713 interpreted by the **PEP**.

2714 5.41 Element `<AttributeAssignmentExpression>`

2715 The `<AttributeAssignmentExpression>` element is used for including arguments in **obligations**
 2716 and **advice**. It SHALL contain an `AttributeId` and an expression which SHALL be evaluated into the
 2717 corresponding **attribute** value. The value specified SHALL be understood by the **PEP**, but it is not further
 2718 specified by XACML. See Section 7.18. Section 4.2.4.3 provides a number of examples of arguments
 2719 included in **obligations**.

```

2720 <xs:element name="AttributeAssignmentExpression"
2721   type="xacml:AttributeAssignmentExpressionType"/>
2722 <xs:complexType name="AttributeAssignmentExpressionType">
2723   <xs:sequence>
2724     <xs:element ref="xacml:Expression"/>
2725   </xs:sequence>
2726   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
2727   <xs:attribute name="Category" type="xs:anyURI" use="optional"/>
2728   <xs:attribute name="Issuer" type="xs:string" use="optional"/>
2729 </xs:complexType>

```

2730 The `<AttributeAssignmentExpression>` element is of `AttributeAssignmentExpressionType`
 2731 complex type.

2732 The `<AttributeAssignmentExpression>` element contains the following attributes:

2733 `<Expression>` [Required]
 2734 The expression which evaluates to a constant **attribute** value or a bag of zero or more attribute
 2735 values. See section 5.25.

2736 `AttributeId` [Required]
 2737 The **attribute** identifier. The value of the `AttributeId` attribute in the resulting
 2738 `<AttributeAssignment>` element MUST be equal to this value.

2739 `Category` [Optional]
 2740 An optional category of the **attribute**. If this attribute is missing, the **attribute** has no category.
 2741 The value of the `Category` attribute in the resulting `<AttributeAssignment>` element MUST be
 2742 equal to this value.

2743 Issuer [Optional]
2744 An optional issuer of the **attribute**. If this attribute is missing, the **attribute** has no issuer. The
2745 value of the Issuer attribute in the resulting <AttributeAssignment> element MUST be equal to
2746 this value.

2747 5.42 Element <Request>

2748 The <Request> element is an abstraction layer used by the **policy** language. For simplicity of
2749 expression, this document describes **policy** evaluation in terms of operations on the **context**. However a
2750 conforming **PDP** is not required to actually instantiate the **context** in the form of an XML document. But,
2751 any system conforming to the XACML specification MUST produce exactly the same **authorization**
2752 **decisions** as if all the inputs had been transformed into the form of an <Request> element.

~~2753 The <Request> element contains <Attributes> elements. There may be multiple
2754 <Attributes> elements with the same Category attribute if the **PDP** implements the multiple
2755 decision profile, see [Multi]. Under other conditions, it is a syntax error if there are multiple
2756 <Attributes> elements with the same Category (see Section 7.19.2 for error codes).~~

```
2757 <xs:element name="Request" type="xacml:RequestType"/>  
2758 <xs:complexType name="RequestType">  
2759   <xs:sequence>  
2760     <xs:element ref="xacml:RequestDefaults" minOccurs="0"/>  
2761     <xs:element ref="xacml:Attributes" maxOccurs="unbounded"/>  
2762     <xs:element ref="xacml:MultiRequests" minOccurs="0"/>  
2763   </xs:sequence>  
2764   <xs:attribute name="ReturnPolicyIdList" type="xs:boolean" use="required"/>  
2765   <xs:attribute name="CombinedDecision" type="xs:boolean" use="required" />  
2766 </xs:complexType>
```

2767 The <Request> element is of RequestType complex type.

2768 The <Request> element contains the following elements and attributes:

2769 ReturnPolicyIdList [Required]

2770 This attribute is used to request that the **PDP** return a list of all fully applicable **policies** and
2771 **policy sets** which were used in the decision as a part of the decision response.

2772 CombinedDecision [Required]

2773 This attribute is used to request that the **PDP** combines multiple decisions into a single decision.
2774 The use of this attribute is specified in [Multi]. If the **PDP** does not implement the relevant
2775 functionality in [Multi], then the **PDP** must return an Indeterminate with a status code of
2776 urn:oasis:names:tc:xacml:1.0:status:processing-error if it receives a request with this attribute set
2777 to "true".

2778 <RequestDefaults> [Optional]

2779 Contains default values for the request, such as XPath version. See section 5.43.

2780 <Attributes> [One to Many]

2781 Specifies information about **attributes** of the request **context** by listing a sequence of
2782 <Attribute> elements associated with an **attribute** category. One or more <Attributes>
2783 elements are allowed. Different <Attributes> elements with different categories are used to
2784 represent information about the **subject**, **resource**, **action**, **environment** or other categories of
2785 the **access** request.

~~2786 The <Request> element contains <Attributes> elements. There may be multiple
2787 <Attributes> elements with the same Category attribute if the **PDP** implements the multiple
2788 decision profile, see [Multi]. Under other conditions, it is a syntax error if there are multiple
2789 <Attributes> elements with the same Category (see Section 7.19.2 for error codes).~~

2790 <MultiRequests> [Optional]

2791 Lists multiple **request contexts** by references to the <Attributes> elements. Implementation
2792 of this element is optional. The semantics of this element is defined in [Multi]. If the
2793 implementation does not implement this element, it MUST return an Indeterminate result if it
2794 encounters this element. See section 5.50.

2795 5.43 Element <RequestDefaults>

2796 The <RequestDefaults> element SHALL specify default values that apply to the <Request> element.

```
2797 <xs:element name="RequestDefaults" type="xacml:RequestDefaultsType"/>  
2798 <xs:complexType name="RequestDefaultsType">  
2799   <xs:sequence>  
2800     <xs:choice>  
2801       <xs:element ref="xacml:XPathVersion"/>  
2802     </xs:choice>  
2803   </xs:sequence>  
2804 </xs:complexType>
```

2805 <RequestDefaults> element is of RequestDefaultsType complex type.

2806 The <RequestDefaults> element contains the following elements:

2807 <XPathVersion> [Optional]

2808 Default XPath version for XPath expressions occurring in the request.

2809 5.44 Element <Attributes>

2810 The <Attributes> element specifies **attributes** of a **subject**, **resource**, **action**, **environment** or
2811 another category by listing a sequence of <Attribute> elements associated with the category.

```
2812 <xs:element name="Attributes" type="xacml:AttributesType"/>  
2813 <xs:complexType name="AttributesType">  
2814   <xs:sequence>  
2815     <xs:element ref="xacml:Content" minOccurs="0"/>  
2816     <xs:element ref="xacml:Attribute" minOccurs="0"  
2817       maxOccurs="unbounded"/>  
2818   </xs:sequence>  
2819   <xs:attribute name="Category" type="xs:anyURI" use="required"/>  
2820   <xs:attribute ref="xml:id" use="optional"/>  
2821 </xs:complexType><del><xs:complexType name="SubjectType"></del>
```

2822 The <Attributes> element is of AttributesType complex type.

2823 The <Attributes> element contains the following elements and attributes:

2824 Category [Required]

2825 This attribute indicates which **attribute** category the contained **attributes** belong to. The
2826 Category attribute is used to differentiate between **attributes** of **subject**, **resource**, **action**,
2827 **environment** or other categories.

2828 xml:id [Optional]

2829 This attribute provides a unique identifier for this <Attributes> element. See [XMLid] It is
2830 primarily intended to be referenced in multiple requests. See [Multi].

2831 <Content> [Optional]

2832 Specifies additional sources of **attributes** in free form XML document format which can be
2833 referenced using <AttributeSelector> elements.

2834 <Attribute> [Any Number]

2835 A sequence of **attributes** that apply to the category of the request.

2836 5.45 Element <Content>

2837 The <Content> element is a notional placeholder for additional **attributes**, typically the content of the
2838 **resource**.

```
2839 <xs:element name="Content" type="xacml:ContentType"/>  
2840 <xs:complexType name="ContentType" mixed="true">  
2841   <xs:sequence>  
2842     <xs:any namespace="##any" processContents="lax"/>  
2843   </xs:sequence>  
2844 </xs:complexType>
```

2845 The <Content> element is of ContentType complex type.

2846 The <Content> element has exactly one arbitrary type child element.

2847 5.46 Element <Attribute>

2848 The <Attribute> element is the central abstraction of the request **context**. It contains **attribute** meta-
2849 data and one or more **attribute** values. The **attribute** meta-data comprises the **attribute** identifier and
2850 the **attribute** issuer. <AttributeDesignator> elements in the **policy** MAY refer to **attributes** by
2851 means of this meta-data.

```
2852 <xs:element name="Attribute" type="xacml:AttributeType"/>  
2853 <xs:complexType name="AttributeType">  
2854   <xs:sequence>  
2855     <xs:element ref="xacml:AttributeValue" maxOccurs="unbounded"/>  
2856   </xs:sequence>  
2857   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>  
2858   <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2859   <xs:attribute name="IncludeInResult" type="xs:boolean" use="required"/>  
2860 </xs:complexType>
```

2861 The <Attribute> element is of AttributeType complex type.

2862 The <Attribute> element contains the following attributes and elements:

2863 AttributeId [Required]

2864 The **Attribute** identifier. A number of identifiers are reserved by XACML to denote commonly
2865 used **attributes**. See Appendix Appendix B.

2866 Issuer [Optional]

2867 The **Attribute** issuer. For example, this attribute value MAY be an x500Name that binds to a
2868 public key, or it may be some other identifier exchanged out-of-band by issuing and relying
2869 parties.

2870 IncludeInResult [Default: false]

2871 Whether to include this **attribute** in the result. This is useful to correlate requests with their
2872 responses in case of multiple requests.

2873 <AttributeValue> [One to Many]

2874 One or more **attribute** values. Each **attribute** value MAY have contents that are empty, occur
2875 once or occur multiple times.

2876 5.47 Element <Response>

2877 The <Response> element is an abstraction layer used by the **policy** language. Any proprietary system
2878 using the XACML specification MUST transform an XACML **context** <Response> element into the form
2879 of its **authorization decision**.

2880 The <Response> element encapsulates the **authorization decision** produced by the **PDP**. It includes a
2881 sequence of one or more results, with one <Result> element per requested **resource**. Multiple results

2882 MAY be returned by some implementations, in particular those that support the XACML Profile for
2883 Requests for Multiple Resources [**Multi**]. Support for multiple results is **OPTIONAL**.

```
2884 <xs:element name="Response" type="xacml:ResponseType"/>  
2885 <xs:complexType name="ResponseType">  
2886   <xs:sequence>  
2887     <xs:element ref="xacml:Result" maxOccurs="unbounded"/>  
2888   </xs:sequence>  
2889 </xs:complexType>
```

2890 The <Response> element is of ResponseType complex type.

2891 The <Response> element contains the following elements:

2892 <Result> [One to Many]

2893 An **authorization decision** result. See Section 5.48.

2894 5.48 Element <Result>

2895 The <Result> element represents an **authorization decision** result. It MAY include a set of
2896 **obligations** that MUST be fulfilled by the **PEP**. If the **PEP** does not understand or cannot fulfill an
2897 **obligation**, then the action of the PEP is determined by its bias, see section 7.1. It MAY include a set of
2898 **advice** with supplemental information which MAY be safely ignored by the **PEP**.

```
2899 <xs:complexType name="ResultType">  
2900   <xs:sequence>  
2901     <xs:element ref="xacml:Decision"/>  
2902     <xs:element ref="xacml:Status" minOccurs="0"/>  
2903     <xs:element ref="xacml:Obligations" minOccurs="0"/>  
2904     <xs:element ref="xacml:AssociatedAdvice" minOccurs="0"/>  
2905     <xs:element ref="xacml:Attributes" minOccurs="0"  
2906       maxOccurs="unbounded"/>  
2907     <xs:element ref="xacml:PolicyIdentifierList" minOccurs="0"/>  
2908   </xs:sequence>  
2909 </xs:complexType>
```

2910 The <Result> element is of ResultType complex type.

2911 The <Result> element contains the following attributes and elements:

2912 <Decision> [Required]

2913 The **authorization decision**: "Permit", "Deny", "Indeterminate" or "NotApplicable".

2914 <Status> [Optional]

2915 Indicates whether errors occurred during evaluation of the **decision request**, and optionally,
2916 information about those errors. If the <Response> element contains <Result> elements whose
2917 <Status> elements are all identical, and the <Response> element is contained in a protocol
2918 wrapper that can convey status information, then the common status information MAY be placed
2919 in the protocol wrapper and this <Status> element MAY be omitted from all <Result>
2920 elements.

2921 <Obligations> [Optional]

2922 A list of **obligations** that MUST be fulfilled by the **PEP**. If the **PEP** does not understand or cannot
2923 fulfill an **obligation**, then the action of the PEP is determined by its bias, see section 7.2. See
2924 Section 7.18 for a description of how the set of **obligations** to be returned by the **PDP** is
2925 determined.

2926 <AssociatedAdvice> [Optional]

2927 A list of **advice** that provide supplemental information to the **PEP**. If the **PEP** does not
2928 understand an **advice**, the PEP may safely ignore the **advice**. See Section 7.18 for a description
2929 of how the set of **advice** to be returned by the **PDP** is determined.

2930 <Attributes> [Optional]

2931 A list of **attributes** that were part of the request. The choice of which **attributes** are included here
2932 is made with the `IncludeInResult` attribute of the <Attribute> elements of the request. See
2933 section 5.46.

2934 <PolicyIdentifierList> [Optional]

2935 If the `ReturnPolicyIdList` attribute in the <Request> is true (see section 5.42), a **PDP** that
2936 implements this optional feature **MUST** return a list which includes the identifiers of all **policies**
2937 which were found to be fully applicable. ~~That is, all policies where both the <Target> matched~~
2938 ~~and the <Condition> evaluated to true~~, whether or not the <Effect> (after rule combining)
2939 was the same or different from the <Decision>. The list MAY include the identifiers of other
2940 policies which are currently in force, as long as no policies required for the decision are omitted. A
2941 PDP MAY satisfy this requirement by including all policies currently in force, or by including all
2942 policies which were evaluated in making the decision, or by including all policies which did not
2943 evaluate to "NotApplicable", or by any other algorithm which does not omit any policies which
2944 contributed to the decision. However, a decision which returns "NotApplicable" MUST return an
2945 empty list.

2946 5.49 Element <PolicyIdentifierList>

2947 The <PolicyIdentifierList> element contains a list of **policy** and **policy set** identifiers of **policies**
2948 which have been applicable to a request. The list is unordered.

```
2949 <xs:element name="PolicyIdentifierList"  
2950   type="xacml:PolicyIdentifierListType"/>  
2951 <xs:complexType name="PolicyIdentifierListType">  
2952   <xs:choice minOccurs="0" maxOccurs="unbounded">  
2953     <xs:element ref="xacml:PolicyIdReference"/>  
2954     <xs:element ref="xacml:PolicySetIdReference"/>  
2955   </xs:choice>  
2956 </xs:complexType>
```

2957 The <PolicyIdentifierList> element is of `PolicyIdentifierListType` complex type.

2958 The <PolicyIdentifierList> element contains the following elements.

2959 <PolicyIdReference> [Any number]

2960 The identifier and version of a **policy** which was applicable to the request. See section 5.11. The
2961 <PolicyIdReference> element **MUST** use the `Version` attribute to specify the version and
2962 **MUST NOT** use the `LatestVersion` or `EarliestVersion` attributes.

2963 <PolicySetIdReference> [Any number]

2964 The identifier and version of a **policy set** which was applicable to the request. See section 5.10.
2965 The <PolicySetIdReference> element **MUST** use the `Version` attribute to specify the
2966 version and **MUST NOT** use the `LatestVersion` or `EarliestVersion` attributes.

2967 5.50 Element <MultiRequests>

2968 The <MultiRequests> element contains a list of requests by reference to <Attributes> elements in
2969 the enclosing <Request> element. The semantics of this element are defined in [Multi]. Support for this
2970 element is optional. If an implementation does not support this element, but receives it, the
2971 implementation **MUST** generate an "Indeterminate" response.

```
2972 <xs:element name="MultiRequests" type="xacml:MultiRequestsType"/>  
2973 <xs:complexType name="MultiRequestsType">  
2974   <xs:sequence>  
2975     <xs:element ref="xacml:RequestReference" maxOccurs="unbounded"/>  
2976   </xs:sequence>  
2977 </xs:complexType>
```

2978 The <MultiRequests> element contains the following elements.

2979 <RequestReference> [one to many]

2980 Defines a request instance by reference to <Attributes> elements in the enclosing
2981 <Request> element. See section 5.51.

2982 5.51 Element <RequestReference>

2983 The <RequestReference> element defines an instance of a request in terms of references to
2984 <Attributes> elements. The semantics of this element are defined in [Multi]. Support for this element
2985 is optional.

```
2986 <xs:element name="RequestReference" type="xacml:RequestReference" />  
2987 <xs:complexType name="RequestReferenceType">  
2988   <xs:sequence>  
2989     <xs:element ref="xacml:AttributesReference" maxOccurs="unbounded" />  
2990   </xs:sequence>  
2991 </xs:complexType>
```

2992 The <RequestReference> element contains the following elements.

2993 <AttributesReference> [one to many]

2994 A reference to an <Attributes> element in the enclosing <Request> element. See section
2995 5.52.

2996 5.52 Element <AttributesReference>

2997 The <AttributesReference> element makes a reference to an <Attributes> element. The
2998 meaning of this element is defined in [Multi]. Support for this element is optional.

```
2999 <xs:element name="AttributesReference" type="xacml:AttributesReference" />  
3000 <xs:complexType name="AttributesReferenceType">  
3001   <xs:attribute name="ReferenceId" type="xs:IDREF" use="required" />  
3002 </xs:complexType>
```

3003 The <AttributesReference> element contains the following attributes.

3004 ReferenceId [required]

3005 A reference to the xml:id attribute of an <Attributes> element in the enclosing <Request>
3006 element.

3007 5.53 Element <Decision>

3008 The <Decision> element contains the result of *policy* evaluation.

```
3009 <xs:element name="Decision" type="xacml:DecisionType" />  
3010 <xs:simpleType name="DecisionType">  
3011   <xs:restriction base="xs:string">  
3012     <xs:enumeration value="Permit" />  
3013     <xs:enumeration value="Deny" />  
3014     <xs:enumeration value="Indeterminate" />  
3015     <xs:enumeration value="NotApplicable" />  
3016   </xs:restriction>  
3017 </xs:simpleType>
```

3018 The <Decision> element is of DecisionType simple type.

3019 The values of the <Decision> element have the following meanings:

3020 "Permit": the requested **access** is permitted.

3021 "Deny": the requested **access** is denied.

- 3022 "Indeterminate": the **PDP** is unable to evaluate the requested **access**. Reasons for such inability
 3023 include: missing **attributes**, network errors while retrieving **policies**, division by zero during
 3024 **policy** evaluation, syntax errors in the **decision request** or in the **policy**, etc.
- 3025 "NotApplicable": the **PDP** does not have any **policy** that applies to this **decision request**.

3026 5.54 Element <Status>

3027 The <Status> element represents the status of the **authorization decision** result.

```
3028 <xs:element name="Status" type="xacml:StatusType"/>
3029 <xs:complexType name="StatusType">
3030   <xs:sequence>
3031     <xs:element ref="xacml:StatusCode"/>
3032     <xs:element ref="xacml:StatusMessage" minOccurs="0"/>
3033     <xs:element ref="xacml:StatusDetail" minOccurs="0"/>
3034   </xs:sequence>
3035 </xs:complexType>
```

3036 The <Status> element is of StatusType complex type.

3037 The <Status> element contains the following elements:

3038 <StatusCode> [Required]

3039 Status code.

3040 <StatusMessage> [Optional]

3041 A status message describing the status code.

3042 <StatusDetail> [Optional]

3043 Additional status information.

3044 5.55 Element <StatusCode>

3045 The <StatusCode> element contains a major status code value and an optional [sequence recursive](#)
 3046 [series](#) of minor status codes.

```
3047 <xs:element name="StatusCode" type="xacml:StatusCodeType"/>
3048 <xs:complexType name="StatusCodeType">
3049   <xs:sequence>
3050     <xs:element ref="xacml:StatusCode" minOccurs="0"/>
3051   </xs:sequence>
3052   <xs:attribute name="Value" type="xs:anyURI" use="required"/>
3053 </xs:complexType>
```

3054 The <StatusCode> element is of StatusCodeType complex type.

3055 The <StatusCode> element contains the following attributes and elements:

3056 Value [Required]

3057 See Section B.8 for a list of values.

3058 <StatusCode> [Any Number]

3059 Minor status code. This status code qualifies its parent status code.

3060 5.56 Element <StatusMessage>

3061 The <StatusMessage> element is a free-form description of the status code.

```
3062 <xs:element name="StatusMessage" type="xs:string"/>
```

3063 The <StatusMessage> element is of xs:string type.

3064 5.57 Element <StatusDetail>

3065 The <StatusDetail> element qualifies the <Status> element with additional information.

```
3066 <xs:element name="StatusDetail" type="xacml:StatusDetailType"/>
3067 <xs:complexType name="StatusDetailType">
3068   <xs:sequence>
3069     <xs:any namespace="##any" processContents="lax" minOccurs="0"
3070       maxOccurs="unbounded"/>
3071   </xs:sequence>
3072 </xs:complexType>
```

3073 The <StatusDetail> element is of StatusDetailType complex type.

3074 The <StatusDetail> element allows arbitrary XML content.

3075 Inclusion of a <StatusDetail> element is optional. However, if a **PDP** returns one of the following
3076 XACML-defined <StatusCode> values and includes a <StatusDetail> element, then the following
3077 rules apply.

3078 urn:oasis:names:tc:xacml:1.0:status:ok

3079 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “ok” status value.

3080 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

3081 A **PDP** MAY choose not to return any <StatusDetail> information or MAY choose to return a
3082 <StatusDetail> element containing one or more <MissingAttributeDetail> elements.

3083 urn:oasis:names:tc:xacml:1.0:status:syntax-error

3084 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “syntax-error” status
3085 value. A syntax error may represent either a problem with the **policy** being used or with the request
3086 **context**. The **PDP** MAY return a <StatusMessage> describing the problem.

3087 urn:oasis:names:tc:xacml:1.0:status:processing-error

3088 A **PDP** MUST NOT return <StatusDetail> element in conjunction with the “processing-error” status
3089 value. This status code indicates an internal problem in the **PDP**. For security reasons, the **PDP** MAY
3090 choose to return no further information to the **PEP**. In the case of a divide-by-zero error or other
3091 computational error, the **PDP** MAY return a <StatusMessage> describing the nature of the error.

