ITUPublications
Recommendations

International Telecommunication Union
Standardization Sector

Recommendation

ITU-T X.1144 (04/2024)

SERIES X: Data networks, open system communications and security

Secure applications and services (I) – Web security (I)

---

eXtensible Access Control Markup Language (XACML) 3.1

ITU-T X-SERIES RECOMMENDATIONS

Data networks, open system communications and security

PUBLIC DATA NETWORKS	X.1-X.199
OPEN SYSTEMS INTERCONNECTION	X.200-X.299
INTERWORKING BETWEEN NETWORKS	X.300-X.399
MESSAGE HANDLING SYSTEMS	X.400-X.499
DIRECTORY	X.500-X.599
OSI NETWORKING AND SYSTEM ASPECTS	X.600-X.699
OSI MANAGEMENT	X.700-X.799
SECURITY	X.800-X.849
OSI APPLICATIONS	X.850-X.899
OPEN DISTRIBUTED PROCESSING	X.900-X.999
INFORMATION AND NETWORK SECURITY	X.1000-X.1099
SECURE APPLICATIONS AND SERVICES (I)	X.1100-X.1199
Multicast security	X.1100-X.1109
Home network security	X.1110-X.1119
Mobile security	X.1120-X.1139
Web security (I)	**X.1140-X.1149**
Application Security (I)	X.1150-X.1159
Peer-to-peer security	X.1160-X.1164
Data protection (I) and networked ID security	X.1165-X.1179
Countering fraud	X.1180-X.1189
IPTV security	X.1190-X.1199
CYBERSPACE SECURITY	X.1200-X.1299
SECURE APPLICATIONS AND SERVICES (II)	X.1300-X.1499
CYBERSECURITY INFORMATION EXCHANGE	X.1500-X.1599
CLOUD COMPUTING SECURITY	X.1600-X.1699
QUANTUM COMMUNICATION	X.1700-X.1729
DATA SECURITY	X.1750-X.1799
INTERNATIONAL MOBILE TELECOMMUNICATIONS (IMT) SECURITY	X.1800-X.1839
METAVERSE AND DIGITAL TWIN SECURITY	X.2000-X.2199
SOFTWARE SUPPLY CHAIN SECURITY	X.2150-X.2199
ARTIFICIAL INTELLIGENCE (AI) / MACHINE LEARNING (ML) SECURITY	X.2200-X.2249

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T X.1144

eXtensible Access Control Markup Language (XACML) 3.1

Summary

Recommendation ITU-T X.1144 (2013) develops extensible access control markup language (XACML 3.0) which is an updated version of Recommendation ITU-T X.1142 (which is equivalent to OASIS XACML 2.0 (06/2006)).

Recommendation ITU-T X.1144 (2024) defines core XACML including syntax of the language, models, context with policy language model, syntax and processing rules. This Recommendation is technically equivalent and compatible with the OASIS XACML 3.1 standard.

History *

Edition	Recommendation	Approval	Study Group	Unique ID
1.0	ITU-T X.1144	2013-10-14	17	11.1002/1000/12044
2.0	ITU-T X.1144	2024-04-29	17	11.1002/1000/15877

Keywords

Coding, programming language, syntax, XACML, XML.

* To access the Recommendation, type the URL https://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Recommendation ITU-T X.1144

eXtensible Access Control Markup Language (XACML) 3.1

1 Scope

This Recommendation defines the eXtensible Access Control Markup Language (XACML) Version 3.1. It defines a common language for expressing security policy. The motivation behind XACML is to develop an XML based policy language that can be used:

– To provide a method for flexible definition of the procedure by which rules and policies are combined.

– To provide a method for dealing with multiple subjects acting in different capacities.

– To provide a method for basing an authorization decision on attributes of the subject and resource.

– To provide a method for dealing with multi-valued attributes.

– To provide a method for basing an authorization decision on the contents of an information resource.

– To provide a set of logical and mathematical operators on attributes of the subject, resource and environment.

– To provide a method for handling a distributed set of policy components, while abstracting the method for locating, retrieving and authenticating the policy components.

– To provide a method for rapidly identifying the policy that applies to a given action, based upon the values of attributes of the subject, resource and action.

– To provide an abstraction-layer that insulates the policy-writer from the details of the application environment.

– To provide a method for specifying a set of actions that must be performed in conjunction with policy enforcement.

The core XACML solutions are included in this Recommendation. Clause 7 develops XACML models. Clause 8 develops policy language. Clause 10 develops policy processing rules. Clause 11 develops guidelines for implementers.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T X.500] Recommendation ITU-T X.500 (2012) | ISO/IEC 9594-1:2014, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services*.

[ITU-T X.509] Recommendation ITU-T X.509 (2012) | ISO/IEC 9594-8:2014, *Information technology – Open Systems Interconnection – The Directory: Public key and attribute certificate frameworks*.

[ITU-T X.690] Recommendation ITU-T X.690 (2008) | ISO/IEC 8825-1:2008, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*.

[IETF RFC 2119] IETF RFC 2119 (1997), *Key words for use in RFCs to Indicate Requirement Levels*.
<http://www.ietf.org/rfc/rfc2119.txt>

[IETF RFC 2256] IETF RFC 2256 (1997), *A summary of the X500(96) User Schema for use with LDAPv3*.
<http://www.ietf.org/rfc/rfc2256.txt>

[IETF RFC 2732] IETF RFC 2732 (1999), *Format for Literal IPv6 Addresses in URL's*.
<http://www.ietf.org/rfc/rfc2732.txt>

[IETF RFC 2798] IETF RFC 2798 (2000), *Definition of the inetOrgPerson LDAP Object Class*.

[IETF RFC 3198] IETF RFC 3198 (2001), *Terminology for Policy-Based Management*.
<http://www.ietf.org/rfc/rfc3198.txt>

[IETF RFC 4514] IETF RFC 4514 (2006), *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*.
<http://www.ietf.org/rfc/rfc4514.txt>

[IETF RFC 5280] IETF RFC 5280 (2008), *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.
<http://www.ietf.org/rfc/rfc5280.txt>

[IETF RFC 5321] IETF RFC 5321 (2008), *Simple Mail Transfer Protocol*.
<http://www.ietf.org/rfc/rfc5321.txt>

[IEEE 754] IEEE 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*.

[IEEE 854] IEEE 854-1987, *IEEE Standard for Radix-Independent Floating-Point Arithmetic*.

[INFOSET] XML Information Set (Second Edition), W3C Recommendation 4 February 2004, <https://www.w3.org/TR/xml-infoset/>

[W3C DS] W3C Recommendation (10 June 2008), *XML-Signature Syntax and Processing*.
<http://www.w3.org/TR/xmldsig-core/>

[W3C EXC-C14N] W3C Recommendation (18 July 2002), *Exclusive XML Canonicalization, Version 1.0*.
<http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/>

[W3C MathML] W3C Recommendation (21 October 2003), *Mathematical Markup Language (MathML), Version 2.0*.
<http://www.w3.org/TR/2003/REC-MathML2-20031021/>

[W3C XF] W3C Recommendation (23 January 2007), *XQuery 1.0 and XPath 2.0 Functions and Operators*.
<http://www.w3.org/TR/2007/REC-xpath-functions-20070123/>

[W3C XML] W3C Recommendation (26 November 2008), *Extensible Markup Language (XML) 1.0 (Fifth Edition)*.
<http://www.w3.org/TR/2008/REC-xml-20081126/>

[W3C XS] W3C Recommendation (28 October 2004), *XML Schema, parts 1 and 2*.
<http://www.w3.org/TR/xmlschema-1/> and <http://www.w3.org/TR/xmlschema-2/>

[W3C XPath] W3C Recommendation (16 November 1999), *XML Path Language (XPath), Version 1.0*.
<http://www.w3.org/TR/xpath>

[W3C XPathFunc] W3C Recommendation (14 December 2010), *XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)*. <http://www.w3.org/TR/2010/REC-xpath-functions-20101214/>

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 access [b-IETF RFC 4949]: The ability and means to communicate with or otherwise interact with a system to use system resources either to handle information or to gain knowledge of the information the system contains.

3.1.2 access control [b-IETF RFC 4949]: A process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy.

3.1.3 access control information [b-ITU-T X.812]: Any information used for access control purposes, including contextual information.

3.1.4 data object [W3C DS]: The actual binary/octet data being operated on (transformed, digested, or signed) by an application – frequently an HTTP entity.

NOTE – The proper noun Object designates a specific XML element. Occasionally we refer to a data object as a document or as a resource's content. The term element content is used to describe the data between XML start and end tags. The term XML document is used to describe data objects which conform to the XML specification.

3.1.5 namespace [b-W3C Glossary]: A qualifier added to an XML tag to ensure uniqueness among XML elements.

3.1.6 policy decision point [IETF RFC 3198]: The system entity that evaluates applicable policy and renders an authorization decision. This term is defined in a joint effort by the IETF Policy Framework Working Group and the Distributed Management Task Force (DMTF)/common information model (CIM) in [IETF RFC 3198]. This term corresponds to "access control decision function" (ADF) in [b-ITU-T X.812].

3.1.7 policy enforcement point [IETF RFC 3198]: The system entity that performs access control, by making decision requests and enforcing authorization decisions. This term is defined in a joint effort by the IETF Policy Framework Working Group and the Distributed Management Task Force (DMTF)/common information model (CIM) in [IETF RFC 3198]. This term corresponds to "access control enforcement function" (AEF) in [b-ITU-T X.812].

3.1.8 principle [b-ITU-T X.811]: An entity whose identity can be authenticated.

3.1.9 security architecture [b-IETF RFC 4949]: A plan and set of principles that describe (a) the security services that a system is required to provide to meet the needs of its users, (b) the system components required to implement the services, and (c) the performance levels required in the components to deal with the threat environment.

3.1.10 security policy [b-IETF RFC 4949]: A set of policy rules (or principles) that direct how a system (or an organization) provides security services to protect sensitive and critical system resources.

3.1.11 security service [b-IETF RFC 4949]: A processing or communication service that is provided by a system to give a specific kind of protection to system resources.

3.1.12 type unification: The method by which two type expressions are "unified". The type expressions are matched along their structure. Where a type variable appears in one expression it is then "unified" to represent the corresponding structure element of the other expression, be it another

variable or subexpression. All variable assignments must remain consistent in both structures. Unification fails if the two expressions cannot be aligned, either by having dissimilar structure, or by having instance conflicts, such as a variable needs to represent both "xs:string" and "xs:integer". For a full explanation of type unification, please see [b-Hancock].

3.1.13 uniform resource identifier (URI) [b-IETF RFC 3986]: A URI is an identifier consisting of a sequence of characters matching the syntax rule named <URI> that consists of a hierarchical sequence of components referred to as the scheme, authority, path, query, and fragment. It enables uniform identification of resources via a separately defined extensible set of naming schemes.

3.1.14 URI reference [b-IETF RFC 3986]: URI-reference is used to denote the most common usage of a resource identifier (URI-reference = URI / relative-ref). A URI-reference is either a URI or a relative reference. If the URI-reference's prefix does not match the syntax of a scheme followed by its colon separator, then the URI-reference is a relative reference.

3.1.15 user [b-ITU-T X.812]: An entity (e.g., human user or computer-based entity) that attempts to access other entities.

3.1.16 XML Schema [b-W3C Technology]: An XML Schema is a language for expressing constraints about XML documents.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 action: An operation on a resource.

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

3.2.3 applicable policy: The set of policies and policy sets that governs access for a specific decision request.

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

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

3.2.6 bag: An unordered collection of values, in which there may be duplicate values.

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

3.2.8 conjunctive sequence: A sequence of predicates combined using the logical 'AND' operation.

3.2.9 context: The canonical representation of a decision request and an authorization decision.

3.2.10 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.

3.2.11 decision: The result of evaluating a rule, policy or policy set.

3.2.12 decision request: The request from PEP to PDP to render an authorization decision.