3092 5.58 Element <MissingAttributeDetail>

3093 The <MissingAttributeDetail> element conveys information about **attributes** required for **policy**
3094 evaluation that were missing from the request **context**.

```
3095 <xs:element name="MissingAttributeDetail"
3096   type="xacml:MissingAttributeDetailType"/>
3097 <xs:complexType name="MissingAttributeDetailType">
3098   <xs:sequence>
3099     <xs:element ref="xacml:AttributeValue" minOccurs="0"
3100       maxOccurs="unbounded"/>
3101   </xs:sequence>
3102   <xs:attribute name="Category" type="xs:anyURI" use="required"/>
3103   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
3104   <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
3105   <xs:attribute name="Issuer" type="xs:string" use="optional"/>
3106 </xs:complexType>
```

3107 The <MissingAttributeDetail> element is of MissingAttributeDetailType complex type.

3108 The <MissingAttributeDetail> element contains the following attributes and elements:

3109 <AttributeValue> [Optional]

3110 The required value of the missing **attribute**.

- 3111 Category [Required]
- 3112 The category identifier of the missing **attribute**.
- 3113 AttributeId [Required]
- 3114 The identifier of the missing **attribute**.
- 3115 DataType [Required]
- 3116 The data-type of the missing **attribute**.
- 3117 Issuer [Optional]
- 3118 This attribute, if supplied, SHALL specify the required `Issuer` of the missing **attribute**.
- 3119 If the **PDP** includes `<AttributeValue>` elements in the `<MissingAttributeDetail>` element, then
- 3120 this indicates the acceptable values for that **attribute**. If no `<AttributeValue>` elements are included,
- 3121 then this indicates the names of **attributes** that the **PDP** failed to resolve during its evaluation. The list of
- 3122 **attributes** may be partial or complete. There is no guarantee by the **PDP** that supplying the missing
- 3123 values or **attributes** will be sufficient to satisfy the **policy**.

3124

6 XPath 2.0 definitions

3125 The XPath 2.0 specification leaves a number of aspects of behavior implementation defined. This section
3126 defines how XPath 2.0 SHALL behave when hosted in XACML.

3127 <http://www.w3.org/TR/2007/REC-xpath20-20070123/#id-impl-defined-items> defines the following items:

- 3128 1. The version of Unicode that is used to construct expressions.
3129 XACML leaves this implementation defined. It is RECOMMENDED that the latest version is used.
- 3130 2. The statically-known collations.
3131 XACML leaves this implementation defined.
- 3132 3. The implicit timezone.
3133 XACML defined the implicit time zone as UTC.
- 3134 4. The circumstances in which warnings are raised, and the ways in which warnings are handled.
3135 XACML leaves this implementation defined.
- 3136 5. The method by which errors are reported to the external processing environment.
3137 An XPath error causes an XACML Indeterminate value in the element where the XPath error
3138 occurs. The StatusCode value SHALL be "urn:oasis:names:tc:xacml:1.0:status:processing-error".
3139 Implementations MAY provide additional details about the error in the response or by some other
3140 means.
- 3141 6. Whether the implementation is based on the rules of XML 1.0 or 1.1.
3142 XACML is based on XML 1.0.
- 3143 7. Whether the implementation supports the namespace axis.
3144 XACML leaves this implementation defined. It is RECOMMENDED that users of XACML do not
3145 make use of the namespace axis.
- 3146 8. Any static typing extensions supported by the implementation, if the Static Typing Feature is
3147 supported.
3148 XACML leaves this implementation defined.

3149

3150 <http://www.w3.org/TR/2007/REC-xpath-datamodel-20070123/#implementation-defined> defines the
3151 following items:

- 3152 1. Support for additional user-defined or implementation-defined types is implementation-defined.
3153 It is RECOMMENDED that implementations of XACML do not define any additional types and it is
3154 RECOMMENDED that users of XACML do not make user of any additional types.
- 3155 2. Some typed values in the data model are undefined. Attempting to access an undefined property
3156 is always an error. Behavior in these cases is implementation-defined and the host language is
3157 responsible for determining the result.
3158 An XPath error causes an XACML Indeterminate value in the element where the XPath error
3159 occurs. The StatusCode value SHALL be "urn:oasis:names:tc:xacml:1.0:status:processing-error".
3160 Implementations MAY provide additional details about the error in the response or by some other
3161 means.

3162

3163 <http://www.w3.org/TR/2007/REC-xpath-functions-20070123/#impl-def> defines the following items:

- 3164 1. The destination of the trace output is implementation-defined.
3165 XACML leaves this implementation defined.
- 3166 2. For xs:integer operations, implementations that support limited-precision integer operations must
3167 either raise an error [err:FOAR0002] or provide an implementation-defined mechanism that
3168 allows users to choose between raising an error and returning a result that is modulo the largest
3169 representable integer value.
3170 XACML leaves this implementation defined. If an implementation chooses to raise an error, the

- 3171 StatusCode value SHALL be “urn:oasis:names:tc:xacml:1.0:status:processing-error”.
3172 Implementations MAY provide additional details about the error in the response or by some other
3173 means.
- 3174 3. For xs:decimal values the number of digits of precision returned by the numeric operators is
3175 implementation-defined.
3176 XACML leaves this implementation defined.
- 3177 4. If the number of digits in the result of a numeric operation exceeds the number of digits that the
3178 implementation supports, the result is truncated or rounded in an implementation-defined manner.
3179 XACML leaves this implementation defined.
- 3180 5. It is implementation-defined which version of Unicode is supported.
3181 XACML leaves this implementation defined. It is RECOMMENDED that the latest version is used.
- 3182 6. For fn:normalize-unicode, conforming implementations must support normalization form "NFC"
3183 and may support normalization forms "NFD", "NFKC", "NFKD", "FULLY-NORMALIZED". They
3184 may also support other normalization forms with implementation-defined semantics.
3185 XACML leaves this implementation defined.
- 3186 7. The ability to decompose strings into collation units suitable for substring matching is an
3187 implementation-defined property of a collation.
3188 XACML leaves this implementation defined.
- 3189 8. All minimally conforming processors must support year values with a minimum of 4 digits (i.e.,
3190 YYYY) and a minimum fractional second precision of 1 millisecond or three digits (i.e., s.sss).
3191 However, conforming processors may set larger implementation-defined limits on the maximum
3192 number of digits they support in these two situations.
3193 XACML leaves this implementation defined, and it is RECOMMENDED that users of XACML do
3194 not expect greater limits and precision.
- 3195 9. The result of casting a string to xs:decimal, when the resulting value is not too large or too small
3196 but nevertheless has too many decimal digits to be accurately represented, is implementation-
3197 defined.
3198 XACML leaves this implementation defined.
- 3199 10. Various aspects of the processing provided by fn:doc are implementation-defined.
3200 Implementations may provide external configuration options that allow any aspect of the
3201 processing to be controlled by the user.
3202 XACML leaves this implementation defined.
- 3203 11. The manner in which implementations provide options to weaken the stable characteristic of
3204 fn:collection and fn:doc are implementation-defined.
3205 XACML leaves this implementation defined.

3206 7 Functional requirements

3207 This section specifies certain functional requirements that are not directly associated with the production
3208 or consumption of a particular XACML element.

3209 Note that in each case an implementation is conformant as long as it produces the same result as is
3210 specified here, regardless of how and in what order the implementation behaves internally.

3211 7.1 Unicode issues

3212 7.1.1 Normalization

3213 In Unicode, some equivalent characters can be represented by more than one different Unicode
3214 character sequence. See [CMF]. The process of converting Unicode strings into equivalent character
3215 sequences is called "normalization" [UAX15]. Some operations, such as string comparison, are sensitive
3216 to normalization. An operation is normalization-sensitive if its output(s) are different depending on the
3217 state of normalization of the input(s); if the output(s) are textual, they are deemed different only if they
3218 would remain different were they to be normalized.

3219 For more information on normalization see [CM].

3220 An XACML implementation MUST behave as if each normalization-sensitive operation normalizes input
3221 strings into Unicode Normalization Form C ("NFC"). An implementation MAY use some other form of
3222 internal processing (such as using a non-Unicode, "legacy" character encoding) as long as the externally
3223 visible results are identical to this specification.

3224 7.1.2 Version of Unicode

3225 The version of Unicode used by XACML is implementation defined. It is RECOMMENDED that the latest
3226 version is used. Also note security issues in section 9.3.

3227 7.2 Policy enforcement point

3228 This section describes the requirements for the *PEP*.

3229 An application functions in the role of the *PEP* if it guards *access* to a set of *resources* and asks the
3230 *PDP* for an *authorization decision*. The *PEP* MUST abide by the *authorization decision* as described
3231 in one of the following sub-sections

3232 In any case any *advice* in the *decision* may be safely ignored by the *PEP*.

3233 7.2.1 Base PEP

3234 If the *decision* is "Permit", then the *PEP* SHALL permit *access*. If *obligations* accompany the *decision*,
3235 then the *PEP* SHALL permit *access* only if it understands and it can and will discharge those
3236 *obligations*.

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

3239 If the *decision* is "Not Applicable", then the *PEP*'s behavior is undefined.

3240 If the *decision* is "Indeterminate", then the *PEP*'s behavior is undefined.

3241 7.2.2 Deny-biased PEP

3242 If the *decision* is "Permit", then the *PEP* SHALL permit *access*. If *obligations* accompany the *decision*,
3243 then the *PEP* SHALL permit *access* only if it understands and it can and will discharge those
3244 *obligations*.

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

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

3248 7.2.3 Permit-biased PEP

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

3251 All other **decisions** SHALL result in the permission of **access**.

3252 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of
3253 the **decision request**, etc., are not prohibited.

3254 7.3 Attribute evaluation

3255 **Attributes** are represented in the request **context** by the **context handler**, regardless of whether or not
3256 they appeared in the original **decision request**, and are referred to in the **policy** by attribute designators
3257 and attribute selectors. A **named attribute** is the term used for the criteria that the specific attribute
3258 designators use to refer to particular **attributes** in the <Attributes> elements of the request **context**.

3259 7.3.1 Structured attributes

3260 <AttributeValue> elements MAY contain an instance of a structured XML data-type, for example
3261 <ds:KeyInfo>. XACML 3.0 supports several ways for comparing the contents of such elements.

3262 1. In some cases, such elements MAY be compared using one of the XACML string functions, such
3263 as "string-regexp-match", described below. This requires that the element be given the data-type
3264 "http://www.w3.org/2001/XMLSchema#string". For example, a structured data-type that is
3265 actually a ds:KeyInfo/KeyName would appear in the **Context** as:

```
3266 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3267 <ds:KeyName>jhibbert-key</ds:KeyName>  
3268 </AttributeValue>
```

3269 In general, this method will not be adequate unless the structured data-type is quite simple.

3270 2. The structured **attribute** MAY be made available in the <Content> element of the appropriate
3271 **attribute** category and an <AttributeSelector> element MAY be used to select the contents
3272 of a leaf sub-element of the structured data-type by means of an XPath expression. That value
3273 MAY then be compared using one of the supported XACML functions appropriate for its primitive
3274 data-type. This method requires support by the **PDP** for the optional XPath expressions feature.

3275 3. The structured **attribute** MAY be made available in the <Content> element of the appropriate
3276 **attribute** category and an <AttributeSelector> element MAY be used to select any node in
3277 the structured data-type by means of an XPath expression. This node MAY then be compared
3278 using one of the XPath-based functions described in Section A.3.15. This method requires
3279 support by the **PDP** for the optional XPath expressions and XPath functions features.

3280 See also Section 7.3.

3281 7.3.2 Attribute bags

3282 XACML defines implicit collections of its data-types. XACML refers to a collection of values that are of a
3283 single data-type as a **bag**. **Bags** of data-types are needed because selections of nodes from an XML
3284 **resource** or XACML request **context** may return more than one value.

3285 The <AttributeSelector> element uses an XPath expression to specify the selection of data from
3286 free form XML. The result of an XPath expression is termed a node-set, which contains all the nodes
3287 from the XML content that match the **predicate** in the XPath expression. Based on the various indexing
3288 functions provided in the XPath specification, it SHALL be implied that a resultant node-set is the

3289 collection of the matching nodes. XACML also defines the <AttributeDesignator> element to have
3290 the same matching methodology for **attributes** in the XACML request **context**.

3291 The values in a **bag** are not ordered, and some of the values may be duplicates. There SHALL be no
3292 notion of a **bag** containing **bags**, or a **bag** containing values of differing types; i.e., a **bag** in XACML
3293 SHALL contain only values that are of the same data-type.

3294 7.3.3 Multivalued attributes

3295 If a single <Attribute> element in a request **context** contains multiple <AttributeValue> child
3296 elements, then the **bag** of values resulting from evaluation of the <Attribute> element MUST be
3297 identical to the **bag** of values that results from evaluating a **context** in which each <AttributeValue>
3298 element appears in a separate <Attribute> element, each carrying identical meta-data.

3299 7.3.4 Attribute Matching

3300 A **named attribute** includes specific criteria with which to match **attributes** in the **context**. An **attribute**
3301 specifies a *Category*, *AttributeId* and *DataType*, and a **named attribute** also specifies the
3302 *Issuer*. A **named attribute** SHALL match an **attribute** if the values of their respective *Category*,
3303 *AttributeId*, *DataType* and optional *Issuer* attributes match. The *Category* of the **named**
3304 **attribute** MUST match, by **identifier equality**, the *Category* of the corresponding **context attribute**.
3305 The *AttributeId* of the **named attribute** MUST match, by **identifier equality**, the *AttributeId* of
3306 the corresponding **context attribute**. The *DataType* of the **named attribute** MUST match, by **identifier**
3307 **equality**, the *DataType* of the corresponding **context attribute**. If *Issuer* is supplied in the **named**
3308 **attribute**, then it MUST match, using the urn:oasis:names:tc:xacml:1.0:function:string-equal function, the
3309 *Issuer* of the corresponding **context attribute**. If *Issuer* is not supplied in the **named attribute**, then
3310 the matching of the **context attribute** to the **named attribute** SHALL be governed by *AttributeId* and
3311 *DataType* alone, regardless of the presence, absence, or actual value of *Issuer* in the corresponding
3312 **context attribute**. In the case of an attribute selector, the matching of the **attribute** to the **named**
3313 **attribute** SHALL be governed by the XPath expression and *DataType*.

3314 7.3.5 Attribute Retrieval

3315 The **PDP** SHALL request the values of **attributes** in the request **context** from the **context handler**. The
3316 **context handler** MAY also add **attributes** to the request **context** without the **PDP** requesting them. The
3317 **PDP** SHALL reference the **attributes** as if they were in a physical request **context** document, but the
3318 **context handler** is responsible for obtaining and supplying the requested values by whatever means it
3319 deems appropriate, including by retrieving them from one or more Policy Information Points. The **context**
3320 **handler** SHALL return the values of **attributes** that match the attribute designator or attribute selector
3321 and form them into a **bag** of values with the specified data-type. If no **attributes** from the request
3322 **context** match, then the **attribute** SHALL be considered missing. If the **attribute** is missing, then
3323 *MustBePresent* governs whether the attribute designator or attribute selector returns an empty **bag** or
3324 an “Indeterminate” result. If *MustBePresent* is “False” (default value), then a missing **attribute** SHALL
3325 result in an empty **bag**. If *MustBePresent* is “True”, then a missing **attribute** SHALL result in
3326 “Indeterminate”. This “Indeterminate” result SHALL be handled in accordance with the specification of the
3327 encompassing expressions, **rules**, **policies** and **policy sets**. If the result is “Indeterminate”, then the
3328 *AttributeId*, *DataType* and *Issuer* of the **attribute** MAY be listed in the **authorization decision** as
3329 described in Section 7.17. However, a **PDP** MAY choose not to return such information for security
3330 reasons.

3331 Regardless of any dynamic modifications of the request **context** during policy evaluation, the **PDP**
3332 SHALL behave as if each bag of **attribute** values is fully populated in the **context** before it is first tested,
3333 and is thereafter immutable during evaluation. (That is, every subsequent test of that **attribute** shall use
3334 the same bag of values that was initially tested.)

3335 7.3.6 Environment Attributes

3336 Standard *environment attributes* are listed in Section B.7. If a value for one of these *attributes* is
3337 supplied in the *decision request*, then the *context handler* SHALL use that value. Otherwise, the
3338 *context handler* SHALL supply a value. In the case of date and time *attributes*, the supplied value
3339 SHALL have the semantics of the "date and time that apply to the *decision request*".

3340 7.3.7 AttributeSelector evaluation

3341 An <AttributeSelector> element will be evaluated according to the following processing model.

3342

3343 NOTE: It is not necessary for an implementation to actually follow these steps. It is only
3344 necessary to produce results identical to those that would be produced by following these
3345 steps.

- 3346 1. Construct if the *attributes* category given by the *Category* attribute is not found or does not
3347 have a <Content> child element, then the return value is either "Indeterminate" or an empty *bag*
3348 as determined by the *MustBePresent* attribute; otherwise, construct an XML data structure
3349 suitable for xpath processing from the <Content> element in the *attributes* category given by
3350 the *Category* attribute. The data structure shall be constructed so that the document node of
3351 this structure contains a single document element which corresponds to the single child element
3352 of the <Content> element. The constructed data structure shall be equivalent to one that would
3353 result from parsing a stand-alone XML document consisting of the contents of the <Content>
3354 element (including any comment and processing-instruction markup). Namespace declarations
3355 which are not "visibly utilized", as defined by [exc-c14n], MAY not be present and MUST NOT be
3356 utilized by the XPath expression in step 3. The data structure must meet the requirements of the
3357 applicable xpath version.
- 3358 2. Select a context node for xpath processing from this data structure. If there is a
3359 *ContextSelectorId* attribute, the context node shall be the node selected by applying the
3360 XPath expression given in the *attribute* value of the designated *attribute* (in the *attributes*
3361 category given by the <AttributeSelector> *Category* attribute). It shall be an error if this
3362 evaluation returns no node or more than one node, in which case the return value MUST be an
3363 "Indeterminate" with a status code "urn:oasis:names:tc:xacml:1.0:status:syntax-error". If there is
3364 no *ContextSelectorId*, the document node of the data structure shall be the context node.
- 3365 3. Evaluate the XPath expression given in the *Path* attribute against the xml data structure, using
3366 the context node selected in the previous step. It shall be an error if this evaluation returns
3367 anything other than a sequence of nodes (possibly empty), in which case the
3368 <AttributeSelector> MUST return "Indeterminate" with a status code
3369 "urn:oasis:names:tc:xacml:1.0:status:syntax-error". If the evaluation returns an empty sequence
3370 of nodes, then the return value is either "Indeterminate" or an empty *bag* as determined by the
3371 *MustBePresent* attribute.
- 3372 4. If the data type is a primitive data type, convert the text value of each selected node to the
3373 desired data type, as specified in the *DataType* attribute. Each value shall be constructed using
3374 the appropriate constructor function from [XF] Section 5 listed below, corresponding to the
3375 specified data type.

3376 xs:string()

3377 xs:boolean()

3378 xs:integer()

3379 xs:double()

3380 xs:dateTime()

3381 xs:date()

3382 xs:time()

3383 xs:hexBinary()

3384 xs:base64Binary()

3386 xs:anyURI()
3387 xs:yearMonthDuration()
3388 xs:dayTimeDuration()
3389

3390 If the `DataType` is not one of the primitive types listed above, then the return values shall be
3391 constructed from the nodeset in a manner specified by the particular `DataType` extension
3392 specification. If the data type extension does not specify an appropriate constructor function, then
3393 the `<AttributeSelector>` MUST return "Indeterminate" with a status code
3394 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".
3395

3396 If an error occurs when converting the values returned by the XPath expression to the specified
3397 `DataType`, then the result of the `<AttributeSelector>` MUST be "Indeterminate", with a
3398 status code "urn:oasis:names:tc:xacml:1.0:status:processing-error"

3399 7.4 Expression evaluation

3400 XACML specifies expressions in terms of the elements listed below, of which the `<Apply>` and
3401 `<Condition>` elements recursively compose greater expressions. Valid expressions SHALL be type
3402 correct, which means that the types of each of the elements contained within `<Apply>` elements SHALL
3403 agree with the respective argument types of the function that is named by the `FunctionId` attribute.
3404 The resultant type of the `<Apply>` element SHALL be the resultant type of the function, which MAY be
3405 narrowed to a primitive data-type, or a **bag** of a primitive data-type, by type-unification. XACML defines
3406 an evaluation result of "Indeterminate", which is said to be the result of an invalid expression, or an
3407 operational error occurring during the evaluation of the expression.

3408 XACML defines these elements to be in the substitution group of the `<Expression>` element:

- 3409 • `<xacml:AttributeValue>`
- 3410 • `<xacml:AttributeDesignator>`
- 3411 • `<xacml:AttributeSelector>`
- 3412 • `<xacml:Apply>`
- 3413 • `<xacml:Function>`
- 3414 • `<xacml:VariableReference>`

3415 7.5 Arithmetic evaluation

3416 IEEE 754 [IEEE754] specifies how to evaluate arithmetic functions in a context, which specifies defaults
3417 for precision, rounding, etc. XACML SHALL use this specification for the evaluation of all integer and
3418 double functions relying on the Extended Default Context, enhanced with double precision:

3419 flags - all set to 0

3420 trap-enablers - all set to 0 (IEEE 854 §7) with the exception of the "division-by-zero" trap enabler,
3421 which SHALL be set to 1

3422 precision - is set to the designated double precision

3423 rounding - is set to round-half-even (IEEE 854 §4.1)

3424 7.6 Match evaluation

3425 The **attribute** matching element `<Match>` appears in the `<Target>` element of **rules**, **policies** and
3426 **policy sets**.

3427 This element represents a Boolean expression over **attributes** of the request **context**. A matching
3428 element contains a `MatchId` attribute that specifies the function to be used in performing the match
3429 evaluation, an `<AttributeValue>` and an `<AttributeDesignator>` or `<AttributeSelector>`
3430 element that specifies the **attribute** in the **context** that is to be matched against the specified value.

3431 The `MatchId` attribute SHALL specify a function that takes two arguments, returning a result type of
3432 "http://www.w3.org/2001/XMLSchema#boolean". The **attribute** value specified in the matching element
3433 SHALL be supplied to the `MatchId` function as its first argument. An element of the **bag** returned by the
3434 `<AttributeDesignator>` or `<AttributeSelector>` element SHALL be supplied to the `MatchId`
3435 function as its second argument, as explained below. The `DataType` of the `<AttributeValue>`
3436 SHALL match the data-type of the first argument expected by the `MatchId` function. The `DataType` of
3437 the `<AttributeDesignator>` or `<AttributeSelector>` element SHALL match the data-type of the
3438 second argument expected by the `MatchId` function.

3439 In addition, functions that are strictly within an extension to XACML MAY appear as a value for the
3440 `MatchId` attribute, and those functions MAY use data-types that are also extensions, so long as the
3441 extension function returns a Boolean result and takes two single base types as its inputs. The function
3442 used as the value for the `MatchId` attribute SHOULD be easily indexable. Use of non-indexable or
3443 complex functions may prevent efficient evaluation of **decision requests**.

3444 The evaluation semantics for a matching element is as follows. If an operational error were to occur while
3445 evaluating the `<AttributeDesignator>` or `<AttributeSelector>` element, then the result of the
3446 entire expression SHALL be "Indeterminate". If the `<AttributeDesignator>` or
3447 `<AttributeSelector>` element were to evaluate to an empty **bag**, then the result of the expression
3448 SHALL be "False". Otherwise, the `MatchId` function SHALL be applied between the
3449 `<AttributeValue>` and each element of the **bag** returned from the `<AttributeDesignator>` or
3450 `<AttributeSelector>` element. If at least one of those function applications were to evaluate to
3451 "True", then the result of the entire expression SHALL be "True". Otherwise, if at least one of the function
3452 applications results in "Indeterminate", then the result SHALL be "Indeterminate". Finally, if all function
3453 applications evaluate to "False", then the result of the entire expression SHALL be "False".

3454 It is also possible to express the semantics of a **target** matching element in a **condition**. For instance,
3455 the **target** match expression that compares a "**subject-name**" starting with the name "John" can be
3456 expressed as follows:

```
3457 <Match
3458 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
3459   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
3460     John.*
3461   </AttributeValue>
3462   <AttributeDesignator
3463     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
3464 subject"
3465     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
3466     DataType="http://www.w3.org/2001/XMLSchema#string"/>
3467 </Match>
```

3468 Alternatively, the same match semantics can be expressed as an `<Apply>` element in a **condition** by
3469 using the "urn:oasis:names:tc:xacml:3.0:function:any-of" function, as follows:

```
3470 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
3471   <Function
3472     FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match"/>
3473   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
3474     John.*
3475   </AttributeValue>
3476   <AttributeDesignator
3477     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
3478 subject"
3479     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
3480     DataType="http://www.w3.org/2001/XMLSchema#string"/>
3481 </Apply>
```

3482 **7.7 Target evaluation**

3483 An empty **target** matches any request. Otherwise the **target** value SHALL be "Match" if all the AnyOf
 3484 specified in the **target** match values in the request **context**. Otherwise, if any one of the AnyOf specified
 3485 in the **target** is "No Match", then the **target** SHALL be "No Match". Otherwise, the **target** SHALL be
 3486 "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" |

3487 *Table 1 Target match table*

3488 The AnyOf SHALL match values in the request **context** if at least one of their <AllOf> elements
 3489 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" |

3490 *Table 2 AnyOf match table*

3491 An AllOf SHALL match a value in the request **context** if the value of all its <Match> elements is "True".

3492 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" |

3493 *Table 3 AllOf match table*

3494 **7.8 VariableReference Evaluation**

3495 The <VariableReference> element references a single <VariableDefinition> element contained
 3496 within the same <Policy> element. A <VariableReference> that does not reference a particular
 3497 <VariableDefinition> element within the encompassing <Policy> element is called an undefined
 3498 reference. **Policies** with undefined references are invalid.

3499 In any place where a <VariableReference> occurs, it has the effect as if the text of the
 3500 <Expression> element defined in the <VariableDefinition> element replaces the
 3501 <VariableReference> element. Any evaluation scheme that preserves this semantic is acceptable.
 3502 For instance, the expression in the <VariableDefinition> element may be evaluated to a particular
 3503 value and cached for multiple references without consequence. (I.e. the value of an <Expression>
 3504 element remains the same for the entire **policy** evaluation.) This characteristic is one of the benefits of
 3505 XACML being a declarative language.

3506 A variable reference containing circular references is invalid. The PDP MUST detect circular references
 3507 either at policy loading time or during runtime evaluation. If the PDP detects a circular reference during

3508 runtime the variable reference evaluates to "Indeterminate" with status code
3509 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3510 7.9 Condition evaluation

3511 The **condition** value SHALL be "True" if the <Condition> element is absent, or if it evaluates to "True".
3512 Its value SHALL be "False" if the <Condition> element evaluates to "False". The **condition** value
3513 SHALL be "Indeterminate", if the expression contained in the <Condition> element evaluates to
3514 "Indeterminate."

3515 7.10 Extended Indeterminate

3516 Some **combining algorithms** are defined in terms of an extended set of "Indeterminate" values. The
3517 extended set associated with the "Indeterminate" contains the potential effect values which could have
3518 occurred if there would not have been an error causing the "Indeterminate". The possible extended set
3519 "Indeterminate" values are

- 3520 • "Indeterminate{D}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Deny",
3521 but not "Permit"
- 3522 • "Indeterminate{P}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Permit",
3523 but not "Deny"
- 3524 • "Indeterminate{DP}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Deny"
3525 or "Permit".

3526 The **combining algorithms** which are defined in terms of the extended "Indeterminate" make use of the
3527 additional information to allow for better treatment of errors in the algorithms.

3528 The final decision returned by a **PDP** cannot be an extended Indeterminate. Any such decision at the top
3529 level **policy** or **policy set** is returned as a plain Indeterminate in the response from the **PDP**.

3530 The tables in the following four sections define how extended "Indeterminate" values are produced during
3531 **Rule**, **Policy** and **PolicySet** evaluation.

3532 7.11 Rule evaluation

3533 A **rule** has a value that can be calculated by evaluating its contents. **Rule** evaluation involves separate
3534 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 |

3535 Table 4 Rule truth table.

3536 7.12 Policy evaluation

3537 The value of a **policy** SHALL be determined only by its contents, considered in relation to the contents of
3538 the request **context**. A **policy's** value SHALL be determined by evaluation of the **policy's target** and,
3539 according to the specified **rule-combining algorithm, rules**,.

3540 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 |

3541 *Table 5 Policy truth table*

3542 Note that none of the **rule-combining algorithms** defined by XACML 3.0 take parameters. However,
 3543 non-standard combining algorithms MAY take parameters. In such a case, the values of these
 3544 parameters associated with the **rules**, MUST be taken into account when evaluating the **policy**. The
 3545 parameters and their types should be defined in the specification of the combining algorithm. If the
 3546 implementation supports combiner parameters and if combiner parameters are present in a **policy**, then
 3547 the parameter values MUST be supplied to the combining algorithm implementation.

3548 7.13 Policy Set evaluation

3549 The value of a **policy set** SHALL be determined by its contents, considered in relation to the contents of
 3550 the request **context**. A **policy set**'s value SHALL be determined by evaluation of the **policy set**'s **target**,
 3551 and, according to the specified **policy-combining algorithm**, **policies** and **policy sets**,

3552 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 |

3553 *Table 6 Policy set truth table*

3554 Note that none of the **policy-combining algorithms** defined by XACML 3.0 take parameters. However,
 3555 non-standard combining algorithms MAY take parameters. In such a case, the values of these
 3556 parameters associated with the **policies**, MUST be taken into account when evaluating the **policy set**.
 3557 The parameters and their types should be defined in the specification of the combining algorithm. If the
 3558 implementation supports combiner parameters and if combiner parameters are present in a **policy**, then
 3559 the parameter values MUST be supplied to the combining algorithm implementation.

3560 7.14 Policy and Policy set value for Indeterminate Target

3561 If the **target** of a **policy** or **policy set** evaluates to “Indeterminate”, the value of the **policy** or **policy set**
 3562 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}" |
|--------------------|--------------------|

3563 Table 7 The value of a **policy** or **policy set** when the target is "Indeterminate".

3564 7.15 PolicySetIdReference and PolicyIdReference evaluation

3565 A policy set id reference or a policy id reference is evaluated by resolving the reference and evaluating
3566 the referenced policy set or policy.

3567 If resolving the reference fails, the reference evaluates to "Indeterminate" with status code
3568 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3569 A policy set id reference or a policy id reference containing circular references is invalid. The PDP MUST
3570 detect circular references either at policy loading time or during runtime evaluation. If the PDP detects a
3571 circular reference during runtime the reference evaluates to "Indeterminate" with status code
3572 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3573 7.16 Hierarchical resources

3574 It is often the case that a **resource** is organized as a hierarchy (e.g. file system, XML document). XACML
3575 provides several optional mechanisms for supporting hierarchical **resources**. These are described in the
3576 XACML Profile for Hierarchical Resources [**Hier**] and in the XACML Profile for Requests for Multiple
3577 Resources [**Multi**].

3578 7.17 Authorization decision

3579 In relation to a particular **decision request**, the **PDP** is defined by a **policy-combining algorithm** and a
3580 set of **policies** and/or **policy sets**. The **PDP** SHALL return a response **context** as if it had evaluated a
3581 single **policy set** consisting of this **policy-combining algorithm** and the set of **policies** and/or **policy**
3582 **sets**.

3583 The **PDP** MUST evaluate the **policy set** as specified in Sections 5 and 7. The **PDP** MUST return a
3584 response **context**, with one <Decision> element of value "Permit", "Deny", "Indeterminate" or
3585 "NotApplicable".

3586 If the **PDP** cannot make a **decision**, then an "Indeterminate" <Decision> element SHALL be returned.

3587 7.18 Obligations and advice

3588 A **rule**, **policy**, or **policy set** may contain one or more **obligation** or **advice** expressions. When such a
3589 **rule**, **policy**, or **policy set** is evaluated, the **obligation** or **advice** expression SHALL be evaluated to an
3590 **obligation** or **advice** respectively, which SHALL be passed up to the next level of evaluation (the
3591 enclosing or referencing **policy**, **policy set**, or **authorization decision**) only if the result of the **rule**,
3592 **policy**, or **policy set** being evaluated matches the value of the FulfillOn attribute of the **obligation** or
3593 the AppliesTo attribute of the **advice**. If any of the **attribute** assignment expressions in an **obligation**
3594 or **advice** expression with a matching FulfillOn or AppliesTo attribute evaluates to "Indeterminate",
3595 then the whole **rule**, **policy**, or **policy set** SHALL be "Indeterminate". If the FulfillOn or AppliesTo
3596 attribute does not match the result of the combining algorithm or the **rule** evaluation, then any
3597 indeterminate in an **obligation** or **advice** expression has no effect.

3598 As a consequence of this procedure, no **obligations** or **advice** SHALL be returned to the **PEP** if the **rule**,
3599 **policies**, or **policy sets** from which they are drawn are not evaluated, or if their evaluated result is
3600 "Indeterminate" or "NotApplicable", or if the **decision** resulting from evaluating the **rule**, **policy**, or **policy**
3601 **set** does not match the **decision** resulting from evaluating an enclosing **policy set**.

3602 If the **PDP**'s evaluation is viewed as a tree of **rules**, **policy sets** and **policies**, each of which returns
3603 "Permit" or "Deny", then the set of **obligations** and **advice** returned by the **PDP** to the **PEP** will include
3604 only the **obligations** and **advice** associated with those paths where the result at each level of evaluation
3605 is the same as the result being returned by the **PDP**. In situations where any lack of determinism is
3606 unacceptable, a deterministic combining algorithm, such as ordered-deny-overrides, should be used.

3607 Also see Section 7.2.

3608 7.19 Exception handling

3609 XACML specifies behavior for the **PDP** in the following situations.

3610 7.19.1 Unsupported functionality

3611 If the **PDP** attempts to evaluate a **policy set** or **policy** that contains an optional element type or function
3612 that the **PDP** does not support, then the **PDP** SHALL return a <Decision> value of "Indeterminate". If a
3613 <StatusCode> element is also returned, then its value SHALL be
3614 "urn:oasis:names:tc:xacml:1.0:status:syntax-error" in the case of an unsupported element type, and
3615 "urn:oasis:names:tc:xacml:1.0:status:processing-error" in the case of an unsupported function.

3616 7.19.2 Syntax and type errors

3617 If a **policy** that contains invalid syntax is evaluated by the XACML **PDP** at the time a **decision request** is
3618 received, then the result of that **policy** SHALL be "Indeterminate" with a `StatusCode` value of
3619 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".

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

3623 7.19.3 Missing attributes

3624 The absence of matching **attributes** in the request **context** for any of the attribute designators attribute or
3625 selectors that are found in the **policy** will result in an enclosing <AllOf> element to return a value of
3626 "Indeterminate", if the designator or selector has the `MustBePresent` XML attribute set to true, as
3627 described in Sections 5.29 and 5.30 and may result in a <Decision> element containing the
3628 "Indeterminate" value. If, in this case a status code is supplied, then the value

3629 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute"

3630 SHALL be used, to indicate that more information is needed in order for a definitive **decision** to be
3631 rendered. In this case, the <Status> element MAY list the names and data-types of any **attributes** that
3632 are needed by the **PDP** to refine its **decision** (see Section 5.58). A **PEP** MAY resubmit a refined request
3633 **context** in response to a <Decision> element contents of "Indeterminate" with a status code of

3634 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute"

3635 by adding **attribute** values for the **attribute** names that were listed in the previous response. When the
3636 **PDP** returns a <Decision> element contents of "Indeterminate", with a status code of

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

3638 it MUST NOT list the names and data-types of any **attribute** for which values were supplied in the original
3639 request. Note, this requirement forces the **PDP** to eventually return an **authorization decision** of
3640 "Permit", "Deny", or "Indeterminate" with some other status code, in response to successively-refined
3641 requests.

3642 7.20 Identifier equality

3643 XACML makes use of URIs and strings as identifiers. When such identifiers are compared for equality,
3644 the comparison MUST be done so that the identifiers are equal if they have the same length and the
3645 characters in the two identifiers are equal codepoint by codepoint.

3646 The following is a list of the identifiers which MUST use this definition of equality.

3647 The content of the element <XPathVersion>.

3648 The XML attribute `Value` in the element <StatusCode>.

3649 The XML attributes `Category`, `AttributeId`, `DataType` and `Issuer` in the element
3650 `<MissingAttributeDetail>`.

3651 The XML attribute `Category` in the element `<Attributes>`.

3652 The XML attributes `AttributeId` and `Issuer` in the element `<Attribute>`.

3653 The XML attribute `ObligationId` in the element `<Obligation>`.

3654 The XML attribute `AdviceId` in the element `<Advice>`.

3655 The XML attributes `AttributeId` and `Category` in the element `<AttributeAssignment>`.

3656 The XML attribute `ObligationId` in the element `<ObligationExpression>`.

3657 The XML attribute `AdviceId` in the element `<AdviceExpression>`.

3658 The XML attributes `AttributeId`, `Category` and `Issuer` in the element
3659 `<AttributeAssignmentExpression>`.

3660 The XML attributes `PolicySetId` and `PolicyCombiningAlgId` in the element `<PolicySet>`.

3661 The XML attribute `ParameterName` in the element `<CombinerParameter>`.

3662 The XML attribute `RuleIdRef` in the element `<RuleCombinerParameters>`.

3663 The XML attribute `PolicyIdRef` in the element `<PolicyCombinerParameters>`.

3664 The XML attribute `PolicySetIdRef` in the element `<PolicySetCombinerParameters>`.

3665 The anyURI in the content of the complex type `IdReferenceType`.

3666 The XML attributes `PolicyId` and `RuleCombiningAlgId` in the element `<Policy>`.

3667 The XML attribute `RuleId` in the element `<Rule>`.

3668 The XML attribute `MatchId` in the element `<Match>`.

3669 The XML attribute `VariableId` in the element `<VariableDefinition>`.

3670 The XML attribute `VariableId` in the element `<VariableReference>`.

3671 The XML attributes `Category`, `ContextSelectorId` and `DataType` in the element
3672 `<AttributeSelector>`.

3673 The XML attributes `Category`, `AttributeId`, `DataType` and `Issuer` in the element
3674 `<AttributeDesignator>`.

3675 The XML attribute `DataType` in the element `<AttributeValue>`.

3676 The XML attribute `FunctionId` in the element `<Function>`.

3677 The XML attribute `FunctionId` in the element `<Apply>`.

3678

3679 It is RECOMMENDED that extensions to XACML use the same definition of identifier equality for similar
3680 identifiers.

3681 It is RECOMMENDED that extensions which define identifiers do not define identifiers which could be
3682 easily misinterpreted by people as being subject to other kind of processing, such as URL character
3683 escaping, before matching.

3684 8 XACML extensibility points (non-normative)

3685 This section describes the points within the XACML model and schema where extensions can be added.

3686 8.1 Extensible XML attribute types

3687 The following XML attributes have values that are URIs. These may be extended by the creation of new
3688 URIs associated with new semantics for these attributes.

3689 `Category,`

3690 `AttributeId,`

3691 `DataType,`

3692 `FunctionId,`

3693 `MatchId,`

3694 `ObligationId,`

3695 `AdviceId,`

3696 `PolicyCombiningAlgId,`

3697 `RuleCombiningAlgId,`

3698 `StatusCodes,`

3699 `SubjectCategory.`

3700 See Section 5 for definitions of these *attribute* types.

3701 8.2 Structured attributes

3702 <AttributeValue> elements MAY contain an instance of a structured XML data-type. Section 7.3.1
3703 describes a number of standard techniques to identify data items within such a structured *attribute*.
3704 Listed here are some additional techniques that require XACML extensions.

- 3705 1. For a given structured data-type, a community of XACML users MAY define new *attribute*
3706 identifiers for each leaf sub-element of the structured data-type that has a type conformant with
3707 one of the XACML-defined primitive data-types. Using these new *attribute* identifiers, the *PEPs*
3708 or *context handlers* used by that community of users can flatten instances of the structured
3709 data-type into a sequence of individual <Attribute> elements. Each such <Attribute>
3710 element can be compared using the XACML-defined functions. Using this method, the structured
3711 data-type itself never appears in an <AttributeValue> element.
- 3712 2. A community of XACML users MAY define a new function that can be used to compare a value of
3713 the structured data-type against some other value. This method may only be used by *PDPs* that
3714 support the new function.

3715 9 Security and privacy considerations (non- 3716 normative)

3717 This section identifies possible security and privacy compromise scenarios that should be considered
3718 when implementing an XACML-based system. The section is informative only. It is left to the
3719 implementer to decide whether these compromise scenarios are practical in their environment and to
3720 select appropriate safeguards.

3721 9.1 Threat model

3722 We assume here that the adversary has access to the communication channel between the XACML
3723 actors and is able to interpret, insert, delete, and modify messages or parts of messages.

3724 Additionally, an actor may use information from a former message maliciously in subsequent transactions.
3725 It is further assumed that *rules* and *policies* are only as reliable as the actors that create and use them.
3726 Thus it is incumbent on each actor to establish appropriate trust in the other actors upon which it relies.
3727 Mechanisms for trust establishment are outside the scope of this specification.

3728 The messages that are transmitted between the actors in the XACML model are susceptible to attack by
3729 malicious third parties. Other points of vulnerability include the *PEP*, the *PDP*, and the *PAP*. While some
3730 of these entities are not strictly within the scope of this specification, their compromise could lead to the
3731 compromise of *access control* enforced by the *PEP*.

3732 It should be noted that there are other components of a distributed system that may be compromised,
3733 such as an operating system and the domain-name system (DNS) that are outside the scope of this
3734 discussion of threat models. Compromise in these components may also lead to a policy violation.

3735 The following sections detail specific compromise scenarios that may be relevant to an XACML system.

3736 9.1.1 Unauthorized disclosure

3737 XACML does not specify any inherent mechanisms to protect the confidentiality of the messages
3738 exchanged between actors. Therefore, an adversary could observe the messages in transit. Under
3739 certain security *policies*, disclosure of this information is a violation. Disclosure of *attributes* or the types
3740 of *decision requests* that a *subject* submits may be a breach of privacy policy. In the commercial
3741 sector, the consequences of unauthorized disclosure of personal data may range from embarrassment to
3742 the custodian, to imprisonment and/or large fines in the case of medical or financial data.

3743 Unauthorized disclosure is addressed by confidentiality safeguards.

3744 9.1.2 Message replay

3745 A message replay attack is one in which the adversary records and replays legitimate messages between
3746 XACML actors. This attack may lead to denial of service, the use of out-of-date information or
3747 impersonation.

3748 Prevention of replay attacks requires the use of message freshness safeguards.

3749 Note that encryption of the message does not mitigate a replay attack since the message is simply
3750 replayed and does not have to be understood by the adversary.

3751 9.1.3 Message insertion

3752 A message insertion attack is one in which the adversary inserts messages in the sequence of messages
3753 between XACML actors.

3754 The solution to a message insertion attack is to use mutual authentication and message sequence
3755 integrity safeguards between the actors. It should be noted that just using SSL mutual authentication is
3756 not sufficient. This only proves that the other party is the one identified by the *subject* of the X.509

3757 certificate. In order to be effective, it is necessary to confirm that the certificate **subject** is authorized to
3758 send the message.

3759 9.1.4 Message deletion

3760 A message deletion attack is one in which the adversary deletes messages in the sequence of messages
3761 between XACML actors. Message deletion may lead to denial of service. However, a properly designed
3762 XACML system should not render an incorrect **authorization decision** as a result of a message deletion
3763 attack.

3764 The solution to a message deletion attack is to use message sequence integrity safeguards between the
3765 actors.

3766 9.1.5 Message modification

3767 If an adversary can intercept a message and change its contents, then they may be able to alter an
3768 **authorization decision**. A message integrity safeguard can prevent a successful message modification
3769 attack.

3770 9.1.6 NotApplicable results

3771 A result of "NotApplicable" means that the **PDP** could not locate a **policy** whose **target** matched the
3772 information in the **decision request**. In general, it is highly recommended that a "Deny" **effect policy** be
3773 used, so that when a **PDP** would have returned "NotApplicable", a result of "Deny" is returned instead.

3774 In some security models, however, such as those found in many web servers, an **authorization decision**
3775 of "NotApplicable" is treated as equivalent to "Permit". There are particular security considerations that
3776 must be taken into account for this to be safe. These are explained in the following paragraphs.

3777 If "NotApplicable" is to be treated as "Permit", it is vital that the matching algorithms used by the **policy** to
3778 match elements in the **decision request** be closely aligned with the data syntax used by the applications
3779 that will be submitting the **decision request**. A failure to match will result in "NotApplicable" and be
3780 treated as "Permit". So an unintended failure to match may allow unintended **access**.

3781 Commercial http responders allow a variety of syntaxes to be treated equivalently. The "%" can be used
3782 to represent characters by hex value. The URL path "/./" provides multiple ways of specifying the same
3783 value. Multiple character sets may be permitted and, in some cases, the same printed character can be
3784 represented by different binary values. Unless the matching algorithm used by the **policy** is sophisticated
3785 enough to catch these variations, unintended **access** may be permitted.

3786 It may be safe to treat "NotApplicable" as "Permit" only in a closed environment where all applications that
3787 formulate a **decision request** can be guaranteed to use the exact syntax expected by the **policies**. In a
3788 more open environment, where **decision requests** may be received from applications that use any legal
3789 syntax, it is strongly recommended that "NotApplicable" NOT be treated as "Permit" unless matching
3790 **rules** have been very carefully designed to match all possible applicable inputs, regardless of syntax or
3791 type variations. Note, however, that according to Section 7.2, a **PEP** must deny **access** unless it
3792 receives an explicit "Permit" **authorization decision**.

3793 9.1.7 Negative rules

3794 A negative **rule** is one that is based on a **predicate** not being "True". If not used with care, negative
3795 **rules** can lead to policy violations, therefore some authorities recommend that they not be used.
3796 However, negative **rules** can be extremely efficient in certain cases, so XACML has chosen to include
3797 them. Nevertheless, it is recommended that they be used with care and avoided if possible.

3798 A common use for negative **rules** is to deny **access** to an individual or subgroup when their membership
3799 in a larger group would otherwise permit them **access**. For example, we might want to write a **rule** that
3800 allows all vice presidents to see the unpublished financial data, except for Joe, who is only a ceremonial
3801 vice president and can be indiscreet in his communications. If we have complete control over the
3802 administration of **subject attributes**, a superior approach would be to define "Vice President" and
3803 "Ceremonial Vice President" as distinct groups and then define **rules** accordingly. However, in some

3804 environments this approach may not be feasible. (It is worth noting in passing that referring to individuals
3805 in **rules** does not scale well. Generally, shared **attributes** are preferred.)