3.2.13 disjunctive sequence: A sequence of predicates combined using the logical 'OR' operation.

3.2.14 effect: The intended consequence of a satisfied rule (either "Permit" or "Deny").

3.2.15 environment: The set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource or action.

3.2.16 identifier equality: The identifier equality operation is defined in clause 10.20.

3.2.17 issuer: A set of attributes describing the source of a policy.

3.2.18 named attribute: A specific instance of an attribute, determined by the attribute name and type, the identity of the attribute holder (which may be of type: subject, resource, action or environment) and (optionally) the identity of the issuing authority.

3.2.19 obligation: An operation specified in a rule, policy or policy set that should be performed by PEP in conjunction with the enforcement of an authorization decision.

3.2.20 policy: A set of rules, an identifier for the rule-combining algorithm and (optionally) a set of obligations or advice. May be a component of a policy set.

3.2.21 policy administration point (PAP): The system entity that creates a policy or policy set.

3.2.22 policy-combining algorithm: The procedure for combining the decision and obligations from multiple policies.

3.2.23 policy information point (PIP): The system entity that acts as a source of attribute values.

3.2.24 policy set: A set of policies, other policy sets, a policy-combining algorithm and (optionally) a set of obligations or advice. May be a component of another policy set.

3.2.25 predicate: A statement about attributes the truth of which can be evaluated.

3.2.26 resource: Data, service or system component.

3.2.27 rule: A target, an effect, a condition and (optionally) a set of obligations or advice. A component of a policy.

3.2.28 rule-combining algorithm: The procedure for combining decisions from multiple rules.

3.2.29 subject: An actor whose attributes may be referenced by a predicate.

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

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

ADF Access control Decision Function

AEF Access control Enforcement Function

CIM Common Information Model

CORBA Common Object Request Broker Architecture

CRL Certificate Revocation List

DMTF Distributed Management Task Force

DNS Domain Name System

HTTP Hypertext Transfer Protocol

ID Identifier

IP Internet Protocol

IPSec	IP Security protocol
IPv4/v6	Internet Protocol version4/version 6
J2SE	Java 2 platform, Standard Edition
LDAP	Lightweight Directory Access Protocol
NFC	Normalization Form C
NFD	Normalization Form D
NFKC	Normalization Form KC
NFKD	Normalization Form KD
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
RDN	Relative Distinguished Name
SAML	Security Assertion Markup Language
SSL	Secure Socket Layer
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTC	Coordinated Universal Time
VPN	Virtual Private Network
WAN	Wide Area Network
XACML	extensible Access Control Markup Language
XML	extensible Markup Language
XPath	XML Path Language
XSLT	extensible Stylesheet Language Transformation

5 Conventions

The key words "**must**", "**must not**", "**required**", "**shall**", "**shall not**", "**should**", "**should not**", "**recommended**", "**may**", and "**optional**" in this Recommendation are to be interpreted as described in [IETF RFC 2119].

In descriptions of syntax, elements in angle brackets ("<", ">") are to be replaced by appropriate values, square brackets ("[", "]") enclose optional elements, elements in quotes are literal components, and "*" indicates that the preceding element may occur zero or more times.

6 Overview

The "economics of scale" have driven computing platform vendors to develop products with much generalized functionality, so that they can be used in the widest possible range of situations. "Out of the box", these products have the maximum possible privilege for accessing data and executing

software, so that they can be used in as many application environments as possible, including those with the most permissive security policies. In the more common case of a relatively restrictive security policy, the platform's inherent privileges must be constrained by configuration.

The security policy of a large enterprise has many elements and many points of enforcement. Elements of policy may be managed by the information systems department, by human resources, by the legal department and by the finance department. Furthermore, the policy may be enforced by the extranet, mail, the wide area network (WAN) and remote-access systems; platforms which inherently implement a permissive security policy. The current practice is to manage the configuration of each point of enforcement independently in order to implement the security policy as accurately as possible. Consequently, it is an expensive and unreliable proposition to modify the security policy. Moreover, it is virtually impossible to obtain a consolidated view of the safeguards in effect throughout the enterprise to enforce the policy. At the same time, there is increasing pressure on corporate and government executives from consumers, shareholders, and regulators to demonstrate "best practice" in the protection of the information assets of the enterprise and its customers.

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

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

6.1 Requirements

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

– To provide a method for combining individual rules and policies into a single policy set that applies to a particular decision request.

– To provide a method for flexible definition of the procedure by which rules and policies are combined.

– To provide a method for dealing with multiple subjects acting in different capacities.

– To provide a method for basing an authorization decision on attributes of the subject and resource.

– To provide a method for dealing with multi-valued attributes.

– To provide a method for basing an authorization decision on the contents of an information resource.

– To provide a set of logical and mathematical operators on attributes of the subject, resource and environment.

– To provide a method for handling a distributed set of policy components, while abstracting the method for locating, retrieving and authenticating the policy components.

– To provide a method for rapidly identifying the policy that applies to a given action, based upon the values of attributes of the subjects, resource and action.

– To provide an abstraction-layer that insulates the policy-writer from the details of the application environment.

– To provide a method for specifying a set of actions that must be performed in conjunction with policy enforcement.

The motivation behind extensible access control markup language (XACML) is to express these well-established ideas in the field of access control policy using an extension language of XML. The XACML solutions for each of these requirements are discussed in the following clauses.

6.2 Rule and policy combining

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

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

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

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

[b-Hinton] discusses the question of the compatibility of separate policies applicable to the same decision request.

6.3 Combining algorithms

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

– Deny-overrides (ordered and unordered),

– Permit-overrides (ordered and unordered),

– First-applicable, and

– Only-one-applicable.

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

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

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

The "Only-one-applicable" policy-combining algorithm only applies to policies. The result of this combining algorithm ensures that one and only one policy or policy set is applicable by virtue of their

targets. If no policy or policy set applies, then the result is "NotApplicable", but if more than one policy or policy set is applicable, then the result is "Indeterminate". When exactly one policy or policy set is applicable, the result of the combining algorithm is the result of evaluating the single applicable policy or policy set.

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

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

6.4 Multiple subjects

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

6.5 Policies based on subject and resource attributes

Another common requirement is to base an authorization decision on some characteristic of the subject other than its identity. Perhaps, the most common application of this idea is the subject's role [b-RBAC]. XACML provides facilities to support this approach. Attributes of subjects contained in the request context may be identified by the `<AttributeDesignator>` element. This element contains a uniform resource name (URN) that identifies the attribute. Alternatively, the `<AttributeSelector>` element may contain an XLM path language (XPath) expression over the `<Content>` element of the subject to identify a particular subject attribute value by its location in the context (see clause 6.11 for an explanation of context).

XACML provides a standard way to reference the attributes defined in the lightweight directory access protocol (LDAP) series of [IETF RFC 2256], and [IETF RFC 2798]. This is intended to encourage implementers to use standard attribute identifiers for some common subject attributes.

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

6.6 Multi-valued attributes

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

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

6.7 Policies based on resource contents

In many applications, it is required to base an authorization decision on data contained in the information resource to which access is requested. For instance, a common component of privacy

policy is that a person should be allowed to read records for which he or she is the subject. The corresponding policy must contain a reference to the subject identified in the information resource itself.

XACML provides facilities for doing this when the information resource can be represented as an XML document. The <AttributeSelector> element may contain an XPath expression over the <Content> element of the resource to identify data in the information resource to be used in the policy evaluation.

In cases where the information resource is not an XML document, specified attributes of the resource can be referenced, as described in clause 6.5.

6.8 Operators

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

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

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

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

Relationship operators (equality and comparison) are also defined for a number of data-types, including the IETF RFC 822 and ITU-T X.500 name-forms, strings, uniform resource identifiers (URIs), etc.

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

The XACML method of representing functions borrows from [W3C MathML] and from the XQuery 1.0 and XPath 2.0 Functions and Operators specification [W3C XF].

6.9 Policy distribution

In a distributed system, individual policy statements may be written by several policy writers and enforced at several enforcement points. In addition to facilitating the collection and combination of independent policy components, this approach allows policies to be updated as required. XACML policy statements may be distributed in any one of a number of ways. However, XACML does not describe any normative way to do this. Regardless of the means of distribution, PDPs are expected to

confirm, by examining the policy's <Target> element that the policy is applicable to the decision request that it is processing.

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

6.10 Policy indexing

For efficiency of evaluation and ease of management, the overall security policy in force across an enterprise may be expressed as multiple independent policy components. In this case, it is necessary to identify and retrieve the applicable policy statement and verify that it is the correct one for the requested action before evaluating it. This is the purpose of the <Target> element in XACML.

Two approaches are supported:

1) Policy statements may be stored in a database. In this case, PDP should form a database query to retrieve just those policies that are applicable to the set of decision requests to which it expects to respond. Additionally, PDP should evaluate the <Target> element of the retrieved policy or policy set statements as defined by this Recommendation.

2) Alternatively, PDP may be loaded with all available policies and evaluate their <Target> elements in the context of a particular decision request, in order to identify the policies and policy sets that are applicable to that request.

The use of constraints limiting the applicability of a policy is described by [b-Sloman].

6.11 Abstraction layer

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

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

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

In the case where the native request/response format is specified in XML schemas (e.g., a SAML-conformant PEP), the transformation between the native format and the XACML context may be specified in the form of an extensible stylesheet language transformation, see [b-XSLT].

Similarly, in the case where the resource to which access is requested is an XML document, the resource itself may be included in, or referenced by, the request context. Then, through the use of XPath expressions [W3C XPath] in the policy, values in the resource may be included in the policy evaluation.

6.12 Actions performed in conjunction with enforcement

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

6.13 Supplemental information about a decision

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

7 XACML models

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

7.1 Data-flow model

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

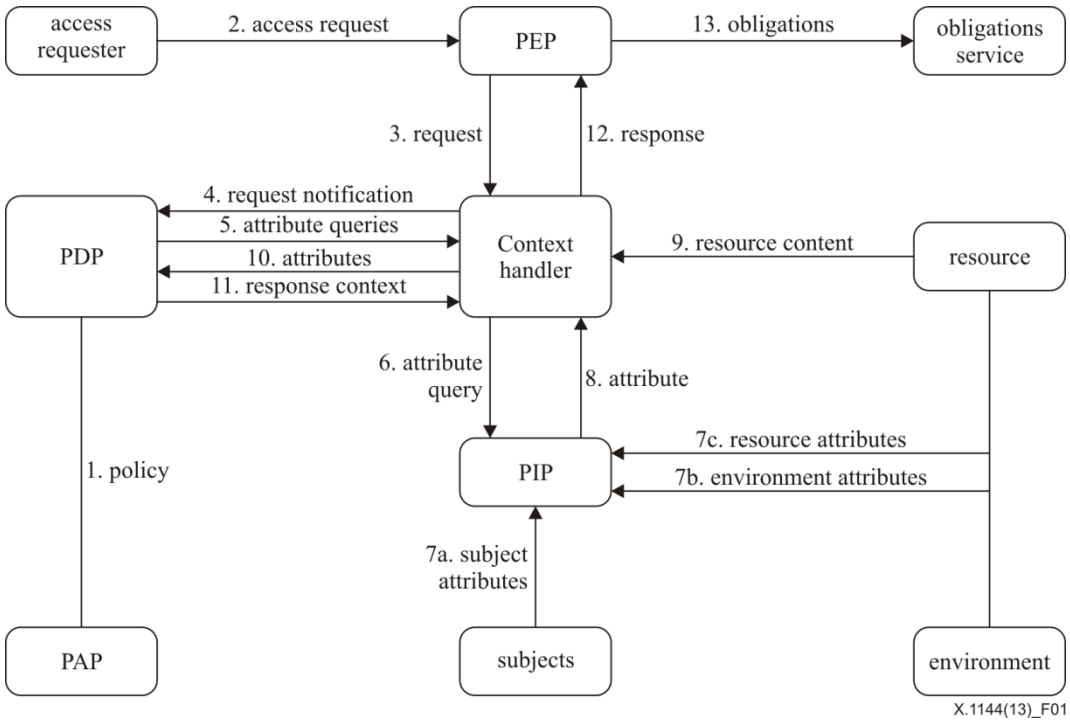

Figure 1 – Data-flow diagram

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

The model operates according to the following steps:

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

2) The ***access*** requester sends a request for ***access*** to ***PEP***.