3806 If not used with care, negative **rules** can lead to policy violations in two common cases: when **attributes**
3807 are suppressed and when the base group changes. An example of suppressed **attributes** would be if we
3808 have a **policy** that **access** should be permitted, unless the **subject** is a credit risk. If it is possible that
3809 the **attribute** of being a credit risk may be unknown to the **PDP** for some reason, then unauthorized
3810 **access** may result. In some environments, the **subject** may be able to suppress the publication of
3811 **attributes** by the application of privacy controls, or the server or repository that contains the information
3812 may be unavailable for accidental or intentional reasons.

3813 An example of a changing base group would be if there is a **policy** that everyone in the engineering
3814 department may change software source code, except for secretaries. Suppose now that the department
3815 was to merge with another engineering department and the intent is to maintain the same **policy**.
3816 However, the new department also includes individuals identified as administrative assistants, who ought
3817 to be treated in the same way as secretaries. Unless the **policy** is altered, they will unintentionally be
3818 permitted to change software source code. Problems of this type are easy to avoid when one individual
3819 administers all **policies**, but when administration is distributed, as XACML allows, this type of situation
3820 must be explicitly guarded against.

3821 9.1.8 Denial of service

3822 A denial of service attack is one in which the adversary overloads an XACML actor with excessive
3823 computations or network traffic such that legitimate users cannot access the services provided by the
3824 actor.

3825 The urn:oasis:names:tc:xacml:3.0:function:access-permitted function may lead to hard to predict behavior
3826 in the **PDP**. It is possible that the function is invoked during the recursive invocations of the **PDP** such that
3827 loops are formed. Such loops may in some cases lead to large numbers of requests to be generated
3828 before the **PDP** can detect the loop and abort evaluation. Such loops could cause a denial of service at
3829 the **PDP**, either because of a malicious **policy** or because of a mistake in a **policy**.

3830 9.2 Safeguards

3831 9.2.1 Authentication

3832 Authentication provides the means for one party in a transaction to determine the identity of the other
3833 party in the transaction. Authentication may be in one direction, or it may be bilateral.

3834 Given the sensitive nature of **access control** systems, it is important for a **PEP** to authenticate the
3835 identity of the **PDP** to which it sends **decision requests**. Otherwise, there is a risk that an adversary
3836 could provide false or invalid **authorization decisions**, leading to a policy violation.

3837 It is equally important for a **PDP** to authenticate the identity of the **PEP** and assess the level of trust to
3838 determine what, if any, sensitive data should be passed. One should keep in mind that even simple
3839 "Permit" or "Deny" responses could be exploited if an adversary were allowed to make unlimited requests
3840 to a **PDP**.

3841 Many different techniques may be used to provide authentication, such as co-located code, a private
3842 network, a VPN, or digital signatures. Authentication may also be performed as part of the
3843 communication protocol used to exchange the **contexts**. In this case, authentication may be performed
3844 either at the message level or at the session level.

3845 9.2.2 Policy administration

3846 If the contents of **policies** are exposed outside of the **access control** system, potential **subjects** may
3847 use this information to determine how to gain unauthorized **access**.

3848 To prevent this threat, the repository used for the storage of **policies** may itself require **access control**.
3849 In addition, the <Status> element should be used to return values of missing **attributes** only when
3850 exposure of the identities of those **attributes** will not compromise security.

3851 9.2.3 Confidentiality

3852 Confidentiality mechanisms ensure that the contents of a message can be read only by the desired
3853 recipients and not by anyone else who encounters the message while it is in transit. There are two areas
3854 in which confidentiality should be considered: one is confidentiality during transmission; the other is
3855 confidentiality within a <Policy> element.

3856 9.2.3.1 Communication confidentiality

3857 In some environments it is deemed good practice to treat all data within an **access control** system as
3858 confidential. In other environments, **policies** may be made freely available for distribution, inspection,
3859 and audit. The idea behind keeping **policy** information secret is to make it more difficult for an adversary
3860 to know what steps might be sufficient to obtain unauthorized **access**. Regardless of the approach
3861 chosen, the security of the **access control** system should not depend on the secrecy of the **policy**.

3862 Any security considerations related to transmitting or exchanging XACML <Policy> elements are
3863 outside the scope of the XACML standard. While it is important to ensure that the integrity and
3864 confidentiality of <Policy> elements is maintained when they are exchanged between two parties, it is
3865 left to the implementers to determine the appropriate mechanisms for their environment.

3866 Communications confidentiality can be provided by a confidentiality mechanism, such as SSL. Using a
3867 point-to-point scheme like SSL may lead to other vulnerabilities when one of the end-points is
3868 compromised.

3869 9.2.3.2 Statement level confidentiality

3870 In some cases, an implementation may want to encrypt only parts of an XACML <Policy> element.

3871 The XML Encryption Syntax and Processing Candidate Recommendation from W3C can be used to
3872 encrypt all or parts of an XML document. This specification is recommended for use with XACML.

3873 It should go without saying that if a repository is used to facilitate the communication of cleartext (i.e.,
3874 unencrypted) **policy** between the **PAP** and **PDP**, then a secure repository should be used to store this
3875 sensitive data.

3876 9.2.4 Policy integrity

3877 The XACML **policy** used by the **PDP** to evaluate the request **context** is the heart of the system.

3878 Therefore, maintaining its integrity is essential. There are two aspects to maintaining the integrity of the
3879 **policy**. One is to ensure that <Policy> elements have not been altered since they were originally
3880 created by the **PAP**. The other is to ensure that <Policy> elements have not been inserted or deleted
3881 from the set of **policies**.

3882 In many cases, both aspects can be achieved by ensuring the integrity of the actors and implementing
3883 session-level mechanisms to secure the communication between actors. The selection of the appropriate
3884 mechanisms is left to the implementers. However, when **policy** is distributed between organizations to
3885 be acted on at a later time, or when the **policy** travels with the protected **resource**, it would be useful to
3886 sign the **policy**. In these cases, the XML Signature Syntax and Processing standard from W3C is
3887 recommended to be used with XACML.

3888 Digital signatures should only be used to ensure the integrity of the statements. Digital signatures should
3889 not be used as a method of selecting or evaluating **policy**. That is, the **PDP** should not request a **policy**
3890 based on who signed it or whether or not it has been signed (as such a basis for selection would, itself,
3891 be a matter of policy). However, the **PDP** must verify that the key used to sign the **policy** is one
3892 controlled by the purported **issuer** of the **policy**. The means to do this are dependent on the specific
3893 signature technology chosen and are outside the scope of this document.

3894 9.2.5 Policy identifiers

3895 Since **policies** can be referenced by their identifiers, it is the responsibility of the **PAP** to ensure that
3896 these are unique. Confusion between identifiers could lead to misidentification of the **applicable policy**.

3897 This specification is silent on whether a **PAP** must generate a new identifier when a **policy** is modified or
3898 may use the same identifier in the modified **policy**. This is a matter of administrative practice. However,
3899 care must be taken in either case. If the identifier is reused, there is a danger that other **policies** or
3900 **policy sets** that reference it may be adversely affected. Conversely, if a new identifier is used, these
3901 other **policies** may continue to use the prior **policy**, unless it is deleted. In either case the results may
3902 not be what the **policy** administrator intends.

3903 If a **PDP** is provided with **policies** from distinct sources which might not be fully trusted, as in the use of
3904 the administration profile [**XACMLAdmin**], there is a concern that someone could intentionally publish a
3905 **policy** with an id which collides with another **policy**. This could cause **policy** references that point to the
3906 wrong **policy**, and may cause other unintended consequences in an implementation which is predicated
3907 upon having unique **policy** identifiers.

3908 If this issue is a concern it is RECOMMENDED that distinct **policy** issuers or sources are assigned
3909 distinct namespaces for **policy** identifiers. One method is to make sure that the **policy** identifier begins
3910 with a string which has been assigned to the particular **policy** issuer or source. The remainder of the
3911 **policy** identifier is an issuer-specific unique part. For instance, Alice from Example Inc. could be assigned
3912 the **policy** identifiers which begin with `http://example.com/xacml/policyId/alice/`. The **PDP** or another
3913 trusted component can then verify that the authenticated source of the **policy** is Alice at Example Inc, or
3914 otherwise reject the **policy**. Anyone else will be unable to publish **policies** with identifiers which collide
3915 with the **policies** of Alice.

3916 9.2.6 Trust model

3917 Discussions of authentication, integrity and confidentiality safeguards necessarily assume an underlying
3918 trust model: how can one actor come to believe that a given key is uniquely associated with a specific,
3919 identified actor so that the key can be used to encrypt data for that actor or verify signatures (or other
3920 integrity structures) from that actor? Many different types of trust models exist, including strict
3921 hierarchies, distributed authorities, the Web, the bridge, and so on.

3922 It is worth considering the relationships between the various actors of the **access control** system in terms
3923 of the interdependencies that do and do not exist.

- 3924 • None of the entities of the authorization system are dependent on the **PEP**. They may collect data
3925 from it, (for example authentication data) but are responsible for verifying it themselves.
- 3926 • The correct operation of the system depends on the ability of the **PEP** to actually enforce **policy**
3927 **decisions**.
- 3928 • The **PEP** depends on the **PDP** to correctly evaluate **policies**. This in turn implies that the **PDP** is
3929 supplied with the correct inputs. Other than that, the **PDP** does not depend on the **PEP**.
- 3930 • The **PDP** depends on the **PAP** to supply appropriate **policies**. The **PAP** is not dependent on other
3931 components.

3932 9.2.7 Privacy

3933 It is important to be aware that any transactions that occur with respect to **access control** may reveal
3934 private information about the actors. For example, if an XACML **policy** states that certain data may only
3935 be read by **subjects** with "Gold Card Member" status, then any transaction in which a **subject** is
3936 permitted **access** to that data leaks information to an adversary about the **subject's** status. Privacy
3937 considerations may therefore lead to encryption and/or to **access control** requirements surrounding the
3938 enforcement of XACML **policy** instances themselves: confidentiality-protected channels for the
3939 request/response protocol messages, protection of **subject attributes** in storage and in transit, and so
3940 on.

3941 Selection and use of privacy mechanisms appropriate to a given environment are outside the scope of
3942 XACML. The **decision** regarding whether, how, and when to deploy such mechanisms is left to the
3943 implementers associated with the environment.

3944 **9.3 Unicode security issues**

3945 There are many security considerations related to use of Unicode. An XACML implementation SHOULD
3946 follow the advice given in the relevant version of **[UTR36]**.

3947 **9.4 Identifier equality**

3948 Section 7.20 defines the identifier equality operation for XACML. This definition of equality does not do
3949 any kind of canonicalization or escaping of characters. The identifiers defined in the XACML specification
3950 have been selected to not include any ambiguity regarding these aspects. It is RECOMMENDED that
3951 identifiers defined by extensions also do not introduce any identifiers which might be mistaken for being
3952 subject to processing, like for instance URL character encoding using “%”.

3953 **10 Conformance**

3954 **10.1 Introduction**

3955 The XACML specification addresses the following aspect of conformance:

3956 The XACML specification defines a number of functions, etc. that have somewhat special applications,
3957 therefore they are not required to be implemented in an implementation that claims to conform with the
3958 OASIS standard.

3959 **10.2 Conformance tables**

3960 This section lists those portions of the specification that **MUST** be included in an implementation of a **PDP**
3961 that claims to conform to XACML v3.0. A set of test cases has been created to assist in this process.
3962 These test cases can be located from the OASIS XACML TC Web page. The site hosting the test cases
3963 contains a full description of the test cases and how to execute them.

3964 Note: "M" means mandatory-to-implement. "O" means optional.

3965 The implementation **MUST** follow sections 5, 6, 7, Appendix A, Appendix B and Appendix C where they
3966 apply to implemented items in the following tables.

3967 **10.2.1 Schema elements**

3968 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 |

3969 **10.2.2 Identifier Prefixes**

3970 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 |

3971 **10.2.3 Algorithms**

3972 The implementation MUST include the **rule-** and **policy-combining algorithms** associated with the
 3973 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 |

3974 10.2.4 Status Codes

3975 Implementation support for the <StatusCode> element is optional, but if the element is supported, then
3976 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 |

3977 10.2.5 Attributes

3978 The implementation MUST support the **attributes** associated with the following identifiers as specified by
3979 XACML. If values for these **attributes** are not present in the **decision request**, then their values MUST
3980 be supplied by the **context handler**. So, unlike most other **attributes**, their semantics are not
3981 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 |

3982 10.2.6 Identifiers

3983 The implementation MUST use the **attributes** associated with the following identifiers in the way XACML
3984 has defined. This requirement pertains primarily to implementations of a **PAP** or **PEP** that uses XACML,
3985 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> | <u>O</u> |
| urn:oasis:names:tc:xacml:1.0:action:action-id | O |
| <u>urn:oasis:names:tc:xacml:1.0:action:action-namespace</u> | <u>O</u> |
| urn:oasis:names:tc:xacml:1.0:action:implied-action | O |

3986 **10.2.7 Data-types**

3987 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 |

3988 **10.2.8 Functions**

3989 The implementation MUST properly process those functions associated with the identifiers marked with
3990 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 |

3991 **10.2.9 Identifiers planned for future deprecation**

3992 These identifiers are associated with previous versions of XACML and newer alternatives exist in XACML
3993 3.0. They are planned to be deprecated at some unspecified point in the future. It is RECOMMENDED
3994 that these identifiers not be used in new policies and requests.

3995 The implementation MUST properly process those features associated with the identifiers marked with an
3996 "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 |

3997

3998

Appendix A. Data-types and functions (normative)

3999

A.1 Introduction

4000 This section specifies the data-types and functions used in XACML to create *predicates* for *conditions*
4001 and *target* matches.

4002 This specification combines the various standards set forth by IEEE and ANSI for string representation of
4003 numeric values, as well as the evaluation of arithmetic functions. It describes the primitive data-types and
4004 *bags*. The standard functions are named and their operational semantics are described.

A.2 Data-types

4006 Although XML instances represent all data-types as strings, an XACML *PDP* must operate on types of
4007 data that, while they have string representations, are not just strings. Types such as Boolean, integer,
4008 and double MUST be converted from their XML string representations to values that can be compared
4009 with values in their domain of discourse, such as numbers. The following primitive data-types are
4010 specified for use with XACML and have explicit data representations:

- 4011 • <http://www.w3.org/2001/XMLSchema#string>
- 4012 • <http://www.w3.org/2001/XMLSchema#boolean>
- 4013 • <http://www.w3.org/2001/XMLSchema#integer>
- 4014 • <http://www.w3.org/2001/XMLSchema#double>
- 4015 • <http://www.w3.org/2001/XMLSchema#time>
- 4016 • <http://www.w3.org/2001/XMLSchema#date>
- 4017 • <http://www.w3.org/2001/XMLSchema#dateTime>
- 4018 • <http://www.w3.org/2001/XMLSchema#anyURI>
- 4019 • <http://www.w3.org/2001/XMLSchema#hexBinary>
- 4020 • <http://www.w3.org/2001/XMLSchema#base64Binary>
- 4021 • <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 4022 • <http://www.w3.org/2001/XMLSchema#yearMonthDuration>
- 4023 • <urn:oasis:names:tc:xacml:1.0:data-type:x500Name>
- 4024 • <urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name>
- 4025 • <urn:oasis:names:tc:xacml:2.0:data-type:ipAddress>
- 4026 • <urn:oasis:names:tc:xacml:2.0:data-type:dnsName>
- 4027 • <urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression>

4028 For the sake of improved interoperability, it is RECOMMENDED that all time references be in UTC time.

4029 An XACML *PDP* SHALL be capable of converting string representations into various primitive data-types.
4030 For doubles, XACML SHALL use the conversions described in [IEEE754].

4031 XACML defines four data-types representing identifiers for *subjects* or *resources*; these are:

- 4032 “urn:oasis:names:tc:xacml:1.0:data-type:x500Name”,
- 4033 “urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name”
- 4034 “urn:oasis:names:tc:xacml:2.0:data-type:ipAddress” and
- 4035 “urn:oasis:names:tc:xacml:2.0:data-type:dnsName”

4036 These types appear in several standard applications, such as TLS/SSL and electronic mail.

4037 X.500 directory name

4038 The "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" primitive type represents an ITU-T Rec.
4039 X.520 Distinguished Name. The valid syntax for such a name is described in IETF RFC 2253
4040 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished
4041 Names".

4042 **RFC 822 name**

4043 The "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" primitive type represents an electronic
4044 mail address. The valid syntax for such a name is described in IETF RFC 2821, Section 4.1.2,
4045 Command Argument Syntax, under the term "Mailbox".

4046 **IP address**

4047 The "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" primitive type represents an IPv4 or IPv6
4048 network address, with optional mask and optional port or port range. The syntax SHALL be:

4049 ipAddress = address ["/" mask] [":" [portrange]]

4050 For an IPv4 address, the address and mask are formatted in accordance with the syntax for a
4051 "host" in IETF RFC 2396 "Uniform Resource Identifiers (URI): Generic Syntax", section 3.2.

4052 For an IPv6 address, the address and mask are formatted in accordance with the syntax for an
4053 "ipv6reference" in IETF RFC 2732 "Format for Literal IPv6 Addresses in URL's". (Note that an
4054 IPv6 address or mask, in this syntax, is enclosed in literal "[" "]" brackets.)

4055 **DNS name**

4056 The "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" primitive type represents a Domain
4057 Name Service (DNS) host name, with optional port or port range. The syntax SHALL be:

4058 dnsName = hostname [":" portrange]

4059 The hostname is formatted in accordance with IETF RFC 2396 "Uniform Resource Identifiers
4060 (URI): Generic Syntax", section 3.2, except that a wildcard "*" may be used in the left-most
4061 component of the hostname to indicate "any subdomain" under the domain specified to its right.

4062 For both the "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" and
4063 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" data-types, the port or port range syntax
4064 SHALL be

4065 portrange = portnumber | "-"portnumber | portnumber "-"portnumber

4066 where "portnumber" is a decimal port number. If the port number is of the form "-x", where "x" is
4067 a port number, then the range is all ports numbered "x" and below. If the port number is of the
4068 form "x-", then the range is all ports numbered "x" and above. [This syntax is taken from the Java
4069 SocketPermission.]

4070 **XPath expression**

4071 The "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" primitive type represents an
4072 XPath expression over the XML in a <Content> element. The syntax is defined by the XPath
4073 W3C recommendation. The content of this data type also includes the context in which
4074 namespaces prefixes in the expression are resolved, which distinguishes it from a plain string and
4075 the XACML **attribute** category of the <Content> element to which it applies. When the value is
4076 encoded in an <AttributeValue> element, the namespace context is given by the [in-scope](#)
4077 [namespaces](#) (see [\[INFOSET\]](#)) of the <AttributeValue> element, and an XML attribute called
4078 XPathCategory gives the category of the <Content> element where the expression applies.

4079 The XPath expression MUST be evaluated in a context which is equivalent of a stand alone XML
4080 document with the only child of the <Content> element as the document element. Namespace
4081 declarations which are not "visibly utilized", as defined by [\[exc-c14n\]](#), MAY not be present and
4082 MUST NOT be utilized by the XPath expression. The context node of the XPath expression is the
4083 document node of this stand alone document.

4084 A.3 Functions

4085 XACML specifies the following functions. Unless otherwise specified, if an argument of one of these
4086 functions were to evaluate to "Indeterminate", then the function SHALL be set to "Indeterminate".

4087 Note that in each case an implementation is conformant as long as it produces the same result as is
4088 specified here, regardless of how and in what order the implementation behaves internally.

4089 A.3.1 Equality predicates

4090 The following functions are the equality functions for the various primitive types. Each function for a
4091 particular data-type follows a specified standard convention for that data-type.

- 4092 • urn:oasis:names:tc:xacml:1.0:function:string-equal

4093 This function SHALL take two arguments of data-type
4094 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
4095 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if
4096 the value of both of its arguments are of equal length and each string is determined to be equal.
4097 Otherwise, it SHALL return "False". The comparison SHALL use Unicode codepoint collation, as
4098 defined for the identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.

- 4099 • urn:oasis:names:tc:xacml:3.0:function:string-equal-ignore-case

4100 This function SHALL take two arguments of data-type
4101 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
4102 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be "True" if and only if the
4103 two strings are equal as defined by urn:oasis:names:tc:xacml:1.0:function:string-equal after they
4104 have both been converted to lower case with urn:oasis:names:tc:xacml:1.0:function:string-
4105 normalize-to-lower-case.

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

4107 This function SHALL take two arguments of data-type
4108 "http://www.w3.org/2001/XMLSchema#boolean" and SHALL return an
4109 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if
4110 the arguments are equal. Otherwise, it SHALL return "False".

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

4112 This function SHALL take two arguments of data-type
4113 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return an
4114 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if
4115 the two arguments represent the same number.

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

4117 This function SHALL take two arguments of data-type
4118 "http://www.w3.org/2001/XMLSchema#double" and SHALL return an
4119 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation on doubles
4120 according to IEEE 754 **[IEEE754]**.

- 4121 • urn:oasis:names:tc:xacml:1.0:function:date-equal

4122 This function SHALL take two arguments of data-type
4123 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an
4124 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to
4125 the "op:date-equal" function **[XF]** Section 10.4.9.

- 4126 • urn:oasis:names:tc:xacml:1.0:function:time-equal

4127 This function SHALL take two arguments of data-type
4128 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
4129 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to
4130 the "op:time-equal" function **[XF]** Section 10.4.12.

- 4131 • urn:oasis:names:tc:xacml:1.0:function:dateTime-equal
- 4132 This function SHALL take two arguments of data-type
- 4133 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an
- 4134 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to
- 4135 the "op:dateTime-equal" function **[XF]** Section 10.4.6.
- 4136 • urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal
- 4137 This function SHALL take two arguments of data-type
- 4138 "http://www.w3.org/2001/XMLSchema#dayTimeDuration" and SHALL return an
- 4139 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation
- 4140 according to the "op:duration-equal" function **[XF]** Section 10.4.5. Note that the lexical
- 4141 representation of each argument MUST be converted to a value expressed in fractional seconds
- 4142 **[XF]** Section 10.3.2.
- 4143 • urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal
- 4144 This function SHALL take two arguments of data-type
- 4145 "http://www.w3.org/2001/XMLSchema#yearMonthDuration" and SHALL return an
- 4146 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation
- 4147 according to the "op:duration-equal" function **[XF]** Section 10.4.5. Note that the lexical
- 4148 representation of each argument MUST be converted to a value expressed in fractional seconds
- 4149 **[XF]** Section 10.3.2.
- 4150 • urn:oasis:names:tc:xacml:1.0:function:anyURI-equal
- 4151 This function SHALL take two arguments of data-type
- 4152 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return an
- 4153 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL convert the arguments to
- 4154 strings with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI and return "True" if and
- 4155 only if the values of the two arguments are equal on a codepoint-by-codepoint basis.
- 4156 • urn:oasis:names:tc:xacml:1.0:function:x500Name-equal
- 4157 This function SHALL take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name"
- 4158 and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if
- 4159 and only if each Relative Distinguished Name (RDN) in the two arguments matches. Otherwise,
- 4160 it SHALL return "False". Two RDNs shall be said to match if and only if the result of the following
- 4161 operations is "True" .
- 4162 1. Normalize the two arguments according to IETF RFC 2253 "Lightweight Directory Access
 - 4163 Protocol (v3): UTF-8 String Representation of Distinguished Names".
 - 4164 2. If any RDN contains multiple attributeTypeAndValue pairs, re-order the Attribute
 - 4165 ValuePairs in that RDN in ascending order when compared as octet strings (described in
 - 4166 ITU-T Rec. X.690 (1997 E) Section 11.6 "Set-of components").
 - 4167 3. Compare RDNs using the rules in IETF RFC 3280 "Internet X.509 Public Key
 - 4168 Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Section 4.1.2.4
 - 4169 "Issuer".
- 4170 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal
- 4171 This function SHALL take two arguments of data-type "urn:oasis:names:tc:xacml:1.0:data-
- 4172 type:rfc822Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It
- 4173 SHALL return "True" if and only if the two arguments are equal. Otherwise, it SHALL return
- 4174 "False". An RFC822 name consists of a local-part followed by "@" followed by a domain-part.
- 4175 The local-part is case-sensitive, while the domain-part (which is usually a DNS host name) is not
- 4176 case-sensitive. Perform the following operations:
- 4177 1. Normalize the domain-part of each argument to lower case
 - 4178 2. Compare the expressions by applying the function
 - 4179 "urn:oasis:names:tc:xacml:1.0:function:string-equal" to the normalized arguments.