3) ***PEP*** sends the request for ***access*** to the ***context handler*** in its native request format, optionally including ***attributes*** of the ***subjects***, ***resource***, ***action***, ***environment*** and other categories.

4) The ***context handler*** constructs an XACML request context, optionally adds attributes and sends it to ***PDP***.

5) ***PDP*** requests any additional ***subject***, ***resource***, ***action***, ***environment*** and other categories (not shown) ***attributes*** from the ***context handler***.

6) The ***context handler*** requests the ***attributes*** from ***PIP***.

7) ***PIP*** obtains the requested ***attributes***.

8) ***PIP*** returns the requested ***attributes*** to the ***context handler***.

9) Optionally, the ***context handler*** includes the ***resource*** in the ***context***.

10) The ***context handler*** sends the requested ***attributes*** and (optionally) the ***resource*** to ***PDP***. ***PDP*** evaluates the ***policy***.

11) ***PDP*** returns the response ***context*** (including the ***authorization decision***) to the ***context handler***.

12) The ***context handler*** translates the response ***context*** to the native response format of ***PEP***. The ***context handler*** returns the response to ***PEP***.

13) ***PEP*** fulfils the ***obligations***.

14) (Not shown) If ***access*** is permitted, then ***PEP*** permits ***access*** to the ***resource***; otherwise, it denies ***access***.

7.2 XACML context

XACML is intended to be suitable for a variety of application environments. The core language is insulated from the application environment by the XACML ***context***, as shown in Figure 2, where the scope of this Recommendation is indicated by the shaded area. The XACML ***context*** is defined in XML schema, describing a canonical representation for the inputs and outputs of ***PDP***. ***Attributes*** referenced by an instance of XACML ***policy*** may be in the form of XPath expressions over the `<Content>` elements of the ***context***, or attribute designators that identify the ***attribute*** by its category, identifier, data-type and (optionally) its issuer. Implementations must convert between the ***attribute*** representations in the application environment (e.g., SAML, Java 2 Platform, Standard Edition (J2SE), common object request broker architecture (CORBA), and so on) and the ***attribute*** representations in the XACML ***context***. How this is achieved is outside the scope of this Recommendation. In some cases, such as SAML, this conversion may be accomplished in an automated way through the use of an extensible stylesheet language transformation (XSLT).

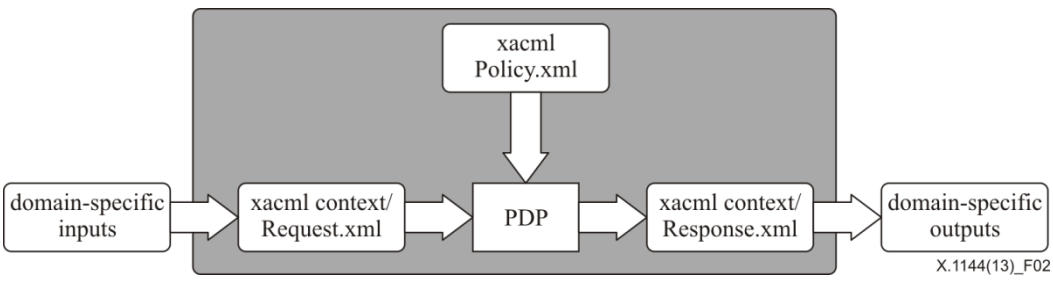

Figure 2 – XACML context

NOTE – ***PDP*** is not required to operate directly on the XACML representation of a ***policy***. It may operate directly on an alternative representation.

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

See clause 10.3.5 for a more detailed discussion of the request ***context.***

7.3 Policy language model

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

Rule;

Policy; and

Policy set.

These are described in the following subclauses.

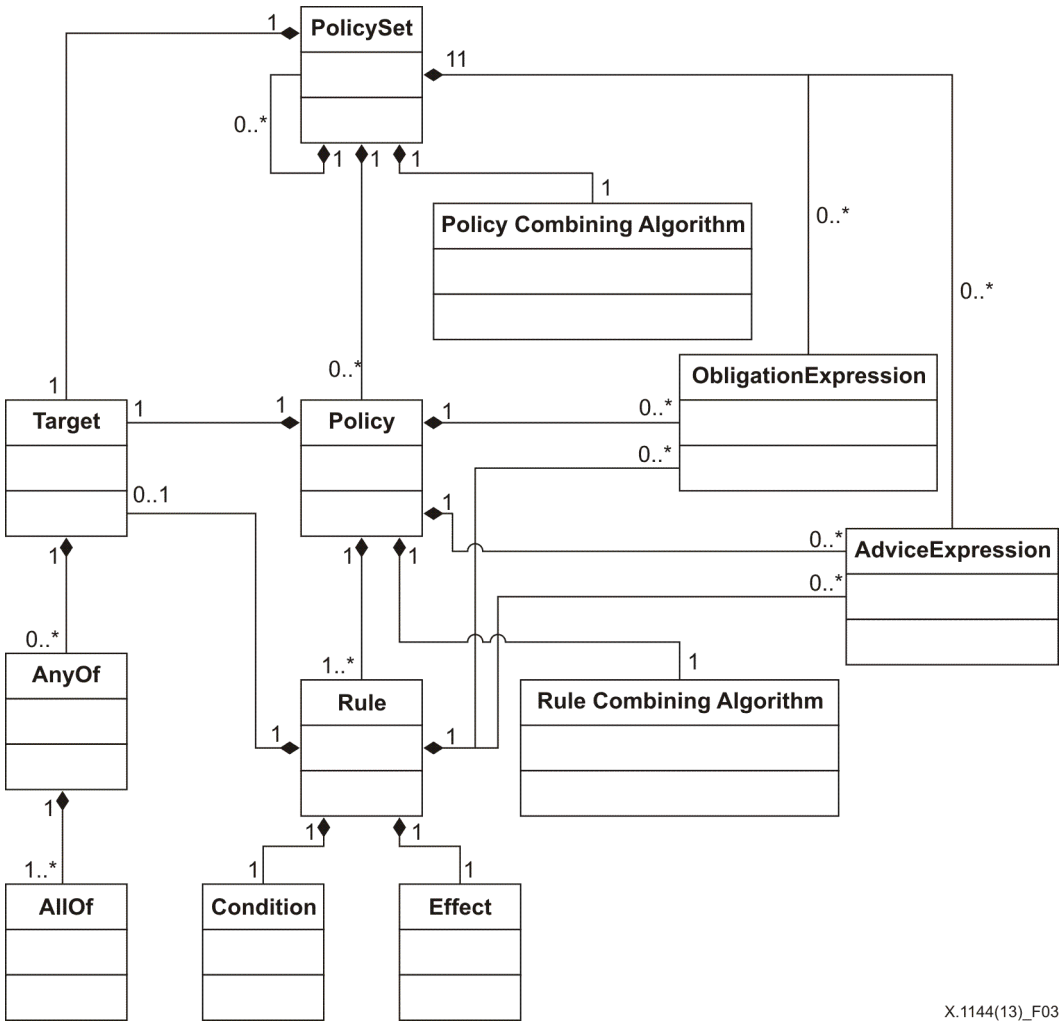

Figure 3 – Policy language model

7.3.1 Rule

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

- a target;

- an effect;
- a condition;
- obligation expressions; and
- advice expressions.

These are discussed in the following subclauses.

7.3.1.1 Rule target

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

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

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

Generally, the name of a node (other than a leaf node) in a structured name-form is also a legal instance of the name-form. So, for instance, the IETF RFC 822 name "med.example.com" is a legal IETF RFC 822 name identifying the set of e-mail addresses hosted by the med.example.com mail server. The XPath value `md:record/md:patient/` is a legal XPath value identifying a node-set in an XML document.

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

In the case of subjects, there is no real entity that corresponds to such a node. As a result, names of this type always refer to the set of subjects subordinate in the name structure to the identified node. Consequently, non-leaf subject names should not be used in equality functions, only in match functions, such as "`urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match`" not "`urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal`" (see clause 11.2.9).

7.3.1.2 Effect

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

7.3.1.3 Condition

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

7.3.1.4 Obligation expressions

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

When PDP evaluates a rule containing obligation expressions, it evaluates the obligation expressions into obligations and returns some of those obligations to PEP in the response context. Clause 10.18 explains which obligations are to be returned.

7.3.1.5 Advice

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

When PDP evaluates a rule containing advice expressions, it evaluates the advice expressions into advice and returns some of those advices to PEP in the response context. Clause 10.18 explains which advice is to be returned. In contrast to obligations, advice may be safely ignored by PEP.

7.3.2 Policy

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

- a target;
- a rule-combining algorithm-identifier;
- a set of rules;
- obligation expressions; and
- advice expressions.

Rules are described above. The remaining components are described in the following subclauses.

7.3.2.1 Policy target

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

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

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

7.3.2.2 Rule-combining algorithm

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

See Annex C for definitions of the normative rule-combining algorithms.

7.3.2.3 Obligation expressions

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

When PDP evaluates a policy containing obligation expressions, it evaluates the obligation expressions into obligations and returns some of those obligations to PEP in the response context. Clause 10.18 explains which obligations are to be returned.

7.3.2.4 Advice

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

When PDP evaluates a policy containing advice expressions, it evaluates the advice expressions into advice and returns some of those advices to PEP in the response context. Clause 10.18 explains which advice is to be returned. In contrast to obligations, advice may be safely ignored by PEP.

7.3.3 Policy set

A policy set comprises four main components:

- a target;
- a policy-combining algorithm-identifier;
- a set of policies;
- obligation expressions; and
- advice expressions.

The target and policy components are described above. The other components are described in the following subclauses.

7.3.3.1 Policy-combining algorithm

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

See Annex C for definitions of the normative policy-combining algorithms.

7.3.3.2 Obligation expressions

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

When PDP evaluates a policy set containing obligations expressions, it evaluates the obligation expressions into obligations and returns certain of those obligations to PEP in its response context. Clause 10.18 explains which obligations are to be returned.

7.3.3.3 Advice expressions

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

When PDP evaluates a policy set containing advice expressions, it evaluates the advice expressions into advice and returns some of those advices to PEP in the response context. Clause 10.18 explains which advice is to be returned. In contrast to obligations, advice may be safely ignored by PEP.

8 Syntax

This clause describes XACML syntax.

8.1 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 clause 10.13 shall be used.

If a <PolicySet> element contains references to other policy sets or policies in the form of uniform resource locators (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.

NOTE 1 – See [b-XACMLAdmin] for an example of the separate administrative policy profile.

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 PDP in making its authorization decision. See clause 10.13.

The <ObligationExpressions> element contains a set of obligation expressions that must be evaluated into obligations by PDP and the resulting obligations must be fulfilled by PEP in conjunction with the authorization decision. If PEP does not understand or cannot fulfil any of the obligations, then it must act according to the PEP bias. See clauses 10.2 and 10.18.

The <AdviceExpressions> element contains a set of advice expressions that must be evaluated into advice by PDP. The resulting advice may be safely ignored by PEP in conjunction with the authorization decision. See clause 10.18.

```
<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>
    <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
    <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
    <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
    <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
</xs:complexType>
```

The <PolicySet> element is of PolicySetType complex type.

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

PolicySetId [Required]

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

`Version` [Required]

The version number of the PolicySet.

`PolicyCombiningAlgId` [Required]

The identifier of the policy-combining algorithm by which the `<PolicySet>`, `<CombinerParameters>`, `<PolicyCombinerParameters>` and `<PolicySetCombinerParameters>` components must be combined. Standard policy-combining algorithms are listed in Annex C. Standard policy-combining algorithm identifiers are listed in clause B.9.

`MaxDelegationDepth` [Optional]

If present, limits the depth of delegation which is authorized by this policy set.

NOTE 2 – See the delegation profile [b-XACMLAdmin].

`<Description>` [Optional]

A free-form description of the policy set.

`<PolicyIssuer>` [Optional]

Attributes of the issuer of the policy set.

`<PolicySetDefaults>` [Optional]

A set of default values applicable to the policy set. The scope of the `<PolicySetDefaults>` element shall be the enclosing policy set.

`<Target>` [Required]

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

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

`<PolicySet>` [Any Number]

A policy set that is included in this policy set.

`<Policy>` [Any Number]

A policy that is included in this policy set.

`<PolicySetIdReference>` [Any Number]

A reference to a policy set that must be included in this policy set. If `<PolicySetIdReference>` is URL, then it may be resolvable.

`<PolicyIdReference>` [Any Number]

A reference to a policy that must be included in this policy set. If the `<PolicyIdReference>` is URL, then it may be resolvable.

`<ObligationExpressions>` [Optional]

Contains the set of `<ObligationExpression>` elements. See clause 10.18 for a description of how the set of ***obligations*** to be returned by ***PDP*** shall be determined.

`<AdviceExpressions>` [Optional]

Contains the set of <AdviceExpression> elements. See clause 10.18 for a description of how the set of ***advice*** to be returned by ***PDP*** shall be determined.

<CombinerParameters> [Optional]

Contains a sequence of <CombinerParameter> elements. The parameters apply to the combining algorithm as such and it is up to the specific combining algorithm to interpret them and adjust its behaviour accordingly.

<PolicyCombinerParameters> [Optional]

Contains a sequence of <CombinerParameter> elements that are associated with a particular <Policy> or <PolicyIdReference> element within the <PolicySet>. It is up to the specific combining algorithm to interpret them and adjust its behaviour accordingly.

<PolicySetCombinerParameters> [Optional]

Contains a sequence of <CombinerParameter> elements that are associated with a particular <PolicySet> or <PolicySetIdReference> element within the <PolicySet>. It is up to the specific combining algorithm to interpret them and adjust its behaviour accordingly.

8.2 Element <Description>

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

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

8.3 Element <PolicyIssuer>

The <PolicyIssuer> element contains ***attributes*** describing the issuer of the ***policy*** or ***policy set***.

NOTE – The use of the ***policy*** issuer element is defined in a separate administration profile [b-XACMLAdmin].

PDP which does not implement the administration profile must report an error or return an "Indeterminate" result if it encounters this element.

```
<xs:element name="PolicyIssuer" type="xacml:PolicyIssuerType"/>
<xs:complexType name="PolicyIssuerType">
  <xs:sequence>
    <xs:element ref="xacml:Content" minOccurs="0"/>
    <xs:element ref="xacml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <PolicyIssuer> element is of PolicyIssuerType complex type.

The <PolicyIssuer> element contains the following elements:

<Content> [Optional]

Free form XML describing the issuer. See clause 8.45.

<Attribute> [Zero to many]

An ***attribute*** of the issuer. See clause 8.46.

8.4 Element <PolicySetDefaults>

The <PolicySetDefaults> element shall specify default values that apply to the <PolicySet> element.

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

<PolicySetDefaults> element is of DefaultsType complex type.

The <PolicySetDefaults> element contains the following elements:

<XPathVersion> [Optional]

Default XPath version.

8.5 Element <XPathVersion>

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

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

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

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

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

8.6 Element <Target>

The <Target> element identifies the set of ***decision requests*** that the parent element is intended to evaluate. The <Target> element shall appear as a child of a <PolicySet> and <Policy> element and may appear as a child of a <Rule> element.

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

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

The <Target> element is of TargetType complex type.

The <Target> element contains the following elements:

<AnyOf> [Zero to Many]

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

8.7 Element <AnyOf>

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

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

The <AnyOf> element is of AnyOfType complex type.

The <AnyOf> element contains the following elements:

<AllOf> [One to Many, Required]

See clause 8.8.

8.8 Element <AllOf>

The <AllOf> element shall contain a ***conjunctive sequence*** of <Match> elements.

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

The <AllOf> element is of AllOfType complex type.

The <AllOf> element contains the following elements:

<Match> [One to Many]

A ***conjunctive sequence*** of individual matches of the ***attributes*** in the request ***context*** and the embedded ***attribute*** values. See clause 8.9.

8.9 Element <Match>

The <Match> element shall identify a set of entities by matching ***attribute*** values in an <Attributes> element of the request ***context*** with the embedded ***attribute*** value.

```
<xs:element name="Match" type="xacml:MatchType"/>
<xs:complexType name="MatchType">
    <xs:sequence>
            <xs:element ref="xacml:AttributeValue"/>
            <xs:choice>
                    <xs:element ref="xacml:AttributeDesignator"/>
                    <xs:element ref="xacml:AttributeSelector"/>
            </xs:choice>
    </xs:sequence>
    <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

The <Match> element is of MatchType complex type.

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

MatchId [Required]

Specifies a matching function. The value of this attribute must be of type xs:anyURI with legal values documented in clause 10.6.

<AttributeValue> [Required]

Embedded ***attribute*** value.

`<AttributeDesignator>` [Required choice]

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

`<AttributeSelector>` [Required choice]

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

8.10 Element <PolicySetIdReference>

The `<PolicySetIdReference>` element shall be used to reference a `<PolicySet>` element by id. If `<PolicySetIdReference>` is URL, then it may be resolvable to the `<PolicySet>` element. However, the mechanism for resolving a ***policy set*** reference to the corresponding ***policy set*** is outside the scope of this Recommendation.

```
<xs:element name="PolicySetIdReference" type="xacml:IdReferenceType"/>
<xs:complexType name="IdReferenceType">
    <xs:simpleContent>
        <xs:extension base="xs:anyURI">
            <xs:attribute name="xacml:Version"
        type="xacml:VersionMatchType" use="optional"/>
            <xs:attribute name="xacml:EarliestVersion"
        type="xacml:VersionMatchType" use="optional"/>
            <xs:attribute name="xacml:LatestVersion"
        type="xacml:VersionMatchType" use="optional"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

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

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

`Version` [Optional]

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

`EarliestVersion` [Optional]

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

`LatestVersion` [Optional]

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

The matching operation is defined in clause 8.13. Any combination of these attributes may be present in a `<PolicySetIdReference>`. The referenced ***policy set*** must match all expressions. If none of these attributes is present, then any version of the ***policy set*** is acceptable. In the case that more than one matching version can be obtained, then the most recent one should be used.

8.11 Element <PolicyIdReference>

The `<PolicyIdReference>` element shall be used to reference a `<Policy>` element by id. If `<PolicyIdReference>` is URL, then it may be resolvable to the `<Policy>` element. However, the mechanism for resolving a ***policy*** reference to the corresponding ***policy*** is outside the scope of this Recommendation.

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

Element `<PolicyIdReference>` is of `xacml:IdReferenceType` complex type (see clause 8.10).

8.12 Simple type VersionType

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

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

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

8.13 Simple type VersionMatchType

Elements of this type shall contain a restricted regular expression matching a version number (see clause 8.12). The expression shall match versions of a referenced ***policy*** or ***policy set*** that is acceptable for inclusion in the referencing ***policy*** or ***policy set***.

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

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

8.14 Element <Policy>

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

A `<Policy>` element may be evaluated, in which case the evaluation procedure defined in clause 10.12 shall be used.

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

A `<Policy>` element may contain a `<PolicyIssuer>` element. The interpretation of the `<PolicyIssuer>` element is explained in the separate administrative ***policy*** profile.

NOTE 1 – See [b-XACMLAdmin] for an example.

The `<Target>` element defines the applicability of the `<Policy>` element to a set of ***decision requests***. If the `<Target>` element within the `<Policy>` element matches the request ***context***, then the `<Policy>` element may be used by ***PDP*** in making its ***authorization decision***. See clause 10.12.

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

Rules included in the `<Policy>` element must be combined by the algorithm specified by the `RuleCombiningAlgId` attribute.

The `<ObligationExpressions>` element contains a set of ***obligation*** expressions that must be evaluated into ***obligations*** by ***PDP*** and the resulting ***obligations*** must be fulfilled by ***PEP*** in conjunction with the ***authorization decision***. If ***PEP*** does not understand, or cannot fulfil, any of the ***obligations***, then it must act according to the PEP bias. See clauses 10.2 and 10.18.

The <AdviceExpressions> element contains a set of ***advice*** expressions that must be evaluated into ***advice*** by ***PDP***. The resulting ***advice*** may be safely ignored by ***PEP*** in conjunction with the ***authorization decision***. See clause 10.18.

```
<xs:element name="Policy" type="xacml:PolicyType"/>
<xs:complexType name="PolicyType">
    <xs:sequence>
            <xs:element ref="xacml:Description" minOccurs="0"/>
            <xs:element ref="xacml:PolicyIssuer" minOccurs="0"/>
            <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
            <xs:element ref="xacml:Target"/>
            <xs:choice maxOccurs="unbounded">
                    <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
                    <xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>
                    <xs:element ref="xacml:VariableDefinition"/>
                    <xs:element ref="xacml:Rule"/>
            </xs:choice>
            <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
            <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
    <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
    <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
    <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
</xs:complexType>
```

The <Policy> element is of PolicyType complex type.

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

PolicyId [Required]

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

Version [Required]

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

RuleCombiningAlgId [Required]

> The identifier of the ***rule-combining algorithm*** by which the <Policy>, <CombinerParameters> and <RuleCombinerParameters> components must be combined. Standard ***rule-combining algorithms*** are listed in Annex C. Standard ***rule-combining algorithm*** identifiers are listed in Annex B.9.

MaxDelegationDepth [Optional]

> If present, limits the depth of delegation which is authorized by this ***policy***.

NOTE 2 – See the delegation profile [b-XACMLAdmin].

<Description> [Optional]

> A free-form description of the ***policy***. See clause 8.2.

<PolicyIssuer> [Optional]

> ***Attributes*** of the ***issuer*** of the ***policy***.

<PolicyDefaults> [Optional]

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

<CombinerParameters> [Optional]

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

<RuleCombinerParameters> [Optional]

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

<Target> [Required]

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

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

<VariableDefinition> [Any Number]

Common function definitions that can be referenced from anywhere in a ***rule*** where an expression can be found.

<Rule> [Any Number]

A sequence of ***rules*** that must be combined according to the RuleCombiningAlgId attribute. ***Rules*** whose <Target> elements and conditions match the ***decision request*** must be considered. ***Rules*** whose <Target> elements or conditions do not match the ***decision request*** shall be ignored.

<ObligationExpressions> [Optional]

A ***conjunctive sequence*** of ***obligation*** expressions which must be evaluated into ***obligations*** by PDP. The corresponding ***obligations*** must be fulfilled by ***PEP*** in conjunction with the ***authorization decision***. See clause 10.18 for a description of how the set of ***obligations*** to be returned by ***PDP*** shall be determined. See clause 10.2 for enforcement of ***obligations***. [b-OASIS]

<AdviceExpressions> [Optional]

A ***conjunctive sequence*** of ***advice*** expressions which must evaluated into ***advice*** by ***PDP***. The corresponding ***advice*** provides supplementary information to ***PEP*** in conjunction with the ***authorization decision***. See clause 10.18 for a description of how the set of ***advice*** to be returned by ***PDP*** shall be determined.

8.15 Element <PolicyDefaults>

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

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

<PolicyDefaults> element is of DefaultsType complex type.

The <PolicyDefaults> element contains the following elements:

<XPathVersion> [Optional]

Default XPath version.

8.16 Element <CombinerParameters>

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

If multiple <CombinerParameters> elements occur within the same ***policy*** or ***policy set***, they shall be considered equal to one <CombinerParameters> element containing the concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned <CombinerParameters> elements, such that the order of occurrence of the <CombinerParameters> elements is preserved in the concatenation of the <CombinerParameter> elements. [b-OASIS]

Note that none of the combining algorithms specified in this Recommendation is parameterized.