- 4180 • urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal
 - 4181 This function SHALL take two arguments of data-type
 - 4182 "http://www.w3.org/2001/XMLSchema#hexBinary" and SHALL return an
 - 4183 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences
 - 4184 represented by the value of both arguments have equal length and are equal in a conjunctive,
 - 4185 point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function.
 - 4186 Otherwise, it SHALL return "False". The conversion from the string representation to an octet
 - 4187 sequence SHALL be as specified in [XS] Section 3.2.15.
- 4188 • urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal
 - 4189 This function SHALL take two arguments of data-type
 - 4190 "http://www.w3.org/2001/XMLSchema#base64Binary" and SHALL return an
 - 4191 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences
 - 4192 represented by the value of both arguments have equal length and are equal in a conjunctive,
 - 4193 point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function.
 - 4194 Otherwise, it SHALL return "False". The conversion from the string representation to an octet
 - 4195 sequence SHALL be as specified in [XS] Section 3.2.16.

4196 A.3.2 Arithmetic functions

4197 All of the following functions SHALL take two arguments of the specified data-type, integer, or double,
 4198 and SHALL return an element of integer or double data-type, respectively. However, the "add" and
 4199 "multiply" functions MAY take more than two arguments. Each function evaluation operating on doubles
 4200 SHALL proceed as specified by their logical counterparts in IEEE 754 [IEEE754]. For all of these
 4201 functions, if any argument is "Indeterminate", then the function SHALL evaluate to "Indeterminate". In the
 4202 case of the divide functions, if the divisor is zero, then the function SHALL evaluate to "Indeterminate".

- 4203 • urn:oasis:names:tc:xacml:1.0:function:integer-add
 - 4204 This function MUST accept two or more arguments.
- 4205 • urn:oasis:names:tc:xacml:1.0:function:double-add
 - 4206 This function MUST accept two or more arguments.
- 4207 • urn:oasis:names:tc:xacml:1.0:function:integer-subtract
 - 4208 The result is the second argument subtracted from the first argument.
- 4209 • urn:oasis:names:tc:xacml:1.0:function:double-subtract
 - 4210 The result is the second argument subtracted from the first argument.
- 4211 • urn:oasis:names:tc:xacml:1.0:function:integer-multiply
 - 4212 This function MUST accept two or more arguments.
- 4213 • urn:oasis:names:tc:xacml:1.0:function:double-multiply
 - 4214 This function MUST accept two or more arguments.
- 4215 • urn:oasis:names:tc:xacml:1.0:function:integer-divide
 - 4216 The result is the first argument divided by the second argument.
- 4217 • urn:oasis:names:tc:xacml:1.0:function:double-divide
 - 4218 The result is the first argument divided by the second argument.
- 4219 • urn:oasis:names:tc:xacml:1.0:function:integer-mod
 - 4220 The result is remainder of the first argument divided by the second argument.

4221 The following functions SHALL take a single argument of the specified data-type. The round and floor
 4222 functions SHALL take a single argument of data-type "http://www.w3.org/2001/XMLSchema#double" and
 4223 return a value of the data-type "http://www.w3.org/2001/XMLSchema#double".

- 4224 • urn:oasis:names:tc:xacml:1.0:function:integer-abs

- 4225 • urn:oasis:names:tc:xacml:1.0:function:double-abs
- 4226 • urn:oasis:names:tc:xacml:1.0:function:round
- 4227 • urn:oasis:names:tc:xacml:1.0:function:floor

4228 **A.3.3 String conversion functions**

4229 The following functions convert between values of the data-type
4230 “http://www.w3.org/2001/XMLSchema#string” primitive types.

- 4231 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-space
 - 4232 This function SHALL take one argument of data-type
 - 4233 “http://www.w3.org/2001/XMLSchema#string” and SHALL normalize the value by stripping off all
 - 4234 leading and trailing white space characters. The whitespace characters are defined in the
 - 4235 metasympol S (Production 3) of [XML].
- 4236 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case
 - 4237 This function SHALL take one argument of data-type
 - 4238 “http://www.w3.org/2001/XMLSchema#string” and SHALL normalize the value by converting each
 - 4239 upper case character to its lower case equivalent. Case mapping shall be done as specified for
 - 4240 the fn:lower-case function in [XF] with no tailoring for particular languages or environments.

4241 **A.3.4 Numeric data-type conversion functions**

4242 The following functions convert between the data-type “http://www.w3.org/2001/XMLSchema#integer”
4243 and” http://www.w3.org/2001/XMLSchema#double” primitive types.

- 4244 • urn:oasis:names:tc:xacml:1.0:function:double-to-integer
 - 4245 This function SHALL take one argument of data-type
 - 4246 “http://www.w3.org/2001/XMLSchema#double” and SHALL truncate its numeric value to a whole
 - 4247 number and return an element of data-type “http://www.w3.org/2001/XMLSchema#integer”.
- 4248 • urn:oasis:names:tc:xacml:1.0:function:integer-to-double
 - 4249 This function SHALL take one argument of data-type
 - 4250 “http://www.w3.org/2001/XMLSchema#integer” and SHALL promote its value to an element of
 - 4251 data-type “http://www.w3.org/2001/XMLSchema#double” with the same numeric value. If the
 - 4252 integer argument is outside the range which can be represented by a double, the result SHALL
 - 4253 be Indeterminate, with the status code “urn:oasis:names:tc:xacml:1.0:status:processing-error”.

4254 **A.3.5 Logical functions**

4255 This section contains the specification for logical functions that operate on arguments of data-type
4256 “http://www.w3.org/2001/XMLSchema#boolean”.

- 4257 • urn:oasis:names:tc:xacml:1.0:function:or
 - 4258 This function SHALL return "False" if it has no arguments and SHALL return "True" if at least one
 - 4259 of its arguments evaluates to "True". The order of evaluation SHALL be from first argument to
 - 4260 last. The evaluation SHALL stop with a result of "True" if any argument evaluates to "True",
 - 4261 leaving the rest of the arguments unevaluated.
- 4262 • urn:oasis:names:tc:xacml:1.0:function:and
 - 4263 This function SHALL return "True" if it has no arguments and SHALL return "False" if one of its
 - 4264 arguments evaluates to "False". The order of evaluation SHALL be from first argument to last.
 - 4265 The evaluation SHALL stop with a result of "False" if any argument evaluates to "False", leaving
 - 4266 the rest of the arguments unevaluated.
- 4267 • urn:oasis:names:tc:xacml:1.0:function:n-of
 - 4268 The first argument to this function SHALL be of data-type
 - 4269 http://www.w3.org/2001/XMLSchema#integer. The remaining arguments SHALL be of data-type

4270 http://www.w3.org/2001/XMLSchema#boolean. The first argument specifies the minimum
4271 number of the remaining arguments that MUST evaluate to "True" for the expression to be
4272 considered "True". If the first argument is 0, the result SHALL be "True". If the number of
4273 arguments after the first one is less than the value of the first argument, then the expression
4274 SHALL result in "Indeterminate". The order of evaluation SHALL be: first evaluate the integer
4275 value, and then evaluate each subsequent argument. The evaluation SHALL stop and return
4276 "True" if the specified number of arguments evaluate to "True". The evaluation of arguments
4277 SHALL stop if it is determined that evaluating the remaining arguments will not satisfy the
4278 requirement.

4279 • urn:oasis:names:tc:xacml:1.0:function:not

4280 This function SHALL take one argument of data-type
4281 "http://www.w3.org/2001/XMLSchema#boolean". If the argument evaluates to "True", then the
4282 result of the expression SHALL be "False". If the argument evaluates to "False", then the result
4283 of the expression SHALL be "True".

4284 Note: When evaluating and, or, or n-of, it **MAY NOT** **may not** be necessary to attempt a full evaluation of
4285 each argument in order to determine whether the evaluation of the argument would result in
4286 "Indeterminate". Analysis of the argument regarding the availability of its **attributes**, or other analysis
4287 regarding errors, such as "divide-by-zero", may render the argument error free. Such arguments
4288 occurring in the expression in a position after the evaluation is stated to stop need not be processed.

4289 A.3.6 Numeric comparison functions

4290 These functions form a minimal set for comparing two numbers, yielding a Boolean result. For doubles
4291 they SHALL comply with the rules governed by IEEE 754 [IEEE754].

- 4292 • urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
- 4293 • urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal
- 4294 • urn:oasis:names:tc:xacml:1.0:function:integer-less-than
- 4295 • urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal
- 4296 • urn:oasis:names:tc:xacml:1.0:function:double-greater-than
- 4297 • urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal
- 4298 • urn:oasis:names:tc:xacml:1.0:function:double-less-than
- 4299 • urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal

4300 A.3.7 Date and time arithmetic functions

4301 These functions perform arithmetic operations with date and time.

4302 • urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration

4303 This function SHALL take two arguments, the first SHALL be of data-type
4304 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be of data-type
4305 "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It SHALL return a result of
4306 "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL return the value by
4307 adding the second argument to the first argument according to the specification of adding
4308 durations to date and time [XS] Appendix E.

4309 • urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration

4310 This function SHALL take two arguments, the first SHALL be a
4311 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a
4312 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
4313 "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL return the value by
4314 adding the second argument to the first argument according to the specification of adding
4315 durations to date and time [XS] Appendix E.

- 4316 • urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
 - 4317 This function SHALL take two arguments, the first SHALL be a
 - 4318 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a
 - 4319 "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It SHALL return a result of
 - 4320 "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration,
 - 4321 then this function SHALL return the value by adding the corresponding negative duration, as per
 - 4322 the specification [XS] Appendix E. If the second argument is a negative duration, then the result
 - 4323 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-
 - 4324 dayTimeDuration" had been applied to the corresponding positive duration.
- 4325 • urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
 - 4326 This function SHALL take two arguments, the first SHALL be a
 - 4327 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a
 - 4328 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
 - 4329 "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration,
 - 4330 then this function SHALL return the value by adding the corresponding negative duration, as per
 - 4331 the specification [XS] Appendix E. If the second argument is a negative duration, then the result
 - 4332 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-
 - 4333 yearMonthDuration" had been applied to the corresponding positive duration.
- 4334 • urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
 - 4335 This function SHALL take two arguments, the first SHALL be a
 - 4336 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a
 - 4337 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
 - 4338 "http://www.w3.org/2001/XMLSchema#date". This function SHALL return the value by adding the
 - 4339 second argument to the first argument according to the specification of adding duration to date
 - 4340 [XS] Appendix E.
- 4341 • urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration
 - 4342 This function SHALL take two arguments, the first SHALL be a
 - 4343 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a
 - 4344 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
 - 4345 "http://www.w3.org/2001/XMLSchema#date". If the second argument is a positive duration, then
 - 4346 this function SHALL return the value by adding the corresponding negative duration, as per the
 - 4347 specification [XS] Appendix E. If the second argument is a negative duration, then the result
 - 4348 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration"
 - 4349 had been applied to the corresponding positive duration.

4350 **A.3.8 Non-numeric comparison functions**

4351 These functions perform comparison operations on two arguments of non-numerical types.

- 4352 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than
 - 4353 This function SHALL take two arguments of data-type
 - 4354 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
 - 4355 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
 - 4356 argument is lexicographically strictly greater than the second argument. Otherwise, it SHALL
 - 4357 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
 - 4358 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].
- 4359 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal
 - 4360 This function SHALL take two arguments of data-type
 - 4361 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
 - 4362 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
 - 4363 argument is lexicographically greater than or equal to the second argument. Otherwise, it SHALL
 - 4364 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
 - 4365 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].

- 4366 • urn:oasis:names:tc:xacml:1.0:function:string-less-than
- 4367 This function SHALL take two arguments of data-type
 4368 “http://www.w3.org/2001/XMLSchema#string” and SHALL return an
 4369 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only the first
 4370 argument is lexicographically strictly less than the second argument. Otherwise, it SHALL return
 4371 “False”. The comparison SHALL use Unicode codepoint collation, as defined for the identifier
 4372 http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.
- 4373 • urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal
- 4374 This function SHALL take two arguments of data-type
 4375 “http://www.w3.org/2001/XMLSchema#string” and SHALL return an
 4376 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only the first
 4377 argument is lexicographically less than or equal to the second argument. Otherwise, it SHALL
 4378 return “False”. The comparison SHALL use Unicode codepoint collation, as defined for the
 4379 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.
- 4380 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than
- 4381 This function SHALL take two arguments of data-type
 4382 “http://www.w3.org/2001/XMLSchema#time” and SHALL return an
 4383 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 4384 argument is greater than the second argument according to the order relation specified for
 4385 “http://www.w3.org/2001/XMLSchema#time” **[XS]** Section 3.2.8. Otherwise, it SHALL return
 4386 “False”. Note: it is illegal to compare a time that includes a time-zone value with one that does
 4387 not. In such cases, the time-in-range function should be used.
- 4388 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal
- 4389 This function SHALL take two arguments of data-type
 4390 “http://www.w3.org/2001/XMLSchema#time” and SHALL return an
 4391 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 4392 argument is greater than or equal to the second argument according to the order relation
 4393 specified for “http://www.w3.org/2001/XMLSchema#time” **[XS]** Section 3.2.8. Otherwise, it
 4394 SHALL return “False”. Note: it is illegal to compare a time that includes a time-zone value with
 4395 one that does not. In such cases, the time-in-range function should be used.
- 4396 • urn:oasis:names:tc:xacml:1.0:function:time-less-than
- 4397 This function SHALL take two arguments of data-type
 4398 “http://www.w3.org/2001/XMLSchema#time” and SHALL return an
 4399 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 4400 argument is less than the second argument according to the order relation specified for
 4401 “http://www.w3.org/2001/XMLSchema#time” **[XS]** Section 3.2.8. Otherwise, it SHALL return
 4402 “False”. Note: it is illegal to compare a time that includes a time-zone value with one that does
 4403 not. In such cases, the time-in-range function should be used.
- 4404 • urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal
- 4405 This function SHALL take two arguments of data-type
 4406 “http://www.w3.org/2001/XMLSchema#time” and SHALL return an
 4407 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 4408 argument is less than or equal to the second argument according to the order relation specified
 4409 for “http://www.w3.org/2001/XMLSchema#time” **[XS]** Section 3.2.8. Otherwise, it SHALL return
 4410 “False”. Note: it is illegal to compare a time that includes a time-zone value with one that does
 4411 not. In such cases, the time-in-range function should be used.
- 4412 • urn:oasis:names:tc:xacml:2.0:function:time-in-range
- 4413 This function SHALL take three arguments of data-type
 4414 “http://www.w3.org/2001/XMLSchema#time” and SHALL return an
 4415 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if the first argument falls
 4416 in the range defined inclusively by the second and third arguments. Otherwise, it SHALL return

4417 “False”. Regardless of its value, the third argument SHALL be interpreted as a time that is equal
4418 to, or later than by less than twenty-four hours, the second argument. If no time zone is provided
4419 for the first argument, it SHALL use the default time zone at the **context handler**. If no time zone
4420 is provided for the second or third arguments, then they SHALL use the time zone from the first
4421 argument.

- 4422 • urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than

4423 This function SHALL take two arguments of data-type
4424 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an
4425 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4426 argument is greater than the second argument according to the order relation specified for
4427 “http://www.w3.org/2001/XMLSchema#dateTime” by [XS] part 2, section 3.2.7. Otherwise, it
4428 SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an
4429 implicit time-zone value SHALL be assigned, as described in [XS].

- 4430 • urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal

4431 This function SHALL take two arguments of data-type
4432 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an
4433 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4434 argument is greater than or equal to the second argument according to the order relation
4435 specified for “http://www.w3.org/2001/XMLSchema#dateTime” by [XS] part 2, section 3.2.7.
4436 Otherwise, it SHALL return “False”. Note: if a dateTime value does not include a time-zone
4437 value, then an implicit time-zone value SHALL be assigned, as described in [XS].

- 4438 • urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than

4439 This function SHALL take two arguments of data-type
4440 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an
4441 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4442 argument is less than the second argument according to the order relation specified for
4443 “http://www.w3.org/2001/XMLSchema#dateTime” by [XS, part 2, section 3.2.7]. Otherwise, it
4444 SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an
4445 implicit time-zone value SHALL be assigned, as described in [XS].

- 4446 • urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal

4447 This function SHALL take two arguments of data-type “http://www.w3.org/2001/XMLSchema#
4448 dateTime” and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL
4449 return “True” if and only if the first argument is less than or equal to the second argument
4450 according to the order relation specified for “http://www.w3.org/2001/XMLSchema#dateTime” by
4451 [XS] part 2, section 3.2.7. Otherwise, it SHALL return “False”. Note: if a dateTime value does
4452 not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described
4453 in [XS].

- 4454 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than

4455 This function SHALL take two arguments of data-type
4456 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
4457 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4458 argument is greater than the second argument according to the order relation specified for
4459 “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it SHALL
4460 return “False”. Note: if a date value does not include a time-zone value, then an implicit time-
4461 zone value SHALL be assigned, as described in [XS].

- 4462 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal

4463 This function SHALL take two arguments of data-type
4464 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
4465 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4466 argument is greater than or equal to the second argument according to the order relation
4467 specified for “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9.

- 4468 Otherwise, it SHALL return “False”. Note: if a date value does not include a time-zone value,
4469 then an implicit time-zone value SHALL be assigned, as described in [XS].
- 4470 • urn:oasis:names:tc:xacml:1.0:function:date-less-than
 - 4471 This function SHALL take two arguments of data-type
 - 4472 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
 - 4473 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 - 4474 argument is less than the second argument according to the order relation specified for
 - 4475 “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it SHALL
 - 4476 return “False”. Note: if a date value does not include a time-zone value, then an implicit time-
 - 4477 zone value SHALL be assigned, as described in [XS].
 - 4478 • urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal
 - 4479 This function SHALL take two arguments of data-type
 - 4480 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
 - 4481 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 - 4482 argument is less than or equal to the second argument according to the order relation specified
 - 4483 for “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it
 - 4484 SHALL return “False”. Note: if a date value does not include a time-zone value, then an implicit
 - 4485 time-zone value SHALL be assigned, as described in [XS].

4486 **A.3.9 String functions**

4487 The following functions operate on strings and convert to and from other data types.

- 4488 • urn:oasis:names:tc:xacml:2.0:function:string-concatenate
 - 4489 This function SHALL take two or more arguments of data-type
 - 4490 “http://www.w3.org/2001/XMLSchema#string” and SHALL return a
 - 4491 “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the concatenation, in order,
 - 4492 of the arguments.
- 4493 • urn:oasis:names:tc:xacml:3.0:function:boolean-from-string
 - 4494 This function SHALL take one argument of data-type
 - 4495 “http://www.w3.org/2001/XMLSchema#string”, and SHALL return an
 - 4496 “http://www.w3.org/2001/XMLSchema#boolean”. The result SHALL be the string converted to a
 - 4497 boolean. If the argument is not a valid lexical representation of a boolean, then the result SHALL
 - 4498 be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4499 • urn:oasis:names:tc:xacml:3.0:function:string-from-boolean
 - 4500 This function SHALL take one argument of data-type
 - 4501 “http://www.w3.org/2001/XMLSchema#boolean”, and SHALL return an
 - 4502 “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the boolean converted to a
 - 4503 string in the canonical form specified in [XS].
- 4504 • urn:oasis:names:tc:xacml:3.0:function:integer-from-string
 - 4505 This function SHALL take one argument of data-type
 - 4506 “http://www.w3.org/2001/XMLSchema#string”, and SHALL return an
 - 4507 “http://www.w3.org/2001/XMLSchema#integer”. The result SHALL be the string converted to an
 - 4508 integer. If the argument is not a valid lexical representation of an integer, then the result SHALL
 - 4509 be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4510 • urn:oasis:names:tc:xacml:3.0:function:string-from-integer
 - 4511 This function SHALL take one argument of data-type
 - 4512 “http://www.w3.org/2001/XMLSchema#integer”, and SHALL return an
 - 4513 “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the integer converted to a
 - 4514 string in the canonical form specified in [XS].
- 4515 • urn:oasis:names:tc:xacml:3.0:function:double-from-string

- 4516 This function SHALL take one argument of data-type
4517 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
4518 "http://www.w3.org/2001/XMLSchema#double". The result SHALL be the string converted to a
4519 double. If the argument is not a valid lexical representation of a double, then the result SHALL be
4520 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4521 • urn:oasis:names:tc:xacml:3.0:function:string-from-double
4522 This function SHALL take one argument of data-type
4523 "http://www.w3.org/2001/XMLSchema#double", and SHALL return an
4524 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the double converted to a
4525 string in the canonical form specified in **[XS]**.
 - 4526 • urn:oasis:names:tc:xacml:3.0:function:time-from-string
4527 This function SHALL take one argument of data-type
4528 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
4529 "http://www.w3.org/2001/XMLSchema#time". The result SHALL be the string converted to a time.
4530 If the argument is not a valid lexical representation of a time, then the result SHALL be
4531 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4532 • urn:oasis:names:tc:xacml:3.0:function:string-from-time
4533 This function SHALL take one argument of data-type
4534 "http://www.w3.org/2001/XMLSchema#time", and SHALL return an
4535 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the time converted to a
4536 string in the canonical form specified in **[XS]**.
 - 4537 • urn:oasis:names:tc:xacml:3.0:function:date-from-string
4538 This function SHALL take one argument of data-type
4539 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
4540 "http://www.w3.org/2001/XMLSchema#date". The result SHALL be the string converted to a
4541 date. If the argument is not a valid lexical representation of a date, then the result SHALL be
4542 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4543 • urn:oasis:names:tc:xacml:3.0:function:string-from-date
4544 This function SHALL take one argument of data-type
4545 "http://www.w3.org/2001/XMLSchema#date", and SHALL return an
4546 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the date converted to a
4547 string in the canonical form specified in **[XS]**.
 - 4548 • urn:oasis:names:tc:xacml:3.0:function:dateTime-from-string
4549 This function SHALL take one argument of data-type
4550 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
4551 "http://www.w3.org/2001/XMLSchema#dateTime". The result SHALL be the string converted to a
4552 dateTime. If the argument is not a valid lexical representation of a dateTime, then the result
4553 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4554 urn:oasis:names:tc:xacml:3.0:function:string-from-dateTime
4555 This function SHALL take one argument of data-type
4556 "http://www.w3.org/2001/XMLSchema#dateTime", and SHALL return an
4557 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the dateTime converted to a
4558 string in the canonical form specified in **[XS]**.
 - 4559 • urn:oasis:names:tc:xacml:3.0:function:anyURI-from-string
4560 This function SHALL take one argument of data-type
4561 "http://www.w3.org/2001/XMLSchema#string", and SHALL return a
4562 "http://www.w3.org/2001/XMLSchema#anyURI". The result SHALL be the URI constructed by
4563 converting the argument to an URI. If the argument is not a valid lexical representation of a URI,
4564 then the result SHALL be Indeterminate with status code
4565 urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- 4566 • urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI
 4567 This function SHALL take one argument of data-type
 4568 "http://www.w3.org/2001/XMLSchema#anyURI", and SHALL return an
 4569 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the URI converted to a
 4570 string in the form it was originally represented in XML form.
- 4571 • urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-from-string
 4572 This function SHALL take one argument of data-type
 4573 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4574 "http://www.w3.org/2001/XMLSchema#dayTimeDuration ". The result SHALL be the string
 4575 converted to a dayTimeDuration. If the argument is not a valid lexical representation of a
 4576 dayTimeDuration, then the result SHALL be Indeterminate with status code
 4577 urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4578 • urn:oasis:names:tc:xacml:3.0:function:string-from-dayTimeDuration
 4579 This function SHALL take one argument of data-type
 4580 "http://www.w3.org/2001/XMLSchema#dayTimeDuration ", and SHALL return an
 4581 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the dayTimeDuration
 4582 converted to a string in the canonical form specified in **[XPathFunc]**.
- 4583 • urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-from-string
 4584 This function SHALL take one argument of data-type
 4585 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4586 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". The result SHALL be the string
 4587 converted to a yearMonthDuration. If the argument is not a valid lexical representation of a
 4588 yearMonthDuration, then the result SHALL be Indeterminate with status code
 4589 urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4590 • urn:oasis:names:tc:xacml:3.0:function:string-from-yearMonthDuration
 4591 This function SHALL take one argument of data-type
 4592 "http://www.w3.org/2001/XMLSchema#yearMonthDuration", and SHALL return an
 4593 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the yearMonthDuration
 4594 converted to a string in the canonical form specified in **[XPathFunc]**.
- 4595 • urn:oasis:names:tc:xacml:3.0:function:x500Name-from-string
 4596 This function SHALL take one argument of data-type
 4597 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4598 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". The result SHALL be the string converted
 4599 to an x500Name. If the argument is not a valid lexical representation of a X500Name, then the
 4600 result SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4601 • urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name
 4602 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-
 4603 type:x500Name", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
 4604 SHALL be the x500Name converted to a string in the form it was originally represented in XML
 4605 form..
- 4606 • urn:oasis:names:tc:xacml:3.0:function:rfc822Name-from-string
 4607 This function SHALL take one argument of data-type
 4608 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4609 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". The result SHALL be the string converted
 4610 to an rfc822Name. If the argument is not a valid lexical representation of an rfc822Name, then the
 4611 result SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4612 • urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name
 4613 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-
 4614 type:rfc822Name", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The

- 4615 result SHALL be the rfc822Name converted to a string in the form it was originally represented in
4616 XML form.
- 4617 • urn:oasis:names:tc:xacml:3.0:function:ipAddress-from-string
4618 This function SHALL take one argument of data-type
4619 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
4620 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". The result SHALL be the string converted to
4621 an ipAddress. If the argument is not a valid lexical representation of an ipAddress, then the result
4622 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4623 • urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress
4624 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-
4625 type:ipAddress", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
4626 SHALL be the ipAddress converted to a string in the form it was originally represented in XML
4627 form.
 - 4628 • urn:oasis:names:tc:xacml:3.0:function:dnsName-from-string
4629 This function SHALL take one argument of data-type
4630 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
4631 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". The result SHALL be the string converted to
4632 a dnsName. If the argument is not a valid lexical representation of a dnsName, then the result
4633 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4634 • urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName
4635 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-
4636 type:dnsName", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
4637 SHALL be the dnsName converted to a string in the form it was originally represented in XML
4638 form.
 - 4639 • urn:oasis:names:tc:xacml:3.0:function:string-starts-with
4640 This function SHALL take two arguments of data-type
4641 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
4642 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
4643 begins with the first string, and false otherwise. Equality testing SHALL be done as defined for
4644 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4645 • urn:oasis:names:tc:xacml:3.0:function:anyURI-starts-with
4646 This function SHALL take a first argument of data-
4647 type"http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
4648 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
4649 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
4650 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI begins with the string,
4651 and false otherwise. Equality testing SHALL be done as defined for
4652 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4653 • urn:oasis:names:tc:xacml:3.0:function:string-ends-with
4654 This function SHALL take two arguments of data-type
4655 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
4656 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
4657 ends with the first string, and false otherwise. Equality testing SHALL be done as defined for
4658 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4659 • urn:oasis:names:tc:xacml:3.0:function:anyURI-ends-with
4660 This function SHALL take a first argument of data-type
4661 "http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
4662 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
4663 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
4664 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI ends with the string,

- 4665 and false otherwise. Equality testing SHALL be done as defined for
 4666 urn:oasis:names:tc:xacml:1.0:function:string-equal.
- 4667 • urn:oasis:names:tc:xacml:3.0:function:string-contains
 4668 This function SHALL take two arguments of data-type
 4669 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
 4670 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
 4671 contains the first string, and false otherwise. Equality testing SHALL be done as defined for
 4672 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4673 • urn:oasis:names:tc:xacml:3.0:function:anyURI-contains
 4674 This function SHALL take a first argument of data-type
 4675 "http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
 4676 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
 4677 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
 4678 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI contains the string, and
 4679 false otherwise. Equality testing SHALL be done as defined for
 4680 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4681 • urn:oasis:names:tc:xacml:3.0:function:string-substring
 4682 This function SHALL take a first argument of data-type
 4683 "http://www.w3.org/2001/XMLSchema#string" and a second and a third argument of type
 4684 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return a
 4685 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the substring of the first
 4686 argument beginning at the position given by the second argument and ending at the position
 4687 before the position given by the third argument. The first character of the string has position zero.
 4688 The negative integer value -1 given for the third arguments indicates the end of the string. If the
 4689 second or third arguments are out of bounds, then the function MUST evaluate to Indeterminate
 4690 with a status code of urn:oasis:names:tc:xacml:1.0:status:processing-error.
 - 4691 • urn:oasis:names:tc:xacml:3.0:function:anyURI-substring
 4692 This function SHALL take a first argument of data-type
 4693 "http://www.w3.org/2001/XMLSchema#anyURI" and a second and a third argument of type
 4694 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return a
 4695 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the substring of the first
 4696 argument converted to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI
 4697 beginning at the position given by the second argument and ending at the position before the
 4698 position given by the third argument. The first character of the URI converted to a string has
 4699 position zero. The negative integer value -1 given for the third arguments indicates the end of the
 4700 string. If the second or third arguments are out of bounds, then the function MUST evaluate to
 4701 Indeterminate with a status code of
 4702 urn:oasis:names:tc:xacml:1.0:status:processing-error. If the resulting substring
 4703 is not syntactically a valid URI, then the function MUST evaluate to Indeterminate with a status
 4704 code of urn:oasis:names:tc:xacml:1.0:status:processing-error.

4705

4706 A.3.10 Bag functions

4707 These functions operate on a **bag** of 'type' values, where type is one of the primitive data-types, and x.x
 4708 is a version of XACML where the function has been defined. Some additional conditions defined for
 4709 each function below SHALL cause the expression to evaluate to "Indeterminate".

- 4710 • urn:oasis:names:tc:xacml:x.x:function:type-one-and-only
 4711 This function SHALL take a **bag** of 'type' values as an argument and SHALL return a value of
 4712 'type'. It SHALL return the only value in the **bag**. If the **bag** does not have one and only one
 4713 value, then the expression SHALL evaluate to "Indeterminate".

- 4714 • urn:oasis:names:tc:xacml:x.x:function:type-bag-size
- 4715 This function SHALL take a **bag** of 'type' values as an argument and SHALL return an
- 4716 "http://www.w3.org/2001/XMLSchema#integer" indicating the number of values in the **bag**.
- 4717 • urn:oasis:names:tc:xacml:x.x:function:type-is-in
- 4718 This function SHALL take an argument of 'type' as the first argument and a **bag** of 'type' values
- 4719 as the second argument and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean".
- 4720 The function SHALL evaluate to "True" if and only if the first argument matches by the
- 4721 "urn:oasis:names:tc:xacml:x.x:function:type-equal" any value in the **bag**. Otherwise, it SHALL
- 4722 return "False".
- 4723 • urn:oasis:names:tc:xacml:x.x:function:type-bag
- 4724 This function SHALL take any number of arguments of 'type' and return a **bag** of 'type' values
- 4725 containing the values of the arguments. An application of this function to zero arguments SHALL
- 4726 produce an empty **bag** of the specified data-type.

4727 A.3.11 Set functions

4728 These functions operate on **bags** mimicking sets by eliminating duplicate elements from a **bag**.

- 4729 • urn:oasis:names:tc:xacml:x.x:function:type-intersection
- 4730 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4731 **bag** of 'type' values such that it contains only elements that are common between the two **bags**,
- 4732 which is determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal". No duplicates, as
- 4733 determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal", SHALL exist in the result.
- 4734 • urn:oasis:names:tc:xacml:x.x:function:type-at-least-one-member-of
- 4735 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4736 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL evaluate to "True" if and
- 4737 only if at least one element of the first argument is contained in the second argument as
- 4738 determined by "urn:oasis:names:tc:xacml:x.x:function:type-is-in".
- 4739 • urn:oasis:names:tc:xacml:x.x:function:type-union
- 4740 This function SHALL take two or more arguments that are both a **bag** of 'type' values. The
- 4741 expression SHALL return a **bag** of 'type' such that it contains all elements of all the argument
- 4742 **bags**. No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",
- 4743 SHALL exist in the result.
- 4744 • urn:oasis:names:tc:xacml:x.x:function:type-subset
- 4745 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4746 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
- 4747 argument is a subset of the second argument. Each argument SHALL be considered to have had
- 4748 its duplicates removed, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",
- 4749 before the subset calculation.
- 4750 • urn:oasis:names:tc:xacml:x.x:function:type-set-equals
- 4751 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4752 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return the result of applying
- 4753 "urn:oasis:names:tc:xacml:1.0:function:and" to the application of
- 4754 "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the first and second arguments and the
- 4755 application of "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the second and first
- 4756 arguments.

4757 A.3.12 Higher-order bag functions

4758 This section describes functions in XACML that perform operations on **bags** such that functions may be

4759 applied to the **bags** in general.

4760 • urn:oasis:names:tc:xacml:3.0:function:any-of

4761 This function applies a Boolean function between specific primitive values and a **bag** of values,
4762 and SHALL return "True" if and only if the **predicate** is "True" for at least one element of the **bag**.

4763 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4764 be an <Function> element that names a Boolean function that takes n arguments of primitive
4765 types. Under the remaining n arguments, n-1 parameters SHALL be values of primitive data-
4766 types and one SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as
4767 if the function named in the <Function> argument were applied to the n-1 non-bag arguments
4768 and each element of the bag argument and the results are combined with
4769 "urn:oasis:names:tc:xacml:1.0:function:or".

4770 For example, the following expression SHALL return "True":

```
4771 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">  
4772   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>  
4773   <AttributeValue  
4774     DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4775   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4776     <AttributeValue  
4777       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>  
4778     <AttributeValue  
4779       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4780     <AttributeValue  
4781       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>  
4782     <AttributeValue  
4783       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4784   </Apply>  
4785 </Apply>
```

4786 This expression is "True" because the first argument is equal to at least one of the elements of
4787 the **bag**, according to the function.

4788 • urn:oasis:names:tc:xacml:3.0:function:all-of

4789 This function applies a Boolean function between a specific primitive value and a **bag** of values,
4790 and returns "True" if and only if the **predicate** is "True" for every element of the **bag**.

4791 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4792 be a <Function> element that names a Boolean function that takes n arguments of primitive
4793 types. Under the remaining n arguments, n-1 parameters SHALL be values of primitive data-
4794 types and one SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as
4795 if the function named in the <Function> argument were applied to the n-1 non-bag arguments
4796 and each element of the bag argument and the results are combined with
4797 "urn:oasis:names:tc:xacml:1.0:function:and".

4798 For example, the following expression SHALL evaluate to "True":

```
4799 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:all-of">  
4800   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4801     greater-than"/>  
4802   <AttributeValue  
4803     DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>  
4804   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4805     <AttributeValue  
4806       DataType="http://www.w3.org/2001/XMLSchema#integer">9</AttributeValue>  
4807     <AttributeValue  
4808       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>  
4809     <AttributeValue  
4810       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>  
4811     <AttributeValue  
4812       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>  
4813   </Apply>  
4814 </Apply>
```

4815 This expression is "True" because the first argument (10) is greater than all of the elements of the
4816 **bag** (9,3,4 and 2).

4817 • urn:oasis:names:tc:xacml:3.0:function:any-of-any

4818 This function applies a Boolean function on each tuple from the cross product on all bags
4819 arguments, and returns "True" if and only if the **predicate** is "True" for at least one inside-function
4820 call.

4821 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4822 be an <Function> element that names a Boolean function that takes n arguments. The
4823 remaining arguments are either primitive data types or bags of primitive types. The expression
4824 SHALL be evaluated as if the function named in the <Function> argument was applied between
4825 every tuple of the cross product on all bags and the primitive values, and the results were
4826 combined using "urn:oasis:names:tc:xacml:1.0:function:or". The semantics are that the result of
4827 the expression SHALL be "True" if and only if the applied **predicate** is "True" for at least one
4828 function call on the tuples from the **bags** and primitive values.

4829 For example, the following expression SHALL evaluate to "True":

```
4830 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of-any">  
4831   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>  
4832   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4833     <AttributeValue  
4834     DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4835     <AttributeValue  
4836     DataType="http://www.w3.org/2001/XMLSchema#string">Mary</AttributeValue>  
4837   </Apply>  
4838   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4839     <AttributeValue  
4840     DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>  
4841     <AttributeValue  
4842     DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4843     <AttributeValue  
4844     DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>  
4845     <AttributeValue  
4846     DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4847   </Apply>  
4848 </Apply>
```

4849 This expression is "True" because at least one of the elements of the first **bag**, namely "Ringo", is
4850 equal to at least one of the elements of the second **bag**.

4851 • urn:oasis:names:tc:xacml:1.0:function:all-of-any

4852 This function applies a Boolean function between the elements of two **bags**. The expression
4853 SHALL be "True" if and only if the supplied **predicate** is "True" between each element of the first
4854 **bag** and any element of the second **bag**.

4855 This function SHALL take three arguments. The first argument SHALL be an <Function>
4856 element that names a Boolean function that takes two arguments of primitive types. The second
4857 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a
4858 primitive data-type. The expression SHALL be evaluated as if the
4859 "urn:oasis:names:tc:xacml:3.0:function:any-of" function had been applied to each value of the first
4860 **bag** and the whole of the second **bag** using the supplied xacml:Function, and the results were
4861 then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4862 For example, the following expression SHALL evaluate to "True":

```
4863 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-any">  
4864   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4865   greater-than"/>  
4866   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4867     <AttributeValue  
4868     DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
```

```

4869         <AttributeValue
4870         DataType="http://www.w3.org/2001/XMLSchema#integer">20</AttributeValue>
4871     </Apply>
4872     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4873         <AttributeValue
4874         DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4875         <AttributeValue
4876         DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4877         <AttributeValue
4878         DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4879         <AttributeValue
4880         DataType="http://www.w3.org/2001/XMLSchema#integer">19</AttributeValue>
4881     </Apply>
4882 </Apply>

```

4883 This expression is "True" because each of the elements of the first **bag** is greater than at least
4884 one of the elements of the second **bag**.

4885 • urn:oasis:names:tc:xacml:1.0:function:any-of-all

4886 This function applies a Boolean function between the elements of two **bags**. The expression
4887 SHALL be "True" if and only if the supplied **predicate** is "True" between each element of the
4888 second **bag** and any element of the first **bag**.

4889 This function SHALL take three arguments. The first argument SHALL be an <Function>
4890 element that names a Boolean function that takes two arguments of primitive types. The second
4891 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a
4892 primitive data-type. The expression SHALL be evaluated as if the
4893 "urn:oasis:names:tc:xacml:3.0:function:any-of" function had been applied to each value of the
4894 second **bag** and the whole of the first **bag** using the supplied xacml:Function, and the results
4895 were then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4896 For example, the following expression SHALL evaluate to "True":

```

4897 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-all">
4898     <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
4899     greater-than"/>
4900     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4901         <AttributeValue
4902         DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4903         <AttributeValue
4904         DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4905     </Apply>
4906     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4907         <AttributeValue
4908         DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4909         <AttributeValue
4910         DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4911         <AttributeValue
4912         DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4913         <AttributeValue
4914         DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4915     </Apply>
4916 </Apply>

```

4917 This expression is "True" because, for all of the values in the second **bag**, there is a value in the
4918 first **bag** that is greater.

4919 • urn:oasis:names:tc:xacml:1.0:function:all-of-all

4920 This function applies a Boolean function between the elements of two **bags**. The expression
4921 SHALL be "True" if and only if the supplied **predicate** is "True" between each and every element
4922 of the first **bag** collectively against all the elements of the second **bag**.

4923 This function SHALL take three arguments. The first argument SHALL be an <Function>
4924 element that names a Boolean function that takes two arguments of primitive types. The second

4925 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a
4926 primitive data-type. The expression is evaluated as if the function named in the <Function>
4927 element were applied between every element of the second argument and every element of the
4928 third argument and the results were combined using "urn:oasis:names:tc:xacml:1.0:function:and".
4929 The semantics are that the result of the expression is "True" if and only if the applied **predicate** is
4930 "True" for all elements of the first **bag** compared to all the elements of the second **bag**.

4931 For example, the following expression SHALL evaluate to "True":

```
4932 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-all">  
4933   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4934   greater-than"/>  
4935   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4936     <AttributeValue  
4937     DataType="http://www.w3.org/2001/XMLSchema#integer">6</AttributeValue>  
4938     <AttributeValue  
4939     DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>  
4940   </Apply>  
4941   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4942     <AttributeValue  
4943     DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>  
4944     <AttributeValue  
4945     DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>  
4946     <AttributeValue  
4947     DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>  
4948     <AttributeValue  
4949     DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>  
4950   </Apply>  
4951 </Apply>
```

4952 This expression is "True" because all elements of the first **bag**, "5" and "6", are each greater than
4953 all of the integer values "1", "2", "3", "4" of the second **bag**.

4954 • urn:oasis:names:tc:xacml:3.0:function:map

4955 This function converts a **bag** of values to another **bag** of values.

4956 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4957 be a <Function> element naming a function that takes a n arguments of a primitive data-type
4958 and returns a value of a primitive data-type Under the remaining n arguments, n-1 parameters
4959 SHALL be values of primitive data-types and one SHALL be a **bag** of a primitive data-type. The
4960 expression SHALL be evaluated as if the function named in the <Function> argument were
4961 applied to the n-1 non-bag arguments and each element of the bag argument and resulting in a
4962 **bag** of the converted value. The result SHALL be a **bag** of the primitive data-type that is returned
4963 by the function named in the <xacml:Function> element.

4964 For example, the following expression,

```
4965 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:map">  
4966   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-  
4967   normalize-to-lower-case">  
4968   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4969     <AttributeValue  
4970     DataType="http://www.w3.org/2001/XMLSchema#string">Hello</AttributeValue>  
4971     <AttributeValue  
4972     DataType="http://www.w3.org/2001/XMLSchema#string">World!</AttributeValue>  
4973   </Apply>  
4974 </Apply>
```

4975 evaluates to a **bag** containing "hello" and "world!".

4976 A.3.13 Regular-expression-based functions

4977 These functions operate on various types using regular expressions and evaluate to
4978 "http://www.w3.org/2001/XMLSchema#boolean".

- 4979 • urn:oasis:names:tc:xacml:1.0:function:string-regexp-match
- 4980 This function decides a regular expression match. It SHALL take two arguments of
 4981 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
 4982 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
 4983 expression and the second argument SHALL be a general string. The function specification
 4984 SHALL be that of the "xf:matches" function with the arguments reversed [XF] Section 7.6.2.
- 4985 • urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match
- 4986 This function decides a regular expression match. It SHALL take two arguments; the first is of
 4987 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
 4988 "http://www.w3.org/2001/XMLSchema#anyURI". It SHALL return an
 4989 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
 4990 expression and the second argument SHALL be a URI. The function SHALL convert the second
 4991 argument to type "http://www.w3.org/2001/XMLSchema#string" with
 4992 urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI, then apply
 4993 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 4994 • urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match
- 4995 This function decides a regular expression match. It SHALL take two arguments; the first is of
 4996 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
 4997 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". It SHALL return an
 4998 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
 4999 expression and the second argument SHALL be an IPv4 or IPv6 address. The function SHALL
 5000 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
 5001 urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress, then apply
 5002 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5003 • urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match
- 5004 This function decides a regular expression match. It SHALL take two arguments; the first is of
 5005 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
 5006 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". It SHALL return an
 5007 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
 5008 expression and the second argument SHALL be a DNS name. The function SHALL convert the
 5009 second argument to type "http://www.w3.org/2001/XMLSchema#string" with
 5010 urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName, then apply
 5011 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5012 • urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match
- 5013 This function decides a regular expression match. It SHALL take two arguments; the first is of
 5014 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
 5015 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". It SHALL return an
 5016 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
 5017 expression and the second argument SHALL be an RFC 822 name. The function SHALL convert
 5018 the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
 5019 urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name, then apply
 5020 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5021 • urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match
- 5022 This function decides a regular expression match. It SHALL take two arguments; the first is of
 5023 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
 5024 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". It SHALL return an
 5025 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
 5026 expression and the second argument SHALL be an X.500 directory name. The function SHALL
 5027 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
 5028 urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name, then apply
 5029 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

5030 **A.3.14 Special match functions**

5031 These functions operate on various types and evaluate to
5032 "http://www.w3.org/2001/XMLSchema#boolean" based on the specified standard matching algorithm.

- 5033 • urn:oasis:names:tc:xacml:1.0:function:x500Name-match

5034 This function shall take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name"
5035 and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and
5036 only if the first argument matches some terminal sequence of RDNs from the second argument
5037 when compared using x500Name-equal.

5038 As an example (non-normative), if the first argument is "O=Medico Corp,C=US" and the second
5039 argument is "cn=John Smith,o=Medico Corp,c=US", then the function will return "True".

- 5040 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match

5041 This function SHALL take two arguments, the first is of data-type
5042 "http://www.w3.org/2001/XMLSchema#string" and the second is of data-type
5043 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and SHALL return an
5044 "http://www.w3.org/2001/XMLSchema#boolean". This function SHALL evaluate to "True" if the
5045 first argument matches the second argument according to the following specification.