```
<xs:element name="CombinerParameters" type="xacml:CombinerParametersType"/>
<xs:complexType name="CombinerParametersType">
    <xs:sequence>
            <xs:element ref="xacml:CombinerParameter" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

The <CombinerParameters> element is of CombinerParametersType complex type.

The <CombinerParameters> element contains the following elements:

<CombinerParameter> [Any Number]

A single parameter. See clause 8.17.

Support for the <CombinerParameters> element is optional.

8.17 Element <CombinerParameter>

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

```
<xs:element name="CombinerParameter" type="xacml:CombinerParameterType"/>
<xs:complexType name="CombinerParameterType">
    <xs:sequence>
            <xs:element ref="xacml:AttributeValue"/>
    </xs:sequence>
    <xs:attribute name="ParameterName" type="xs:string" use="required"/>
</xs:complexType>
```

The <CombinerParameter> element is of CombinerParameterType complex type.

The <CombinerParameter> element contains the following attributes:

ParameterName [Required]

The identifier of the parameter.

<AttributeValue> [Required]

The value of the parameter.

Support for the <CombinerParameter> element is optional.

8.18 Element <RuleCombinerParameters>

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

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 [b-OASIS].

Note that none of the ***rule-combining algorithms*** specified in this Recommendation is parameterized.

```
<xs:element name="RuleCombinerParameters" type="xacml:RuleCombinerParametersType"/>
<xs:complexType name="RuleCombinerParametersType">
    <xs:complexContent>
            <xs:extension base="xacml:CombinerParametersType">
                    <xs:attribute name="RuleIdRef" type="xs:string"
            use="required"/>
            </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The <RuleCombinerParameters> element contains the following attribute:

RuleIdRef [Required]

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

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

8.19 Element <PolicyCombinerParameters>

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

Each <PolicyCombinerParameters> element must be associated with a ***policy*** contained within the same ***policy set***. If multiple <PolicyCombinerParameters> elements reference the same ***policy***, they shall be considered equal to one <PolicyCombinerParameters> element containing the concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned <PolicyCombinerParameters> elements, such that the order of occurrence of the <PolicyCominberParameters> elements is preserved in the concatenation of the <CombinerParameter> elements.

Note that none of the ***policy-combining algorithms*** specified in this Recommendation is parameterized.

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

The <PolicyCombinerParameters> element is of PolicyCombinerParametersType complex type.

The <PolicyCombinerParameters> element contains the following attribute:

PolicyIdRef [Required]

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

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

8.20 Element <PolicySetCombinerParameters>

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

Each <PolicySetCombinerParameters> element must be associated with a ***policy set*** contained within the same ***policy set***. If multiple <PolicySetCombinerParameters> elements reference the same ***policy set***, they shall be considered equal to one <PolicySetCombinerParameters> element containing the concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned <PolicySetCombinerParameters> elements, such that the order of occurrence of the <PolicySetCominberParameters> elements is preserved in the concatenation of the <CombinerParameter> elements.

Note that none of the ***policy-combining algorithms*** specified in this Recommendation is parameterized.

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

The <PolicySetCombinerParameters> element is of PolicySetCombinerParametersType complex type.

The <PolicySetCombinerParameters> element contains the following attribute:

PolicySetIdRef [Required]

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

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

8.21 Element <Rule>

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

A <Rule> element may be evaluated, in which case the evaluation procedure defined in clause 10.10 shall be used.

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

The <Rule> element is of RuleType complex type.

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

RuleId [Required]

A string identifying this ***rule***.

`Effect` [Required]

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

`<Description>` [Optional]

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

`<Target>` [Optional]

Identifies the set of ***decision requests*** that the `<Rule>` element is intended to evaluate. If this element is omitted, then the ***target*** for the `<Rule>` shall be defined by the `<Target>` element of the enclosing `<Policy>` element. See clause 10.7 for details.

`<Condition>` [Optional]

A ***predicate*** that must be satisfied for the ***rule*** to be assigned its `Effect` value.

`<ObligationExpressions>` [Optional]

A ***conjunctive sequence*** of ***obligation*** expressions which must be evaluated into ***obligations*** by PDP. The corresponding ***obligations*** must be fulfilled by ***PEP*** in conjunction with the ***authorization decision***. See clause 10.18 for a description of how the set of ***obligations*** to be returned by ***PDP*** shall be determined. See clause 10.2 for enforcement of ***obligations***.

`<AdviceExpressions>` [Optional]

A ***conjunctive sequence*** of ***advice*** expressions which must evaluated into ***advice*** by ***PDP***. The corresponding ***advice*** provides supplementary information to ***PEP*** in conjunction with the ***authorization decision***. See clause 10.18 for a description of how the set of ***advice*** to be returned by ***PDP*** shall be determined.

8.22 Simple type EffectType

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

```
<xs:simpleType name="EffectType">
    <xs:restriction base="xs:string">
            <xs:enumeration value="Permit"/>
            <xs:enumeration value="Deny"/>
    </xs:restriction>
</xs:simpleType>
```

8.23 Element <VariableDefinition>

The `<VariableDefinition>` element shall be used to define a value that can be referenced by a `<VariableReference>` element. The name supplied for its `VariableId` attribute shall NOT occur in the `VariableId` attribute of any other `<VariableDefinition>` element within the encompassing ***policy***. The `<VariableDefinition>` element may contain undefined `<VariableReference>` elements, but if it does, a corresponding `<VariableDefinition>` element must be defined later in the encompassing ***policy***. `<VariableDefinition>` elements may be grouped together or may be placed close to the reference in the encompassing ***policy***. There may be zero or more references to each `<VariableDefinition>` element.

```
<xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>
<xs:complexType name="VariableDefinitionType">
    <xs:sequence>
            <xs:element ref="xacml:Expression"/>
    </xs:sequence>
    <xs:attribute name="VariableId" type="xs:string" use="required"/>
</xs:complexType>
```

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

<Expression> [Required]

Any element of ExpressionType complex type.

VariableId [Required]

The name of the variable definition.

8.24 Element <VariableReference>

The <VariableReference> element is used to reference a value defined within the same encompassing <Policy> element. The <VariableReference> element shall refer to the <VariableDefinition> element by ***identifier equality*** on the value of their respective VariableId attributes. One and only one <VariableDefinition> must exist within the same encompassing <Policy> element to which the <VariableReference> refers. There may be zero or more <VariableReference> elements that refer to the same <VariableDefinition> element.

```
<xs:element name="VariableReference" type="xacml:VariableReferenceType" substitutionGroup="xacml:Expression"/>
<xs:complexType name="VariableReferenceType">
    <xs:complexContent>
                <xs:extension base="xacml:ExpressionType">
                        <xs:attribute name="VariableId" type="xs:string"
            use="required"/>
                </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

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

The <VariableReference> element has the following attribute:

VariableId [Required]

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

8.25 Element <Expression>

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

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

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

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

8.26 Element <Condition>

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

```
<xs:element name="Condition" type="xacml:ConditionType"/>
<xs:complexType name="ConditionType">
    <xs:sequence>
                <xs:element ref="xacml:Expression"/>
    </xs:sequence>
</xs:complexType>
```

The <Condition> contains one <Expression> element, with the restriction that the <Expression> return data-type must be "http://www.w3.org/2001/XMLSchema#boolean". Evaluation of the <Condition> element is described in clause 10.9.

8.27 Element <Apply>

The <Apply> element denotes application of a function to its arguments, thus encoding a function call. The <Apply> element can be applied to any combination of the members of the <Expression> element substitution group. See clause 8.25.

```
<xs:element name="Apply" type="xacml:ApplyType" substitutionGroup="xacml:Expression"/>
<xs:complexType name="ApplyType">
    <xs:complexContent>
                <xs:extension base="xacml:ExpressionType">
                        <xs:sequence>
                                <xs:element ref="xacml:Description" minOccurs="0"/>
                                <xs:element ref="xacml:Expression" minOccurs="0"
                    maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attribute name="FunctionId" type="xs:anyURI"
              use="required"/>
                </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The <Apply> element is of ApplyType complex type.

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

FunctionId [Required]

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

<Description> [Optional]

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

<Expression> [Optional]

Arguments to the function, which may include other functions.

8.28 Element <Function>

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

```
<xs:element name="Function" type="xacml:FunctionType" substitutionGroup="xacml:Expression"/>
<xs:complexType name="FunctionType">
    <xs:complexContent>
                <xs:extension base="xacml:ExpressionType">
                        <xs:attribute name="FunctionId" type="xs:anyURI"
              use="required"/>
                </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The <Function> element is of FunctionType complex type.

The <Function> element contains the following attribute:

FunctionId [Required]

The identifier of the function.

8.29 Element <AttributeDesignator>

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

The <AttributeDesignator> element shall return a ***bag*** containing all the ***attribute*** values that are matched by the ***named attribute***. In the event that no matching ***attribute*** is present in the ***context***, the MustBePresent attribute governs whether this element returns an empty ***bag*** or "Indeterminate". See clause 10.3.5.

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

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

```
<xs:element name="AttributeDesignator" type="xacml:AttributeDesignatorType"
substitutionGroup="xacml:Expression"/>
<xs:complexType name="AttributeDesignatorType">
    <xs:complexContent>
            <xs:extension base="xacml:ExpressionType">
                    <xs:attribute name="Category" type="xs:anyURI"
            use="required"/>
                    <xs:attribute name="AttributeId" type="xs:anyURI"
            use="required"/>
                    <xs:attribute name="DataType" type="xs:anyURI"
            use="required"/>
                    <xs:attribute name="Issuer" type="xs:string" use="optional"/>
                    <xs:attribute name="MustBePresent" type="xs:boolean"
            use="required"/>
              </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

A ***named attribute*** shall match an ***attribute*** if the values of their respective Category, AttributeId, DataType and Issuer attributes match. The attribute designator's Category must match, by ***identifier equality***, the Category of the <Attributes> element in which the ***attribute*** is present. The attribute designator's AttributeId must match, by ***identifier equality***, the AttributeId of the attribute. The attribute designator's DataType must match, by ***identifier equality***, the DataType of the same ***attribute***.

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

The <AttributeDesignatorType> contains the following attributes:

Category [Required]

This attribute shall specify the Category with which to match the ***attribute***.

AttributeId [Required]

This attribute shall specify the AttributeId with which to match the ***attribute***.

DataType [Required]

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

Issuer [Optional]

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

`MustBePresent` [Required]

This attribute governs whether the element returns "Indeterminate" or an empty ***bag*** in the event the ***named attribute*** is absent from the request ***context***. See clause 10.3.5. See also clauses 10.19.2 and 10.19.3.

8.30 Element <AttributeSelector>

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

See clause 10.3.7 for details of `<AttributeSelector>` evaluation.

```
<xs:element name="AttributeSelector" type="xacml:AttributeSelectorType" substitutionGroup="xacml:Expression"/>
<xs:complexType name="AttributeSelectorType">
    <xs:complexContent>
                <xs:extension base="xacml:ExpressionType">
            <xs:attribute name="Category" type="xs:anyURI"
                use="required"/>
                        <xs:attribute name="ContextSelectorId" type="xs:anyURI"
                use="optional"/>
                        <xs:attribute name="Path" type="xs:string"
                use="required"/>
                        <xs:attribute name="DataType" type="xs:anyURI"
                use="required"/>
                        <xs:attribute name="MustBePresent" type="xs:boolean"
                use="required"/>
                 </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

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

The `<AttributeSelector>` element has the following attributes:

`Category` [Required]

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

`ContextSelectorId` [Optional]

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

`Path` [Required]

This attribute shall contain an `XPath` expression to be evaluated against the specified XML content. See clause 10.3.7 for details of the XPath evaluation during `<AttributeSelector>` processing. The namespace context for the value of the Path attribute is given by the [in-scope namespaces] [INFOSET] of the `<AttributeSelector>` element [b-OASIS].