5046 An RFC822 name consists of a local-part followed by "@" followed by a domain-part. The local-
5047 part is case-sensitive, while the domain-part (which is usually a DNS name) is not case-sensitive.

5048 The second argument contains a complete rfc822Name. The first argument is a complete or
5049 partial rfc822Name used to select appropriate values in the second argument as follows.

5050 In order to match a particular address in the second argument, the first argument must specify the
5051 complete mail address to be matched. For example, if the first argument is
5052 "Anderson@sun.com", this matches a value in the second argument of "Anderson@sun.com"
5053 and "Anderson@SUN.COM", but not "Anne.Anderson@sun.com", "anderson@sun.com" or
5054 "Anderson@east.sun.com".

5055 In order to match any address at a particular domain in the second argument, the first argument
5056 must specify only a domain name (usually a DNS name). For example, if the first argument is
5057 "sun.com", this matches a value in the second argument of "Anderson@sun.com" or
5058 "Baxter@SUN.COM", but not "Anderson@east.sun.com".

5059 In order to match any address in a particular domain in the second argument, the first argument
5060 must specify the desired domain-part with a leading ".". For example, if the first argument is
5061 ".east.sun.com", this matches a value in the second argument of "Anderson@east.sun.com" and
5062 "anne.anderson@ISRG.EAST.SUN.COM" but not "Anderson@sun.com".

5063 **A.3.15 XPath-based functions**

5064 This section specifies functions that take XPath expressions for arguments. An XPath expression
5065 evaluates to a node-set, which is a set of XML nodes that match the expression. A node or node-set is
5066 not in the formal data-type system of XACML. All comparison or other operations on node-sets are
5067 performed in isolation of the particular function specified. The context nodes and namespace mappings
5068 of the XPath expressions are defined by the XPath data-type, see section B.3. The following functions
5069 are defined:

- 5070 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-count

5071 This function SHALL take an "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" as an
5072 argument and evaluates to an "http://www.w3.org/2001/XMLSchema#integer". The value
5073 returned from the function SHALL be the count of the nodes within the node-set that match the
5074 given XPath expression. If the <Content> element of the category to which the XPath
5075 expression applies to is not present in the request, this function SHALL return a value of zero.

- 5076 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal

5077 This function SHALL take two “urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression”
5078 arguments and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. The function
5079 SHALL return "True" if any of the XML nodes in the node-set matched by the first argument
5080 equals any of the XML nodes in the node-set matched by the second argument. Two nodes are
5081 considered equal if they have the same identity. If the <Content> element of the category to
5082 which either XPath expression applies to is not present in the request, this function SHALL return
5083 a value of “False”.

5084 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

5085 This function SHALL take two “urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression”
5086 arguments and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. This function
5087 SHALL evaluate to "True" if one of the following two conditions is satisfied: (1) Any of the XML
5088 nodes in the node-set matched by the first argument is equal to any of the XML nodes in the
5089 node-set matched by the second argument; (2) any node below any of the XML nodes in the
5090 node-set matched by the first argument is equal to any of the XML nodes in the node-set
5091 matched by the second argument. Two nodes are considered equal if they have the same
5092 identity. If the <Content> element of the category to which either XPath expression applies to is
5093 not present in the request, this function SHALL return a value of “False”.

5094 NOTE: The first **condition** is equivalent to "xpath-node-equal", and guarantees that "xpath-node-equal" is
5095 a special case of "xpath-node-match".

5096 A.3.16 Other functions

5097 • urn:oasis:names:tc:xacml:3.0:function:access-permitted

5098 This function SHALL take an “http://www.w3.org/2001/XMLSchema#anyURI” and an
5099 “http://www.w3.org/2001/XMLSchema#string” as arguments. The first argument SHALL be
5100 interpreted as an **attribute** category. The second argument SHALL be interpreted as the XML
5101 content of an <Attributes> element with *Category* equal to the first argument. The function
5102 evaluates to an “http://www.w3.org/2001/XMLSchema#boolean”. This function SHALL return
5103 "True" if and only if the **policy** evaluation described below returns the value of "Permit".

5104 The following evaluation is described as if the **context** is actually instantiated, but it is only
5105 required that an equivalent result be obtained.

5106 The function SHALL construct a new **context**, by copying all the information from the current
5107 **context**, omitting any <Attributes> element with *Category* equal to the first argument. The
5108 second function argument SHALL be added to the **context** as the content of an <Attributes>
5109 element with *Category* equal to the first argument.

5110 The function SHALL invoke a complete **policy** evaluation using the newly constructed **context**.
5111 This evaluation SHALL be completely isolated from the evaluation which invoked the function, but
5112 shall use all current **policies** and combining algorithms, including any per request **policies**.

5113 The **PDP** SHALL detect any loop which may occur if successive evaluations invoke this function
5114 by counting the number of total invocations of any instance of this function during any single initial
5115 invocation of the **PDP**. If the total number of invocations exceeds the bound for such invocations,
5116 the initial invocation of this function evaluates to Indeterminate with a
5117 “urn:oasis:names:tc:xacml:1.0:status:processing-error” status code. Also, see the security
5118 considerations in section 9.1.8.

5119 A.3.17 Extension functions and primitive types

5120 Functions and primitive types are specified by string identifiers allowing for the introduction of functions in
5121 addition to those specified by XACML. This approach allows one to extend the XACML module with
5122 special functions and special primitive data-types.

5123 In order to preserve the integrity of the XACML evaluation strategy, the result of an extension function
5124 SHALL depend only on the values of its arguments. Global and hidden parameters SHALL NOT affect

5125 the evaluation of an expression. Functions SHALL NOT have side effects, as evaluation order cannot be
5126 guaranteed in a standard way.

5127 **A.4 Functions, data types, attributes and algorithms planned for** 5128 **deprecation**

5129 The following functions, data types and algorithms have been defined by previous versions of XACML
5130 and newer and better alternatives are defined in XACML 3.0. Their use is discouraged for new use and
5131 they are candidates for deprecation in future versions of XACML.

5132 The following xpath based functions have been replaced with equivalent functions which use the new
5133 urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression datatype instead of strings.

- 5134 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-count
- 5135 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-count
- 5136 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal
- 5137 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal
- 5138 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-match
- 5139 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

5140 The following URI and string concatenation function has been replaced with a string to URI conversion
5141 function, which allows the use of the general string functions with URI through string conversion.

- 5142 • urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate
- 5143 • Replaced by urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI

5144 The following identifiers have been replaced with official identifiers defined by W3C.

- 5145 • <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration>
- 5146 • Replaced with <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 5147 • <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration>
- 5148 • Replaced with <http://www.w3.org/2001/XMLSchema#yearMonthDuration>

5149 The following functions have been replaced with functions which use the updated dayTimeDuration and
5150 yearMonthDuration data types.

- 5151 • urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal
- 5152 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal
- 5153 • urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal
- 5154 • Replaced with urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal
- 5155 • urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration
- 5156 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration
- 5157 • urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration
- 5158 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration
- 5159 • urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration
- 5160 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
- 5161 • urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration
- 5162 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
- 5163 • urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration
- 5164 • Replaced with urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
- 5165 • urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration
- 5166 • Replaced with urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration

- 5167 The following attribute identifiers have been replaced with new identifiers
- 5168 • urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address
 - 5169 • Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-
5170 address
 - 5171 • urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name
 - 5172 • Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-
5173 locality:dns-name
 - 5174
- 5175 The following combining algorithms have been replaced with new variants which allow for better handling
5176 of “Indeterminate” results.
- 5177 • urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides
 - 5178 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides
 - 5179 • urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides
 - 5180 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides
 - 5181 • urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides
 - 5182 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides
 - 5183 • urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides
 - 5184 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides
 - 5185 • urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides
 - 5186 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides
 - 5187 • urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides
 - 5188 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides
 - 5189 • urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides
 - 5190 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-overrides
 - 5191 • urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides
 - 5192 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-overrides

5193 Appendix B. XACML identifiers (normative)

5194 This section defines standard identifiers for commonly used entities.

5195 B.1 XACML namespaces

5196 XACML is defined using this identifier.

5197 `urn:oasis:names:tc:xacml:3.0:core:schema`

5198 B.2 Attribute categories

5199 The following **attribute** category identifiers MUST be used when an XACML 2.0 or earlier **policy** or
5200 request is translated into XACML 3.0.

5201 **Attributes** previously placed in the **Resource**, **Action**, and **Environment** sections of a request are
5202 placed in an **attribute** category with the following identifiers respectively. It is RECOMMENDED that they
5203 are used to list **attributes of resources**, **actions**, and the **environment** respectively when authoring
5204 XACML 3.0 **policies** or requests.

5205 `urn:oasis:names:tc:xacml:3.0:attribute-category:resource`

5206 `urn:oasis:names:tc:xacml:3.0:attribute-category:action`

5207 `urn:oasis:names:tc:xacml:3.0:attribute-category:environment`

5208 **Attributes** previously placed in the **Subject** section of a request are placed in an **attribute** category
5209 which is identical of the **subject** category in XACML 2.0, as defined below. It is RECOMMENDED that
5210 they are used to list **attributes of subjects** when authoring XACML 3.0 **policies** or requests.

5211 This identifier indicates the system entity that initiated the **access** request. That is, the initial entity in a
5212 request chain. If **subject** category is not specified in XACML 2.0, this is the default translation value.

5213 `urn:oasis:names:tc:xacml:1.0:subject-category:access-subject`

5214 This identifier indicates the system entity that will receive the results of the request (used when it is
5215 distinct from the access-**subject**).

5216 `urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject`

5217 This identifier indicates a system entity through which the **access** request was passed.

5218 `urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject`

5219 This identifier indicates a system entity associated with a local or remote codebase that generated the
5220 request. Corresponding **subject attributes** might include the URL from which it was loaded and/or the
5221 identity of the code-signer.

5222 `urn:oasis:names:tc:xacml:1.0:subject-category:codebase`

5223 This identifier indicates a system entity associated with the computer that initiated the **access** request.
5224 An example would be an IPsec identity.

5225 `urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine`

5226 B.3 Data-types

5227 The following identifiers indicate data-types that are defined in Section A.2.

5228 `urn:oasis:names:tc:xacml:1.0:data-type:x500Name`.

5229 `urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name`

5230 `urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`

5231 `urn:oasis:names:tc:xacml:2.0:data-type:dnsName`

5232 `urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression`

- 5233 The following data-type identifiers are defined by XML Schema [XS].
- 5234 <http://www.w3.org/2001/XMLSchema#string>
- 5235 <http://www.w3.org/2001/XMLSchema#boolean>
- 5236 <http://www.w3.org/2001/XMLSchema#integer>
- 5237 <http://www.w3.org/2001/XMLSchema#double>
- 5238 <http://www.w3.org/2001/XMLSchema#time>
- 5239 <http://www.w3.org/2001/XMLSchema#date>
- 5240 <http://www.w3.org/2001/XMLSchema#dateTime>
- 5241 <http://www.w3.org/2001/XMLSchema#anyURI>
- 5242 <http://www.w3.org/2001/XMLSchema#hexBinary>
- 5243 <http://www.w3.org/2001/XMLSchema#base64Binary>
- 5244 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.
- 5245 <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 5246 <http://www.w3.org/2001/XMLSchema#yearMonthDuration>

5248 B.4 Subject attributes

- 5249 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).
- 5250
- 5251 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.
- 5252
- 5253 This identifier indicates the name of the **subject**.
- 5254 `urn:oasis:names:tc:xacml:1.0:subject:subject-id`
- 5255 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.
- 5256
- 5257 `urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier`
- 5258 This identifier indicates a public key used to confirm the **subject's** identity.
- 5259 `urn:oasis:names:tc:xacml:1.0:subject:key-info`
- 5260 This identifier indicates the time at which the **subject** was authenticated.
- 5261 `urn:oasis:names:tc:xacml:1.0:subject:authentication-time`
- 5262 This identifier indicates the method used to authenticate the **subject**.
- 5263 `urn:oasis:names:tc:xacml:1.0:subject:authentication-method`
- 5264 This identifier indicates the time at which the **subject** initiated the **access** request, according to the **PEP**.
- 5265 `urn:oasis:names:tc:xacml:1.0:subject:request-time`
- 5266 This identifier indicates the time at which the **subject's** current session began, according to the **PEP**.
- 5267 `urn:oasis:names:tc:xacml:1.0:subject:session-start-time`
- 5268 The following identifiers indicate the location where authentication credentials were activated.
- 5269 This identifier indicates that the location is expressed as an IP address.
- 5270 `urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-address`
- 5271 The corresponding **attribute** SHALL be of data-type "`urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`".
- 5272 This identifier indicates that the location is expressed as a DNS name.
- 5273 `urn:oasis:names:tc:xacml:3.0:subject:authn-locality:dns-name`
- 5274 The corresponding **attribute** SHALL be of data-type "`urn:oasis:names:tc:xacml:2.0:data-type:dnsName`".

5275 Where a suitable **attribute** is already defined in LDAP [LDAP-1], [LDAP-2], the XACML identifier SHALL
5276 be formed by adding the **attribute** name to the URI of the LDAP specification. For example, the **attribute**
5277 name for the userPassword defined in the RFC 2256 SHALL be:
5278 `http://www.ietf.org/rfc/rfc2256.txt#userPassword`

5279 B.5 Resource attributes

5280 These identifiers indicate **attributes** of the **resource**. When used, it is RECOMMENDED they appear
5281 within the <Attributes> element of the request **context** with Category
5282 `urn:oasis:names:tc:xacml:3.0:attribute-category:resource`.

5283 This **attribute** identifies the **resource** to which **access** is requested.

5284 `urn:oasis:names:tc:xacml:1.0:resource:resource-id`

5285 This **attribute** identifies the namespace of the top element(s) of the contents of the <Content> element.
5286 In the case where the **resource** content is supplied in the request **context** and the **resource**
5287 namespaces are defined in the **resource**, the **PEP** MAY provide this **attribute** in the request to indicate
5288 the namespaces of the **resource** content. In this case there SHALL be one value of this **attribute** for
5289 each unique namespace of the top level elements in the <Content> element. The type of the
5290 corresponding **attribute** SHALL be “`http://www.w3.org/2001/XMLSchema#anyURI`”.

5291 `urn:oasis:names:tc:xacml:2.0:resource:target-namespace`

5292 B.6 Action attributes

5293 These identifiers indicate **attributes** of the **action** being requested. When used, it is RECOMMENDED
5294 they appear within the <Attributes> element of the request **context** with Category
5295 `urn:oasis:names:tc:xacml:3.0:attribute-category:action`.

5296 This **attribute** identifies the **action** for which **access** is requested.

5297 `urn:oasis:names:tc:xacml:1.0:action:action-id`

5298 Where the **action** is implicit, the value of the action-id **attribute** SHALL be

5299 `urn:oasis:names:tc:xacml:1.0:action:implied-action`

5300 This **attribute** identifies the namespace in which the action-id **attribute** is defined.

5301 `urn:oasis:names:tc:xacml:1.0:action:action-namespace`

5302 B.7 Environment attributes

5303 These identifiers indicate **attributes** of the **environment** within which the **decision request** is to be
5304 evaluated. When used in the **decision request**, it is RECOMMENDED they appear in the
5305 <Attributes> element of the request **context** with Category `urn:oasis:names:tc:xacml:3.0:attribute-`
5306 `category:environment`.

5307 This identifier indicates the current time at the **context handler**. In practice it is the time at which the
5308 request **context** was created. For this reason, if these identifiers appear in multiple places within a
5309 <Policy> or <PolicySet>, then the same value SHALL be assigned to each occurrence in the
5310 evaluation procedure, regardless of how much time elapses between the processing of the occurrences.

5311 `urn:oasis:names:tc:xacml:1.0:environment:current-time`

5312 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#time`”.

5313 `urn:oasis:names:tc:xacml:1.0:environment:current-date`

5314 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#date`”.

5315 `urn:oasis:names:tc:xacml:1.0:environment:current-dateTime`

5316 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#dateTime`”.

5317 **B.8 Status codes**

5318 The following status code values are defined.

5319 This identifier indicates success.

5320 urn:oasis:names:tc:xacml:1.0:status:ok

5321 This identifier indicates that all the **attributes** necessary to make a **policy decision** were not available
5322 (see Section 5.58).

5323 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

5324 This identifier indicates that some **attribute** value contained a syntax error, such as a letter in a numeric
5325 field.

5326 urn:oasis:names:tc:xacml:1.0:status:syntax-error

5327 This identifier indicates that an error occurred during **policy** evaluation. An example would be division by
5328 zero.

5329 urn:oasis:names:tc:xacml:1.0:status:processing-error

5330 **B.9 Combining algorithms**

5331 The deny-overrides **rule-combining algorithm** has the following value for the ruleCombiningAlgId
5332 attribute:

5333 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides

5334 The deny-overrides **policy-combining algorithm** has the following value for the
5335 policyCombiningAlgId attribute:

5336 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides

5337 The permit-overrides **rule-combining algorithm** has the following value for the ruleCombiningAlgId
5338 attribute:

5339 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

5340 The permit-overrides **policy-combining algorithm** has the following value for the
5341 policyCombiningAlgId attribute:

5342 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

5343 The first-applicable **rule-combining algorithm** has the following value for the ruleCombiningAlgId
5344 attribute:

5345 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

5346 The first-applicable **policy-combining algorithm** has the following value for the
5347 policyCombiningAlgId attribute:

5348 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

5349 The only-one-applicable-policy **policy-combining algorithm** has the following value for the
5350 policyCombiningAlgId attribute:

5351 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

5352 The ordered-deny-overrides **rule-combining algorithm** has the following value for the
5353 ruleCombiningAlgId attribute:

5354 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

5355 The ordered-deny-overrides **policy-combining algorithm** has the following value for the
5356 policyCombiningAlgId attribute:

5357 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-
5358 overrides

5359 The ordered-permit-overrides **rule-combining algorithm** has the following value for the
5360 ruleCombiningAlgId attribute:

5361 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-
5362 overrides

5363 The ordered-permit-overrides **policy-combining algorithm** has the following value for the
5364 policyCombiningAlgId attribute:

5365 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-
5366 overrides

5367 The deny-unless-permit **rule-combining algorithm** has the following value for the
5368 policyCombiningAlgId attribute:

5369 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit

5370 The permit-unless-deny **rule-combining algorithm** has the following value for the
5371 policyCombiningAlgId attribute:

5372 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny

5373 The deny-unless-permit **policy-combining algorithm** has the following value for the
5374 policyCombiningAlgId attribute:

5375 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit

5376 The permit-unless-deny **policy-combining algorithm** has the following value for the
5377 policyCombiningAlgId attribute:

5378 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny

5379 The legacy deny-overrides **rule-combining algorithm** has the following value for the
5380 ruleCombiningAlgId attribute:

5381 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

5382 The legacy deny-overrides **policy-combining algorithm** has the following value for the
5383 policyCombiningAlgId attribute:

5384 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides

5385 The legacy permit-overrides **rule-combining algorithm** has the following value for the
5386 ruleCombiningAlgId attribute:

5387 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides

5388 The legacy permit-overrides **policy-combining algorithm** has the following value for the
5389 policyCombiningAlgId attribute:

5390 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides

5391 The legacy ordered-deny-overrides **rule-combining algorithm** has the following value for the
5392 ruleCombiningAlgId attribute:

5393 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides

5394 The legacy ordered-deny-overrides **policy-combining algorithm** has the following value for the
5395 policyCombiningAlgId attribute:

5396 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-
5397 overrides

5398 The legacy ordered-permit-overrides **rule-combining algorithm** has the following value for the
5399 ruleCombiningAlgId attribute:

5400 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-
5401 overrides

5402 The legacy ordered-permit-overrides **policy-combining algorithm** has the following value for the
5403 policyCombiningAlgId attribute:

5404 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-
5405 overrides

5406

5407

Appendix C. Combining algorithms (normative)

5408 This section contains a description of the *rule-* and *policy-combining algorithms* specified by XACML.
5409 Pseudo code is normative, descriptions in English are non-normative.

5410 The legacy *combining algorithms* are defined in previous versions of XACML, and are retained for
5411 compatibility reasons. It is RECOMMENDED that the new *combining algorithms* are used instead of the
5412 legacy *combining algorithms* for new use.

5413 Note that in each case an implementation is conformant as long as it produces the same result as is
5414 specified here, regardless of how and in what order the implementation behaves internally.

5415 C.1 Extended Indeterminate values

5416 Some combining algorithms are defined in terms of an extended set of "Indeterminate" values. See
5417 section 7.10 for the definition of the Extended Indeterminate values. For these algorithms, the *PDP* MUST
5418 keep track of the extended set of "Indeterminate" values during *rule* and *policy* combining.

5419 The output of a combining algorithm which does not track the extended set of "Indeterminate" values
5420 MUST be treated as "Indeterminate{DP}" for the value "Indeterminate" by a combining algorithm which
5421 tracks the extended set of "Indeterminate" values.

5422 A combining algorithm which does not track the extended set of "Indeterminate" values MUST treat the
5423 output of a combining algorithm which tracks the extended set of "Indeterminate" values as an
5424 "Indeterminate" for any of the possible values of the extended set of "Indeterminate".

5425 C.2 Deny-overrides

5426 This section defines the "Deny-overrides" *rule-combining algorithm* of a *policy* and *policy-combining*
5427 *algorithm* of a *policy set*.

5428 This *combining algorithm* makes use of the extended "Indeterminate".

5429 The *rule combining algorithm* defined here has the following identifier:

5430 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides

5431 The *policy combining algorithm* defined here has the following identifier:

5432 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides

5433 The following is a non-normative informative description of this *combining algorithm*.

5434 The deny overrides *combining algorithm* is intended for those cases where a deny
5435 decision should have priority over a permit decision. This algorithm has the following
5436 behavior.

- 5437 1. If any decision is "Deny", the result is "Deny".
5438 2. Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
5439 3. Otherwise, if any decision is "Indeterminate{D}" and another decision is "Indeterminate{P}" or
5440 Permit, the result is "Indeterminate{DP}".
5441 4. Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
5442 5. Otherwise, if any decision is "Permit", the result is "Permit".
5443 6. Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
5444 7. Otherwise, the result is "NotApplicable".