`DataType` [Required]

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

`MustBePresent` [Required]

This attribute governs whether the element returns "Indeterminate" or an empty ***bag*** in the event that the attributes category specified by the `Category` attribute does not exist in the request context, or the attributes category does exist but it does not have a `<Content>` child element, or the `<Content>` element does exist but the XPath expression selects no node. See clause 10.3.5. See also clauses 10.19.2 and 10.19.3 [b-OASIS].

8.31 Element <AttributeValue>

The `<AttributeValue>` element shall contain a literal ***attribute*** value.

```
<xs:element name="AttributeValue" type="xacml:AttributeValueType" substitutionGroup="xacml:Expression"/>
<xs:complexType name="AttributeValueType" mixed="true">
    <xs:complexContent mixed="true">
            <xs:extension base="xacml:ExpressionType">
                    <xs:sequence>
                            <xs:any namespace="##any" processContents="lax"
                minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="DataType" type="xs:anyURI"
            use="required"/>
                    <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

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

The `<AttributeValue>` element has the following attributes:

`DataType` [Required]

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

8.32 Element <Obligations>

The `<Obligations>` element shall contain a set of `<Obligation>` elements.

```
<xs:element name="Obligations" type="xacml:ObligationsType"/>
<xs:complexType name="ObligationsType">
    <xs:sequence>
            <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

The `<Obligations>` element is of `ObligationsType` complex type.

The `<Obligations>` element contains the following element:

`<Obligation>` [One to Many]

A sequence of ***obligations***. See clause 8.34.

8.33 Element <AssociatedAdvice>

The `<AssociatedAdvice>` element shall contain a set of `<Advice>` elements.

```
<xs:element name="AssociatedAdvice" type="xacml:AssociatedAdviceType"/>
<xs:complexType name="AssociatedAdviceType">
    <xs:sequence>
            <xs:element ref="xacml:Advice" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

The `<AssociatedAdvice>` element is of `AssociatedAdviceType` complex type.

The `<AssociatedAdvice>` element contains the following element:

`<Advice>` [One to Many]

A sequence of ***advice***. See clause 8.35.

8.34 Element <Obligation>

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

```
<xs:element name="Obligation" type="xacml:ObligationType"/>
<xs:complexType name="ObligationType">
    <xs:sequence>
            <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

The `<Obligation>` element is of `ObligationType` complex type. See clause 10.18 for a description of how the set of ***obligations*** to be returned by ***PDP*** is determined.

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

`ObligationId` [Required]

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

`<AttributeAssignment>` [Optional]

Obligation arguments assignment. The values of the ***obligation*** arguments shall be interpreted by ***PEP***.

8.35 Element <Advice>

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

```
<xs:element name="Advice" type="xacml:AdviceType"/>
<xs:complexType name="AdviceType">
    <xs:sequence>
            <xs:element ref="xacml:AttributeAssignment" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

The `<Advice>` element is of `AdviceType` complex type. See clause 10.18 for a description of how the set of ***advice*** to be returned by ***PDP*** is determined.

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

`AdviceId` [Required]

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

`<AttributeAssignment>` [Optional]

Advice arguments assignment. The values of the ***advice*** arguments may be interpreted by ***PEP***.

8.36 Element <AttributeAssignment>

The `<AttributeAssignment>` element is used for including arguments in ***obligation*** and ***advice*** expressions. It shall contain an `AttributeId` and the corresponding ***attribute*** value, by extending the `AttributeValueType` type definition. The `<AttributeAssignment>` element may be used in any way that is consistent with the schema syntax, which is a sequence of `<xs:any>` elements. The value specified shall be understood by ***PEP***, but it is not further specified by XACML. See clause 10.18. Clause I.2.4.3 provides a number of examples of arguments included in ***obligation*** expressions.

```
<xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
<xs:complexType name="AttributeAssignmentType" mixed="true">
    <xs:complexContent>
            <xs:extension base="xacml:AttributeValueType">
                    <xs:attribute name="AttributeId" type="xs:anyURI"
            use="required"/>
                    <xs:attribute name="Category" type="xs:anyURI"
            use="optional"/>
                    <xs:attribute name="Issuer" type="xs:string" use="optional"/>
            </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

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

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

`AttributeId` [Required]

The ***attribute*** Identifier.

`Category` [Optional]

An optional category of the ***attribute***. If this attribute is missing, the ***attribute*** has no category. ***PEP*** shall interpret the significance and meaning of any `Category` attribute.

NOTE 1 – An expected use of the category is to disambiguate ***attributes*** which are relayed from the request.

`Issuer` [Optional]

An optional issuer of the ***attribute***. If this attribute is missing, the ***attribute*** has no issuer. ***PEP*** shall interpret the significance and meaning of any `Issuer` attribute.

NOTE 2 – An expected use of the issuer is to disambiguate ***attributes*** which are relayed from the request.

8.37 Element <ObligationExpressions>

The `<ObligationExpressions>` element shall contain a set of `<ObligationExpression>` elements.

```
<xs:element name="ObligationExpressions"
   type="xacml:ObligationExpressionsType"/>
<xs:complexType name="ObligationExpressionsType">
    <xs:sequence>
            <xs:element ref="xacml:ObligationExpression" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

The `<ObligationExpressions>` element is of `ObligationExpressionsType` complex type.

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

`<ObligationExpression>` [One to Many]

A sequence of ***obligations*** expressions. See clause 8.39.

8.38 Element <AdviceExpressions>

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

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

The `<AdviceExpressions>` element is of `AdviceExpressionsType` complex type.

The <AdviceExpressions> element contains the following element:

<AdviceExpression> [One to Many]

A sequence of ***advice*** expressions. See clause 8.40.

8.39 Element <ObligationExpression>

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

```
<xs:element name="ObligationExpression"
    type="xacml:ObligationExpressionType"/>
<xs:complexType name="ObligationExpressionType">
 <xs:sequence>
   <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"
       maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
 <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
</xs:complexType>
```

The <ObligationExpression> element is of ObligationExpressionType complex type. See clause 10.18 for a description of how the set of ***obligations*** to be returned by ***PDP*** is determined.

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

ObligationId [Required]

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

FulfillOn [Required]

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

<AttributeAssignmentExpression> [Optional]

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

8.40 Element <AdviceExpression>

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

```
<xs:element name="AdviceExpression" type="xacml:AdviceExpressionType"/>
<xs:complexType name="AdviceExpressionType">
    <xs:sequence>
            <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>
    <xs:attribute name="AppliesTo" type="xacml:EffectType" use="required"/>
</xs:complexType>
```

The <AdviceExpression> element is of AdviceExpressionType complex type. See clause 10.18 for a description of how the set of ***advice*** to be returned by ***PDP*** is determined.

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

AdviceId [Required]

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

AppliesTo [Required]

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

<AttributeAssignmentExpression> [Optional]

Advice arguments in the form of expressions. The expressions shall be evaluated by PDP to constant <AttributeValue> elements or ***bags***, which shall be the attribute assignments in the <Advice> returned to PEP. If an <AttributeAssignmentExpression> evaluates to an atomic ***attribute*** value, then there must be one resulting <AttributeAssignment> which must contain this single ***attribute*** value. If the <AttributeAssignmentExpression> evaluates to a ***bag***, then there must be a resulting <AttributeAssignment> for each of the values in the ***bag***. If the ***bag*** is empty, there shall be no <AttributeAssignment> from this <AttributeAssignmentExpression>. The values of the ***advice*** arguments may be interpreted by ***PEP***.

8.41 Element <AttributeAssignmentExpression>

The <AttributeAssignmentExpression> element is used for including arguments in ***obligations*** and ***advice***. It shall contain an AttributeId and an expression which shall by evaluated into the corresponding ***attribute*** value. The value specified shall be understood by ***PEP***, but it is not further specified by XACML. See clause 10.18. Clause I.2.4.3 provides a number of examples of arguments included in ***obligations***.

```
<xs:element name="AttributeAssignmentExpression"
   type="xacml:AttributeAssignmentExpressionType"/>
<xs:complexType name="AttributeAssignmentExpressionType">
 <xs:sequence>
   <xs:element ref="xacml:Expression"/>
 </xs:sequence>
 <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
 <xs:attribute name="Category" type="xs:anyURI" use="optional"/>
 <xs:attribute name="Issuer" type="xs:string" use="optional"/>
</xs:complexType>
```

The <AttributeAssignmentExpression> element is of AttributeAssignmentExpressionType complex type.

The <AttributeAssignmentExpression> element contains the following attributes:

<Expression> [Required]

The expression which evaluates to a constant ***attribute*** value or a bag of zero or more attribute values. See clause 8.25.

AttributeId [Required]

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

Category [Optional]

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

`Issuer` [Optional]

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

8.42 Element <Request>

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

```
<xs:element name="Request" type="xacml:RequestType"/>
<xs:complexType name="RequestType">
    <xs:sequence>
            <xs:element ref="xacml:RequestDefaults" minOccurs="0"/>
            <xs:element ref="xacml:Attributes" maxOccurs="unbounded"/>
            <xs:element ref="xacml:MultiRequests" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="ReturnPolicyIdList" type="xs:boolean" use="required"/>
    <xs:attribute name="CombinedDecision" type="xs:boolean" use="required" />
</xs:complexType>
```

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

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

`ReturnPolicyIdList` [Required]

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

`CombinedDecision` [Required]

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

NOTE 1 – The use of this attribute is specified in [b-Multi].

`<RequestDefaults>` [Optional]

Contains default values for the request, such as XPath version. See clause 8.43.

`<Attributes>` [One to Many]

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

The `<Request>` element contains `<Attributes>` elements. There may be multiple `<Attributes>` elements with the same `Category` attribute if ***PDP*** implements the multiple decision profile [b-OASIS].

NOTE 2 – See [b-Multi] for examples of multiple decision profile.

Under other conditions, it is a syntax error if there are multiple <Attributes> elements with the same Category (see clause 10.19.2 for error codes).

<MultiRequests> [Optional]

Lists multiple ***request contexts*** by references to the <Attributes> elements. Implementation of this element is optional. The semantics of this element is defined in [b-Multi]. If the implementation does not implement this element, it must return an "Indeterminate" result if it encounters this element. See clause 8.50.

8.43 Element <RequestDefaults>

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

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

<RequestDefaults> element is of RequestDefaultsType complex type.

The <RequestDefaults> element contains the following elements:

<XPathVersion> [Optional]

Default XPath version for XPath expressions occurring in the request.

8.44 Element <Attributes>

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

```
<xs:element name="Attributes" type="xacml:AttributesType"/>
<xs:complexType name="AttributesType">
    <xs:sequence>
            <xs:element ref="xacml:Content" minOccurs="0"/>
            <xs:element ref="xacml:Attribute" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="Category" type="xs:anyURI" use="required"/>
    <xs:attribute ref="xml:id" use="optional"/>
</xs:complexType>
```

The <Attributes> element is of AttributesType complex type.

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

Category [Required]

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

xml:id [Optional]

This attribute provides a unique identifier for this <Attributes> element.

NOTE – See [W3C XMLid] for more examples. It is primarily intended to be referenced in multiple requests. See [b-Multi] for more examples.

<Content> [Optional]

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

<Attribute> [Any Number]

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

8.45 Element <Content>

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

```
<xs:element name="Content" type="xacml:ContentType"/>
<xs:complexType name="ContentType" mixed="true">
    <xs:sequence>
            <xs:any namespace="##any" processContents="lax"/>
    </xs:sequence>
</xs:complexType>
```

The <Content> element is of ContentType complex type.

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

8.46 Element <Attribute>

The <Attribute> element is the central abstraction of the request ***context***. It contains ***attribute*** metadata and one or more ***attribute*** values. The ***attribute*** metadata comprises the ***attribute*** identifier and the ***attribute*** issuer. <AttributeDesignator> elements in the ***policy*** may refer to ***attributes*** by means of this metadata.

```
<xs:element name="Attribute" type="xacml:AttributeType"/>
<xs:complexType name="AttributeType">
    <xs:sequence>
            <xs:element ref="xacml:AttributeValue" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
    <xs:attribute name="Issuer" type="xs:string" use="optional"/>
    <xs:attribute name="IncludeInResult" type="xs:boolean" use="required"/>
</xs:complexType>
```

The <Attribute> element is of AttributeType complex type.

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

AttributeId [Required]

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

Issuer [Optional]

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

IncludeInResult [Default: false]

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

<AttributeValue> [One to Many]

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

8.47 Element <Response>

The <Response> element is an abstraction layer used by the ***policy*** language. Any proprietary system using this Recommendation must transform an XACML ***context*** <Response> element into the form of its ***authorization decision***.

The <Response> element encapsulates the ***authorization decision*** produced by ***PDP***. It includes a sequence of one or more results, with one <Result> element per requested ***resource***. Multiple results may be returned by some implementations, in particular those that support the XACML profile for requests for multiple resources.

NOTE – Support for multiple results is optional. For examples see [b-Multi].

```
<xs:element name="Response" type="xacml:ResponseType"/>
<xs:complexType name="ResponseType">
    <xs:sequence>
            <xs:element ref="xacml:Result" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

The <Response> element is of ResponseType complex type.

The <Response> element contains the following elements:

<Result> [One to Many]

An ***authorization decision*** result. See clause 8.48.

8.48 Element <Result>

The <Result> element represents an ***authorization decision*** result. It may include a set of ***obligations*** that must be fulfilled by ***PEP***. If ***PEP*** does not understand or cannot fulfil an ***obligation***, then the action of PEP is determined by its bias, see clause 10.2. It may include a set of ***advice*** with supplemental information which may be safely ignored by ***PEP***.

```
<xs:complexType name="ResultType">
    <xs:sequence>
            <xs:element ref="xacml:Decision"/>
            <xs:element ref="xacml:Status" minOccurs="0"/>
            <xs:element ref="xacml:Obligations" minOccurs="0"/>
            <xs:element ref="xacml:AssociatedAdvice" minOccurs="0"/>
            <xs:element ref="xacml:Attributes" minOccurs="0"
        maxOccurs="unbounded"/>
            <xs:element ref="xacml:PolicyIdentifierList" minOccurs="0"/>        </xs:sequence>
</xs:complexType>
```

The <Result> element is of ResultType complex type.

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

<Decision> [Required]

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

<Status> [Optional]

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

<Obligations> [Optional]

A list of ***obligations*** that must be fulfilled by ***PEP***. If ***PEP*** does not understand or cannot fulfil an ***obligation***, then the action of PEP is determined by its bias, see clause 10.2. See clause 10.18 for a description of how the set of ***obligations*** to be returned by ***PDP*** is determined.

<AssociatedAdvice> [Optional]

A list of ***advice*** that provides supplemental information to ***PEP***. If ***PEP*** does not understand an ***advice***, PEP may safely ignore the ***advice***. See clause 10.18 for a description of how the set of ***advice*** to be returned by ***PDP*** is determined.

<Attributes> [Optional]

A list of ***attributes*** that were part of the request. The choice of which ***attributes*** are included here is made with the IncludeInResult attribute of the <Attribute> elements of the request. See clause 8.46.

<PolicyIdentifierList> [Optional]

If the ReturnPolicyIdList attribute in the <Request> is true (see clause 8.42), ***PDP*** that implements this optional feature must return a list which includes the identifiers of all ***policies*** which were found to be fully applicable, whether or not the <Effect> (after rule combining) was the same or different from the <Decision>. The list MAY include the identifiers of other policies which are currently in force, as long as no policies required for the decision are omitted. A ***PDP*** MAY satisfy this requirement by including all policies currently in force, or by including all policies which were evaluated in making the decision, or by including all policies which did not evaluate to "NotApplicable", or by any other algorithm which does not omit any policies which contributed to the decision. However, a decision which returns "NotApplicable" MUST return an empty list [b-OASIS].

8.49 Element <PolicyIdentifierList>

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

```
<xs:element name="PolicyIdentifierList"
  type="xacml:PolicyIdentifierListType"/>
<xs:complexType name="PolicyIdentifierListType">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="xacml:PolicyIdReference"/>
            <xs:element ref="xacml:PolicySetIdReference"/>
      </xs:choice>
</xs:complexType>
```

The <PolicyIdentifierList> element is of PolicyIdentifierListType complex type.

The <PolicyIdentifierList> element contains the following elements.

<PolicyIdReference> [Any number]

The identifier and version of a ***policy*** which was applicable to the request. See clause 8.11. The <PolicyIdReference> element must use the Version attribute to specify the version and must not use the LatestVersion or EarliestVersion attributes.

<PolicySetIdReference> [Any number]

The identifier and version of a ***policy set*** which was applicable to the request. See clause 8.10. The <PolicySetIdReference> element must use the Version attribute to specify the version and must not use the LatestVersion or EarliestVersion attributes.

8.50 Element <MultiRequests>

The <MultiRequests> element contains a list of requests by reference to <Attributes> elements in the enclosing <Request> element.

NOTE – The semantics of this element are defined in [b-Multi]. Support for this element is optional.

If an implementation does not support this element, but receives it, the implementation must generate an "Indeterminate" response.

```
<xs:element name="MultiRequests" type="xacml:MultiRequestsType"/>
<xs:complexType name="MultiRequestsType">
    <xs:sequence>
            <xs:element ref="xacml:RequestReference" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

The `<MultiRequests>` element contains the following elements.

`<RequestReference>` [one to many]

Defines a request instance by reference to the `<Attributes>` elements in the enclosing `<Request>` element. See clause 8.51.

8.51 Element <RequestReference>

The `<RequestReference>` element defines an instance of a request in terms of references to the `<Attributes>` elements.

NOTE – The semantics of this element are defined in [b-Multi]. Support for this element is optional.

```
<xs:element name="RequestReference" type="xacml:RequestReference "/>
<xs:complexType name="RequestReferenceType">
    <xs:sequence>
            <xs:element ref="xacml:AttributesReference" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

The `<RequestReference>` element contains the following elements.

`<AttributesReference>` [one to many]

A reference to an `<Attributes>` element in the enclosing `<Request>` element. See clause 8.52.

8.52 Element <AttributesReference>

The `<AttributesReference>` element makes a reference to an `<Attributes>` element.

NOTE – The meaning of this element is defined in [b-Multi]. Support for this element is optional.

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

The `<AttributesReference>` element contains the following attributes.

`ReferenceId` [required]

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

8.53 Element <Decision>

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

```
<xs:element name="Decision" type="xacml:DecisionType"/>
<xs:simpleType name="DecisionType">
    <xs:restriction base="xs:string">
            <xs:enumeration value="Permit"/>
            <xs:enumeration value="Deny"/>
            <xs:enumeration value="Indeterminate"/>
            <xs:enumeration value="NotApplicable"/>
    </xs:restriction>
</xs:simpleType>
```

The <Decision> element is of DecisionType simple type.

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

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

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

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

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

8.54 Element <Status>

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

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

The <Status> element is of StatusType complex type.

The <Status> element contains the following elements:

<StatusCode> [Required]

Status code.

<StatusMessage> [Optional]

A status message describing the status code.

<StatusDetail> [Optional]

Additional status information.

8.55 Element <StatusCode>

The <StatusCode> element contains a major status code value and an optional recursive series of minor status codes [b-OASIS].

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

The <StatusCode> element is of StatusCodeType complex type.

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

Value [Required]

See clause B.8 for a list of values.

<StatusCode> [Any Number]

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

8.56 Element <StatusMessage>

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

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

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

8.57 Element <StatusDetail>

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

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

The <StatusDetail> element is of StatusDetailType complex type.

The <StatusDetail> element allows arbitrary XML content.

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

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

PDP must not return a <StatusDetail> element in conjunction with the "ok" status value.

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

PDP may choose not to return any <StatusDetail> information or may choose to return a <StatusDetail> element containing one or more <MissingAttributeDetail> elements.

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

PDP must not return a <StatusDetail> element in conjunction with the "syntax-error" status value. A syntax error may represent either a problem with the ***policy*** being used or with the request ***context***. ***PDP*** may return a <StatusMessage> describing the problem.

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

PDP must not return a <StatusDetail> element in conjunction with the "processing-error" status value. This status code indicates an internal problem in ***PDP***. For security reasons, ***PDP*** may choose to return no further information to ***PEP***. In the case of a divide-by-zero error or other computational error, ***PDP*** may return a <StatusMessage> describing the nature of the error.

8.58 Element <MissingAttributeDetail>

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

```
<xs:element name="MissingAttributeDetail" type="xacml:MissingAttributeDetailType"/>
<xs:complexType name="MissingAttributeDetailType">
<xs:sequence>
            <xs:element ref="xacml:AttributeValue" minOccurs="0"
        maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="Category" type="xs:anyURI" use="required"/>
<xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
<xs:attribute name="DataType" type="xs:anyURI" use="required"/>
    <xs:attribute name="Issuer" type="xs:string" use="optional"/>
</xs:complexType>
```

The <MissingAttributeDetail> element is of MissingAttributeDetailType complex type.

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

`<AttributeValue>` [Optional]

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

`Category` [Required]

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

`AttributeId` [Required]

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

`DataType` [Required]

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

`Issuer` [Optional]

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

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

9 XPath 2.0 definitions

The XPath 2.0 specification leaves a number of aspects of behaviour implementation defined. This clause defines how XPath 2.0 shall behave when hosted in XACML.

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

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

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

1) Support for additional user-defined or implementation-defined types is implementation defined. It is recommended that implementations of XACML do not define any additional types and it is recommended that users of XACML do not make use of any additional types.

2) Some typed values in the data model are undefined. Attempting to access an undefined property is always an error. Behaviour in these cases is implementation defined and the host language is responsible for determining the result. An XPath error causes an XACML "Indeterminate" value in the element where the XPath error occurs. The StatusCode value shall be "`urn:oasis:names:tc:xacml:1.0:status:processing-error`". Implementations may provide additional details about the error in the response or by some other means.

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

1) The destination of the trace output is implementation defined. XACML leaves this implementation defined.

2) For `xs:integer` operations, implementations that support limited-precision integer operations must either raise an error [`err:FOAR0002`] or provide an implementation-defined mechanism that allows users to choose between raising an error and returning a result that is modulo the largest representable integer value. XACML leaves this implementation defined. If an implementation chooses to raise an error, the `StatusCode` value shall be "`urn:oasis:names:tc:xacml:1.0:status:processing-error`". Implementations may provide additional details about the error in the response or by some other means.

3) For `xs:decimal` values the number of digits of precision returned by the numeric operators is implementation defined. XACML leaves this implementation defined.

4) If the number of digits in the result of a numeric operation exceeds the number of digits that the implementation supports, the result is truncated or rounded in an implementation-defined manner. XACML leaves this implementation defined.

5) It is implementation defined which version of Unicode is supported. XACML leaves this implementation defined. It is recommended that the latest version is used.

6) For `fn:normalize-unicode`, conforming implementations must support normalization form "NFC" and may support normalization forms "NFD", "NFKC", "NFKD", "FULLY-NORMALIZED". They may also support other normalization forms with implementation-defined semantics. XACML leaves this implementation defined.

7) The ability to decompose strings into collation units suitable for substring matching is an implementation defined property of a collation. XACML leaves this implementation defined.

8) All minimally conforming processors must support year values with a minimum of four digits (i.e., YYYY) and a minimum fractional second precision of one millisecond or three digits (i.e., s.sss). However, conforming processors may set larger implementation-defined limits on the maximum number of digits they support in these two situations. XACML leaves this implementation defined, and it is recommended that users of XACML do not expect greater limits and precision.

9) The result of casting a string to xs:decimal, when the resulting value is not too large or too small but nevertheless has too many decimal digits to be accurately represented, is implementation defined. XACML leaves this implementation defined.