5445 The following pseudo-code represents the normative specification of this *combining algorithm*. The
5446 algorithm is presented here in a form where the input to it is an array with children (the *policies*, *policy*
5447 *sets* or *rules*) of the *policy* or *policy set*. The children may be processed in any order, so the set of
5448 obligations or advice provided by this algorithm is not deterministic.

```

5449 Decision denyOverridesCombiningAlgorithm(Node[] children)
5450 {
5451     Boolean atLeastOneErrorD = false;
5452     Boolean atLeastOneErrorP = false;
5453     Boolean atLeastOneErrorDP = false;
5454     Boolean atLeastOnePermit = false;
5455     for( i=0 ; i < lengthOf(children) ; i++ )
5456     {
5457         Decision decision = children[i].evaluate();
5458         if (decision == Deny)
5459         {
5460             return Deny;
5461         }
5462         if (decision == Permit)
5463         {
5464             atLeastOnePermit = true;
5465             continue;
5466         }
5467         if (decision == NotApplicable)
5468         {
5469             continue;
5470         }
5471         if (decision == Indeterminate{D})
5472         {
5473             atLeastOneErrorD = true;
5474             continue;
5475         }
5476         if (decision == Indeterminate{P})
5477         {
5478             atLeastOneErrorP = true;
5479             continue;
5480         }
5481         if (decision == Indeterminate{DP})
5482         {
5483             atLeastOneErrorDP = true;
5484             continue;
5485         }
5486     }
5487     if (atLeastOneErrorDP)
5488     {
5489         return Indeterminate{DP};
5490     }
5491     if (atLeastOneErrorD && (atLeastOneErrorP || atLeastOnePermit))
5492     {
5493         return Indeterminate{DP};
5494     }
5495     if (atLeastOneErrorD)
5496     {
5497         return Indeterminate{D};
5498     }
5499     if (atLeastOnePermit)
5500     {
5501         return Permit;
5502     }
5503     if (atLeastOneErrorP)
5504     {
5505         return Indeterminate{P};
5506     }
5507     return NotApplicable;
5508 }

```

5509 **Obligations** and **advice** shall be combined as described in Section 7.18.

5510 C.3 Ordered-deny-overrides

5511 The following specification defines the "Ordered-deny-overrides" **rule-combining algorithm** of a **policy**.

5512 The behavior of this algorithm is identical to that of the "Deny-overrides" **rule-combining**
5513 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
5514 match the order as listed in the **policy**.

5515 The **rule combining algorithm** defined here has the following identifier:

5516 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

5517 The following specification defines the "Ordered-deny-overrides" **policy-combining algorithm** of a
5518 **policy set**.

5519 The behavior of this algorithm is identical to that of the "Deny-overrides" **policy-combining**
5520 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
5521 match the order as listed in the **policy set**.

5522 The **policy combining algorithm** defined here has the following identifier:

5523 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-
5524 overrides

5525 C.4 Permit-overrides

5526 This section defines the "Permit-overrides" **rule-combining algorithm** of a **policy** and **policy-combining**
5527 **algorithm** of a **policy set**.

5528 This **combining algorithm** makes use of the extended "Indeterminate".

5529 The **rule combining algorithm** defined here has the following identifier:

5530 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

5531 The **policy combining algorithm** defined here has the following identifier:

5532 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

5533 The following is a non-normative informative description of this combining algorithm.

5534 The permit overrides **combining algorithm** is intended for those cases where a permit
5535 decision should have priority over a deny decision. This algorithm has the following
5536 behavior.

- 5537 1. If any decision is "Permit", the result is "Permit".
- 5538 2. Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
- 5539 3. Otherwise, if any decision is "Indeterminate{P}" and another decision is
5540 "Indeterminate{D} or Deny, the result is "Indeterminate{DP}".
- 5541 4. Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
- 5542 5. Otherwise, if any decision is "Deny", the result is "Deny".
- 5543 6. Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
- 5544 7. Otherwise, the result is "NotApplicable".

5545 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5546 algorithm is presented here in a form where the input to it is an array with all children (the **policies**, **policy**
5547 **sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set of
5548 obligations or advice provided by this algorithm is not deterministic.

```
5549 Decision permitOverridesCombiningAlgorithm(Node[] children)
5550 {
5551     Boolean atLeastOneErrorD = false;
5552     Boolean atLeastOneErrorP = false;
5553     Boolean atLeastOneErrorDP = false;
5554     Boolean atLeastOneDeny = false;
```

```

5555 for( i=0 ; i < lengthOf(children) ; i++ )
5556 {
5557     Decision decision = children[i].evaluate();
5558     if (decision == Deny)
5559     {
5560         atLeastOneDeny = true;
5561         continue;
5562     }
5563     if (decision == Permit)
5564     {
5565         return Permit;
5566     }
5567     if (decision == NotApplicable)
5568     {
5569         continue;
5570     }
5571     if (decision == Indeterminate{D})
5572     {
5573         atLeastOneErrorD = true;
5574         continue;
5575     }
5576     if (decision == Indeterminate{P})
5577     {
5578         atLeastOneErrorP = true;
5579         continue;
5580     }
5581     if (decision == Indeterminate{DP})
5582     {
5583         atLeastOneErrorDP = true;
5584         continue;
5585     }
5586 }
5587 if (atLeastOneErrorDP)
5588 {
5589     return Indeterminate{DP};
5590 }
5591 if (atLeastOneErrorP && (atLeastOneErrorD || atLeastOneDeny))
5592 {
5593     return Indeterminate{DP};
5594 }
5595 if (atLeastOneErrorP)
5596 {
5597     return Indeterminate{P};
5598 }
5599 if (atLeastOneDeny)
5600 {
5601     return Deny;
5602 }
5603 if (atLeastOneErrorD)
5604 {
5605     return Indeterminate{D};
5606 }
5607 return NotApplicable;
5608 }

```

5609 **Obligations** and **advice** shall be combined as described in Section 7.18.

5610 C.5 Ordered-permit-overrides

5611 The following specification defines the "Ordered-permit-overrides" **rule-combining algorithm** of a **policy**.

5612 The behavior of this algorithm is identical to that of the "Permit-overrides" **rule-combining**
5613 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
5614 match the order as listed in the **policy**.

5615 The **rule combining algorithm** defined here has the following identifier:
5616 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-
5617 overrides
5618 The following specification defines the "Ordered-permit-overrides" **policy-combining algorithm** of a
5619 **policy set**.
5620 The behavior of this algorithm is identical to that of the "Permit-overrides" **policy-combining**
5621 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
5622 match the order as listed in the **policy set**.
5623 The **policy combining algorithm** defined here has the following identifier:
5624 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-
5625 overrides

5626 C.6 Deny-unless-permit

5627 This section defines the "Deny-unless-permit" **rule-combining algorithm** of a **policy** or **policy-**
5628 **combining algorithm** of a **policy set**.

5629 The **rule combining algorithm** defined here has the following identifier:
5630 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit

5631 The **policy combining algorithm** defined here has the following identifier:
5632 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit

5633 The following is a non-normative informative description of this **combining algorithm**.

5634 The "Deny-unless-permit" **combining algorithm** is intended for those cases where a
5635 permit decision should have priority over a deny decision, and an "Indeterminate" or
5636 "NotApplicable" must never be the result. It is particularly useful at the top level in a
5637 **policy** structure to ensure that a **PDP** will always return a definite "Permit" or "Deny"
5638 result. This algorithm has the following behavior.

- 5639 1. If any decision is "Permit", the result is "Permit".
- 5640 2. Otherwise, the result is "Deny".

5641 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5642 algorithm is presented here in a form where the input to it is an array with all the children (the **policies**,
5643 **policy sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set
5644 of obligations or advice provided by this algorithm is not deterministic.

```
5645 Decision denyUnlessPermitCombiningAlgorithm(Node[] children)  
5646 {  
5647     for( i=0 ; i < lengthOf(children) ; i++ )  
5648     {  
5649         if (children[i].evaluate() == Permit)  
5650         {  
5651             return Permit;  
5652         }  
5653     }  
5654     return Deny;  
5655 }
```

5656 **Obligations** and **advice** shall be combined as described in Section 7.18.

5657 C.7 Permit-unless-deny

5658 This section defines the "Permit-unless-deny" **rule-combining algorithm** of a **policy** or **policy-**
5659 **combining algorithm** of a **policy set**.

5660 The **rule combining algorithm** defined here has the following identifier:

5661 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny

5662 The **policy combining algorithm** defined here has the following identifier:
5663 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny
5664 The following is a non-normative informative description of this **combining algorithm**.

5665 The "Permit-unless-deny" **combining algorithm** is intended for those cases where a
5666 deny decision should have priority over a permit decision, and an "Indeterminate" or
5667 "NotApplicable" must never be the result. It is particularly useful at the top level in a
5668 **policy** structure to ensure that a **PDP** will always return a definite "Permit" or "Deny"
5669 result. This algorithm has the following behavior.

- 5670 1. If any decision is "Deny", the result is "Deny".
- 5671 2. Otherwise, the result is "Permit".

5672 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5673 algorithm is presented here in a form where the input to it is an array with all the children (the **policies**,
5674 **policy sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set
5675 of obligations or advice provided by this algorithm is not deterministic.

```
5676 Decision permitUnlessDenyCombiningAlgorithm(Node[] children)
5677 {
5678     for( i=0 ; i < lengthOf(children) ; i++ )
5679     {
5680         if (children[i].evaluate() == Deny)
5681         {
5682             return Deny;
5683         }
5684     }
5685     return Permit;
5686 }
```

5687 **Obligations** and **advice** shall be combined as described in Section 7.18.

5688 C.8 First-applicable

5689 This section defines the "First-applicable" **rule-combining algorithm** of a **policy** and **policy-combining**
5690 **algorithm** of a **policy set**.

5691 The **rule combining algorithm** defined here has the following identifier:

5692 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

5693 The following is a non-normative informative description of the "First-Applicable" **rule-combining**
5694 **algorithm** of a **policy**.

5695 Each **rule** SHALL be evaluated in the order in which it is listed in the **policy**. For a particular
5696 **rule**, if the **target** matches and the **condition** evaluates to "True", then the evaluation of the
5697 **policy** SHALL halt and the corresponding **effect** of the **rule** SHALL be the result of the evaluation
5698 of the **policy** (i.e. "Permit" or "Deny"). For a particular **rule** selected in the evaluation, if the
5699 **target** evaluates to "False" or the **condition** evaluates to "False", then the next **rule** in the order
5700 SHALL be evaluated. If no further **rule** in the order exists, then the **policy** SHALL evaluate to
5701 "NotApplicable".

5702 If an error occurs while evaluating the **target** or **condition** of a **rule**, then the evaluation SHALL
5703 halt, and the **policy** shall evaluate to "Indeterminate", with the appropriate error status.

5704 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5705 Decision firstApplicableEffectRuleCombiningAlgorithm(Rule[] rules)
5706 {
5707     for( i = 0 ; i < lengthOf(rules) ; i++ )
5708     {
5709         Decision decision = evaluate(rules[i]);
5710         if (decision == Deny)
5711         {
```

```

5712         return Deny;
5713     }
5714     if (decision == Permit)
5715     {
5716         return Permit;
5717     }
5718     if (decision == NotApplicable)
5719     {
5720         continue;
5721     }
5722     if (decision == Indeterminate)
5723     {
5724         return Indeterminate;
5725     }
5726 }
5727 return NotApplicable;
5728 }

```

5729 The **policy combining algorithm** defined here has the following identifier:

5730 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

5731 The following is a non-normative informative description of the "First-applicable" **policy-combining algorithm** of a **policy set**.

5733 Each **policy** is evaluated in the order that it appears in the **policy set**. For a particular **policy**, if the **target** evaluates to "True" and the **policy** evaluates to a determinate value of "Permit" or "Deny", then the evaluation SHALL halt and the **policy set** SHALL evaluate to the **effect** value of that **policy**. For a particular **policy**, if the **target** evaluate to "False", or the **policy** evaluates to "NotApplicable", then the next **policy** in the order SHALL be evaluated. If no further **policy** exists in the order, then the **policy set** SHALL evaluate to "NotApplicable".

5739 If an error were to occur when evaluating the **target**, or when evaluating a specific **policy**, the reference to the **policy** is considered invalid, or the **policy** itself evaluates to "Indeterminate", then the evaluation of the **policy-combining algorithm** shall halt, and the **policy set** shall evaluate to "Indeterminate" with an appropriate error status.

5743 The following pseudo-code represents the normative specification of this policy-combination algorithm.

```

5744 Decision firstApplicableEffectPolicyCombiningAlgorithm(Policy[] policies)
5745 {
5746     for( i = 0 ; i < lengthOf(policies) ; i++ )
5747     {
5748         Decision decision = evaluate(policies[i]);
5749         if(decision == Deny)
5750         {
5751             return Deny;
5752         }
5753         if(decision == Permit)
5754         {
5755             return Permit;
5756         }
5757         if (decision == NotApplicable)
5758         {
5759             continue;
5760         }
5761         if (decision == Indeterminate)
5762         {
5763             return Indeterminate;
5764         }
5765     }
5766     return NotApplicable;
5767 }

```

5768 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

5769 C.9 Only-one-applicable

5770 This section defines the "Only-one-applicable" **policy-combining algorithm** of a **policy set**.

5771 The **policy combining algorithm** defined here has the following identifier:

5772 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

5773 The following is a non-normative informative description of the "Only-one-applicable" **policy-combining algorithm** of a **policy set**.

5775 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".

5779 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**.

5781 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.

5784 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```
5785 Decision onlyOneApplicablePolicyPolicyCombiningAlogrithm(Policy[] policies)
5786 {
5787     Boolean          atLeastOne      = false;
5788     Policy           selectedPolicy = null;
5789     ApplicableResult appResult;
5790
5791     for ( i = 0; i < lengthOf(policies) ; i++ )
5792     {
5793         appResult = isApplicable(policies[I]);
5794
5795         if ( appResult == Indeterminate )
5796         {
5797             return Indeterminate;
5798         }
5799         if( appResult == Applicable )
5800         {
5801             if ( atLeastOne )
5802             {
5803                 return Indeterminate;
5804             }
5805             else
5806             {
5807                 atLeastOne      = true;
5808                 selectedPolicy = policies[i];
5809             }
5810         }
5811         if ( appResult == NotApplicable )
5812         {
5813             continue;
5814         }
5815     }
5816     if ( atLeastOne )
5817     {
5818         return evaluate(selectedPolicy);
5819     }
5820     else
5821     {
5822         return NotApplicable;
5823     }
5824 }
```

5825 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

5826 C.10 Legacy Deny-overrides

5827 This section defines the legacy “Deny-overrides” **rule-combining algorithm** of a **policy** and **policy-**
5828 **combining algorithm** of a **policy set**.

5829

5830 The **rule combining algorithm** defined here has the following identifier:

5831 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

5832 The following is a non-normative informative description of this combining algorithm.

5833 The “Deny-overrides” rule combining algorithm is intended for those cases where a deny
5834 decision should have priority over a permit decision. This algorithm has the following
5835 behavior.

- 5836 1. If any rule evaluates to "Deny", the result is "Deny".
- 5837 2. Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is
5838 "Indeterminate".
- 5839 3. Otherwise, if any rule evaluates to "Permit", the result is "Permit".
- 5840 4. Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate", the result is
5841 "Indeterminate".
- 5842 5. Otherwise, the result is "NotApplicable".

5843 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5844 Decision denyOverridesRuleCombiningAlgorithm(Rule[] rules)
5845 {
5846     Boolean atLeastOneError = false;
5847     Boolean potentialDeny = false;
5848     Boolean atLeastOnePermit = false;
5849     for( i=0 ; i < lengthOf(rules) ; i++ )
5850     {
5851         Decision decision = evaluate(rules[i]);
5852         if (decision == Deny)
5853         {
5854             return Deny;
5855         }
5856         if (decision == Permit)
5857         {
5858             atLeastOnePermit = true;
5859             continue;
5860         }
5861         if (decision == NotApplicable)
5862         {
5863             continue;
5864         }
5865         if (decision == Indeterminate)
5866         {
5867             atLeastOneError = true;
5868
5869             if (effect(rules[i]) == Deny)
5870             {
5871                 potentialDeny = true;
5872             }
5873             continue;
5874         }
5875     }
5876     if (potentialDeny)
5877     {
5878         return Indeterminate;
5879     }
5880     if (atLeastOnePermit)
5881     {
```

```

5882     return Permit;
5883 }
5884 if (atLeastOneError)
5885 {
5886     return Indeterminate;
5887 }
5888 return NotApplicable;
5889 }

```

5890 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

5891 The **policy combining algorithm** defined here has the following identifier:

5892 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides

5893 The following is a non-normative informative description of this combining algorithm.

5894 The "Deny-overrides" policy combining algorithm is intended for those cases where a
5895 deny decision should have priority over a permit decision. This algorithm has the
5896 following behavior.

- 5897 1. If any policy evaluates to "Deny", the result is "Deny".
- 5898 2. Otherwise, if any policy evaluates to "Indeterminate", the result is "Deny".
- 5899 3. Otherwise, if any policy evaluates to "Permit", the result is "Permit".
- 5900 4. Otherwise, the result is "NotApplicable".

5901 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```

5902 Decision denyOverridesPolicyCombiningAlgorithm(Policy[] policies)
5903 {
5904     Boolean atLeastOnePermit = false;
5905     for( i=0 ; i < lengthOf(policies) ; i++ )
5906     {
5907         Decision decision = evaluate(policies[i]);
5908         if (decision == Deny)
5909         {
5910             return Deny;
5911         }
5912         if (decision == Permit)
5913         {
5914             atLeastOnePermit = true;
5915             continue;
5916         }
5917         if (decision == NotApplicable)
5918         {
5919             continue;
5920         }
5921         if (decision == Indeterminate)
5922         {
5923             return Deny;
5924         }
5925     }
5926     if (atLeastOnePermit)
5927     {
5928         return Permit;
5929     }
5930     return NotApplicable;
5931 }

```

5932 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

5933 C.11 Legacy Ordered-deny-overrides

5934 The following specification defines the legacy "Ordered-deny-overrides" **rule-combining algorithm** of a
5935 **policy**.

5936 The behavior of this algorithm is identical to that of the “Deny-overrides” **rule-combining**
5937 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
5938 match the order as listed in the **policy**.

5939 The **rule combining algorithm** defined here has the following identifier:

5940 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides

5941 The following specification defines the legacy “Ordered-deny-overrides” **policy-combining algorithm** of
5942 a **policy set**.

5943 The behavior of this algorithm is identical to that of the “Deny-overrides” **policy-combining**
5944 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
5945 match the order as listed in the **policy set**.

5946 The **rule combining algorithm** defined here has the following identifier:

5947 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-
5948 overrides

5949 C.12 Legacy Permit-overrides

5950 This section defines the legacy “Permit-overrides” **rule-combining algorithm** of a **policy** and **policy-**
5951 **combining algorithm** of a **policy set**.

5952 The **rule combining algorithm** defined here has the following identifier:

5953 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides

5954 The following is a non-normative informative description of this combining algorithm.

5955 The “Permit-overrides” rule combining algorithm is intended for those cases where a
5956 permit decision should have priority over a deny decision. This algorithm has the
5957 following behavior.

- 5958 1. If any rule evaluates to "Permit", the result is "Permit".
- 5959 2. Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate" the result is
5960 "Indeterminate".
- 5961 3. Otherwise, if any rule evaluates to "Deny", the result is "Deny".
- 5962 4. Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is
5963 "Indeterminate".
- 5964 5. Otherwise, the result is "NotApplicable".

5965 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5966 Decision permitOverridesRuleCombiningAlgorithm(Rule[] rules)
5967 {
5968     Boolean atLeastOneError = false;
5969     Boolean potentialPermit = false;
5970     Boolean atLeastOneDeny = false;
5971     for( i=0 ; i < lengthOf(rules) ; i++ )
5972     {
5973         Decision decision = evaluate(rules[i]);
5974         if (decision == Deny)
5975         {
5976             atLeastOneDeny = true;
5977             continue;
5978         }
5979         if (decision == Permit)
5980         {
5981             return Permit;
5982         }
5983         if (decision == NotApplicable)
5984         {
5985             continue;
5986         }
5987     }
5988 }
```

```

5987     if (decision == Indeterminate)
5988     {
5989         atLeastOneError = true;
5990
5991         if (effect(rules[i]) == Permit)
5992         {
5993             potentialPermit = true;
5994         }
5995         continue;
5996     }
5997 }
5998 if (potentialPermit)
5999 {
6000     return Indeterminate;
6001 }
6002 if (atLeastOneDeny)
6003 {
6004     return Deny;
6005 }
6006 if (atLeastOneError)
6007 {
6008     return Indeterminate;
6009 }
6010 return NotApplicable;
6011 }

```

6012 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

6013 The **policy combining algorithm** defined here has the following identifier:

6014 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides

6015 The following is a non-normative informative description of this combining algorithm.

6016 The "Permit-overrides" policy combining algorithm is intended for those cases where a
6017 permit decision should have priority over a deny decision. This algorithm has the
6018 following behavior.

- 6019 1. If any policy evaluates to "Permit", the result is "Permit".
- 6020 2. Otherwise, if any policy evaluates to "Deny", the result is "Deny".
- 6021 3. Otherwise, if any policy evaluates to "Indeterminate", the result is "Indeterminate".
- 6022 4. Otherwise, the result is "NotApplicable".

6023 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```

6024 Decision permitOverridesPolicyCombiningAlgorithm(Policy[] policies)
6025 {
6026     Boolean atLeastOneError = false;
6027     Boolean atLeastOneDeny = false;
6028     for( i=0 ; i < lengthOf(policies) ; i++ )
6029     {
6030         Decision decision = evaluate(policies[i]);
6031         if (decision == Deny)
6032         {
6033             atLeastOneDeny = true;
6034             continue;
6035         }
6036         if (decision == Permit)
6037         {
6038             return Permit;
6039         }
6040         if (decision == NotApplicable)
6041         {
6042             continue;
6043         }
6044         if (decision == Indeterminate)

```

```

6045     {
6046         atLeastOneError = true;
6047         continue;
6048     }
6049 }
6050 if (atLeastOneDeny)
6051 {
6052     return Deny;
6053 }
6054 if (atLeastOneError)
6055 {
6056     return Indeterminate;
6057 }
6058 return NotApplicable;
6059 }

```

6060 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

6061 C.13 Legacy Ordered-permit-overrides

6062 The following specification defines the legacy "Ordered-permit-overrides" **rule-combining algorithm** of a
6063 **policy**.

6064 The behavior of this algorithm is identical to that of the "Permit-overrides" **rule-combining**
6065 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
6066 match the order as listed in the **policy**.

6067 The **rule combining algorithm** defined here has the following identifier:

6068 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-
6069 overrides

6070 The following specification defines the legacy "Ordered-permit-overrides" **policy-combining algorithm** of
6071 a **policy set**.

6072 The behavior of this algorithm is identical to that of the "Permit-overrides" **policy-combining**
6073 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
6074 match the order as listed in the **policy set**.

6075 The **policy combining algorithm** defined here has the following identifier:

6076 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-
6077 overrides

6078

6079

Appendix D. Acknowledgements

6080 The following individuals have participated in the creation of this specification and are gratefully
6081 acknowledged:

6082

6083 Anil Saldhana

6084 Anil Tappetla

6085 Anne Anderson

6086 Anthony Nadalin

6087 Bill Parducci

6088 Craig Forster

6089 David Chadwick

6090 David Staggs

6091 Dilli Arumugam

6092 Duane DeCouteau

6093 Erik Rissanen

6094 Gareth Richards

6095 Hal Lockhart

6096 Jan Herrmann

6097 John Tolbert

6098 Ludwig Seitz

6099 Michiharu Kudo

6100 Naomaru Itoi

6101 Paul Tyson

6102 Prateek Mishra

6103 Rich Levinson

6104 Ronald Jacobson

6105 Seth Proctor

6106 Sridhar Muppidi

6107 Tim Moses

6108 Vernon Murdoch

6109 [Cyril Dangerville \(not a member of XACML TC\)](#)

6110

Appendix E. Revision History

6111

6112

| 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> |

6113