10) Various aspects of the processing provided by `fn:doc` are implementation defined. Implementations may provide external configuration options that allow any aspect of the processing to be controlled by the user. XACML leaves this implementation defined.

11) The manner in which implementations provide options to weaken the stable characteristic of `fn:collection` and `fn:doc` are implementation defined. XACML leaves this implementation defined.

10 Functional requirements

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

NOTE – In each case an implementation is conformant as long as it produces the same result as is specified here, regardless of how and in what order the implementation behaves internally.

10.1 Unicode issues

10.1.1 Normalization

In Unicode, some equivalent characters can be represented by more than one different Unicode character sequence.

NOTE – See [b-CMF] for examples.

The process of converting Unicode strings into equivalent character sequences is called "normalization" [b-UAX15]. Some operations, such as string comparison, are sensitive to normalization. An operation is normalization sensitive if its output(s) are different depending on the state of normalization of the input(s); if the output(s) are textual, they are deemed different only if they would remain different were they to be normalized.

For more information on normalization, see [b-CM].

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

10.1.2 Version of Unicode

The version of Unicode used by XACML is implementation defined. It is recommended that the latest version be used. In addition, note Unicode security issues in clause III.3.

10.2 Policy enforcement point

This clause describes the requirements for PEP.

An application functions in the role of PEP if it guards access to a set of resources and asks PDP for an authorization decision. PEP must abide by the authorization decision as described in one of the following subclauses.

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

10.2.1 Base PEP

If the decision is "Permit", then PEP shall permit access. If obligations accompany the decision, then PEP shall permit access only if it understands and it can and will discharge those obligations.

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

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

If the decision is "Indeterminate", then PEP's behaviour is undefined.

10.2.2 Deny-biased PEP

If the decision is "Permit", then PEP shall permit access. If obligations accompany the decision, then PEP shall permit access only if it understands and it can and will discharge those obligations.

All other decisions shall result in the denial of access.

NOTE – Other actions, e.g., consultation of additional PDPs, reformulation/resubmission of the decision request, etc., are not prohibited.

10.2.3 Permit-biased PEP

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

All other decisions shall result in the permission of access.

NOTE – Other actions, e.g., consultation of additional PDPs, reformulation/resubmission of the decision request, etc., are not prohibited.

10.3 Attribute evaluation

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

10.3.1 Structured attributes

`<AttributeValue>` elements may contain an instance of a structured XML data-type, for example, `<ds:KeyInfo>`. This Recommendation supports several ways for comparing the contents of such elements.

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

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

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

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

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

See also clause 10.3.

10.3.2 Attribute bags

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

The `<AttributeSelector>` element uses an XPath expression to specify the selection of data from free form XML. The result of an XPath expression is termed a node-set, which contains all the nodes from the XML content that match the predicate in the XPath expression. Based on the various

indexing functions provided in the XPath specification, it shall be implied that a resultant node-set is the collection of the matching nodes. XACML also defines the <AttributeDesignator> element to have the same matching methodology for attributes in the XACML request context.

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

10.3.3 Multivalued attributes

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

10.3.4 Attribute matching

A named attribute includes specific criteria with which to match attributes in the context. An attribute specifies a Category, AttributeId and DataType, and a named attribute also specifies the Issuer. A named attribute shall match an attribute if the values of their respective Category, AttributeId, DataType and optional Issuer attributes match. The Category of the named attribute must match, by identifier equality, the Category of the corresponding context attribute. The AttributeId of the named attribute must match, by identifier equality, the AttributeId of the corresponding context attribute. The DataType of the named attribute must match, by identifier equality, the DataType of the corresponding context attribute. If Issuer is supplied in the named attribute, then it must match, using the urn:oasis:names:tc:xacml:1.0:function:string-equal function, the Issuer of the corresponding context attribute. If Issuer is not supplied in the named attribute, then the matching of the context attribute to the named attribute shall be governed by AttributeId and DataType alone, regardless of the presence, absence, or actual value of Issuer in the corresponding context attribute. In the case of an attribute selector, the matching of the attribute to the named attribute shall be governed by the XPath expression and DataType.

10.3.5 Attribute retrieval

PDP shall request the values of attributes in the request context from the context handler. The context handler may also add attributes to the request context without PDP requesting them. PDP shall reference the attributes as if they were in a physical request context document, but the context handler is responsible for obtaining and supplying the requested values by whatever means it deems appropriate, including by retrieving them from one or more policy information points. The context handler shall return the values of attributes that match the attribute designator or attribute selector and form them into a bag of values with the specified data-type. If no attributes from the request context match, then the attribute shall be considered missing. If the attribute is missing, then MustBePresent governs whether the attribute designator or attribute selector returns an empty bag or an "Indeterminate" result. If MustBePresent is "False" (default value), then a missing attribute shall result in an empty bag. If MustBePresent is "True", then a missing attribute shall result in "Indeterminate". This "Indeterminate" result shall be handled in accordance with the specification of the encompassing expressions, rules, policies and policy sets. If the result is "Indeterminate", then the AttributeId, DataType and Issuer of the attribute may be listed in the authorization decision as described in clause 10.17. However, PDP may choose not to return such information for security reasons.

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

10.3.6 Environment attributes

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

10.3.7 AttributeSelector evaluation

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

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

1) If the attributes category given by the `Catagory` attribute is not found or does not have a `<Content>` child element, then the return value is either "Indeterminate" or an empty ***bag*** as determined by the `MustBePresent` attribute; otherwise, construct an XML data structure suitable for xpath processing from the `<Content>` element in the attributes category given by the `Category` attribute. The data structure shall be constructed so that the document node of this structure contains a single document element which corresponds to the single child element of the `<Content>` element. The constructed data structure shall be equivalent to one that would result from parsing a stand-alone XML document consisting of the contents of the `<Content>` element (including any comment and processing-instruction markup). Namespace declarations which are not "visibly utilized", as defined by [W3C EXC-C14N], may not be present and must not be utilized by the XPath expression in step 3. The data structure must meet the requirements of the applicable xpath version [b-OASIS].

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

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

4) If the data-type is a primitive data-type, convert the text value of each selected node to the desired data-type, as specified in the `DataType` attribute. Each value shall be constructed using the appropriate constructor function from [W3C XF] Section 5 listed below, corresponding to the specified data type.

```
xs:string()
xs:boolean()
xs:integer()
xs:double()
xs:dateTime()
xs:date()
xs:time()
xs:hexBinary()
```

xs:base64Binary()

xs:anyURI()

xs:yearMonthDuration()

xs:dayTimeDuration()

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

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

10.4 Expression evaluation

This Recommendation specifies expressions in terms of the elements listed below, of which the <Apply> and <Condition> elements recursively compose greater expressions. Valid expressions shall be type correct, which means that the types of each of the elements contained within <Apply> elements shall agree with the respective argument types of the function that is named by the FunctionId attribute. The resultant type of the <Apply> element shall be the resultant type of the function, which may be narrowed to a primitive data-type, or a bag of a primitive data-type, by type-unification. This Recommendation defines an evaluation result of "Indeterminate", which is said to be the result of an invalid expression, or an operational error occurring during the evaluation of the expression.

This Recommendation defines these elements to be in the substitution group of the <Expression> element:

- <xacml:AttributeValue>
- <xacml:AttributeDesignator>
- <xacml:AttributeSelector>
- <xacml:Apply>
- <xacml:Function>
- <xacml:VariableReference>

10.5 Arithmetic evaluation

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

flags – all set to 0;

trap-enablers – all set to 0 ([IEEE 854], §7) with the exception of the "division-by-zero" trap enabler, which shall be set to 1;

precision – is set to the designated double precision;

rounding – is set to round-half-even ([IEEE 854], §4.1).

10.6 Match evaluation

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

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

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

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

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

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

```
<Match
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
       John.*
    </AttributeValue>
    <AttributeDesignator
          Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Match>
```

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

```
<Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
    <Function
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match"/>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        John.*
    </AttributeValue>
    <AttributeDesignator
         Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
         AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
         DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Apply>
```

10.7 Target evaluation

An empty target matches any request. Otherwise, the target value shall be "Match" if all the <AnyOf> specified in the target match values in the request context. Otherwise, if any one of the <AnyOf> specified in the target is "No Match", then the target shall be "No Match". Otherwise, the target shall be "Indeterminate". The target match table is shown in Table 1.

Table 1 – Target match table

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

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

Table 2 – <AnyOf> match table

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

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

The <AllOf> table is shown in Table 3.

Table 3 – <AllOf> match table

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

10.8 VariableReference evaluation

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

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

A variable reference containing circular references is invalid. PDP must detect circular references either at policy loading time or during runtime evaluation. If PDP detects a circular reference during runtime, the variable reference evaluates to "Indeterminate" with status code `urn:oasis:names:tc:xacml:1.0:status:processing-error`.

10.9 Condition evaluation

The condition value shall be "True" if the `<Condition>` element is absent, or if it evaluates to "True". Its value shall be "False" if the `<Condition>` element evaluates to "False". The condition value shall be "Indeterminate", if the expression contained in the `<Condition>` element evaluates to "Indeterminate."

10.10 Extended "indeterminate"

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

- "Indeterminate{D}": an "Indeterminate" from a policy or rule which could have evaluated to "Deny", but not "Permit".
- "Indeterminate{P}": an "Indeterminate" from a policy or rule which could have evaluated to "Permit", but not "Deny".
- "Indeterminate{DP}": an "Indeterminate" from a policy or rule which could have evaluated to "Deny" or "Permit".

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

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

Tables 4 to 7 below define how extended "Indeterminate" values are produced during `Rule`, `Policy` and `PolicySet` evaluation.

10.11 Rule evaluation

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

Table 4 – Rule truth table

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"

10.12 Policy evaluation

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

The policy truth table is shown in Table 5.

Table 5 – Policy truth table

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

Note that none of the rule-combining algorithms defined by this Recommendation takes parameters. However, non-standard combining algorithms may take parameters. In such a case, the values of these parameters associated with the rules must be taken into account when evaluating the policy. The parameters and their types should be defined in the specification of the combining algorithm. If the implementation supports combiner parameters and if combiner parameters are present in a policy, then the parameter values must be supplied to the combining algorithm implementation.

10.13 Policy set evaluation

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

The policy set truth table is shown in Table 6.

Table 6 – Policy set truth table

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

Note that none of the policy-combining algorithms defined by this Recommendation takes parameters. However, non-standard combining algorithms may take parameters. In such a case, the values of these parameters associated with the policies must be taken into account when evaluating the policy set. The parameters and their types should be defined in the specification of the combining algorithm. If the implementation supports combiner parameters and if combiner parameters are present in a policy, then the parameter values must be supplied to the combining algorithm implementation.

10.14 Policy and policy set value for i "Indeterminate" target

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

Table 7 – Value of a policy or policy set when the target is "Indeterminate"

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}"

10.15 PolicySetIdReference and PolicyIdReference evaluation

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

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

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

10.16 Hierarchical resources

It is often the case that a resource is organized as a hierarchy (e.g., file system, XML document). XACML provides several optional mechanisms for supporting hierarchical resources.

NOTE – Examples of hierarchical resources are described in the XACML Profile for Hierarchical Resources [b-Hier] and in the XACML Profile for Requests for Multiple Resources [b-Multi].

10.17 Authorization decision

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

PDP must evaluate the policy set as specified in clauses 8 and 10. PDP must return a response context, with one `<Decision>` element of value "Permit", "Deny", "Indeterminate" or "NotApplicable".

If PDP cannot make a decision, then an "Indeterminate" `<Decision>` element shall be returned.

10.18 Obligations and advice

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

As a consequence of this procedure, no obligations or advice shall be returned to PEP if the rule, policies, or policy sets from which they are drawn are not evaluated, or if their evaluated result is

"Indeterminate" or "NotApplicable", or if the decision resulting from evaluating the rule, policy, or policy set does not match the decision resulting from evaluating an enclosing policy set.

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

Also see clause 10.2.

10.19 Exception handling

XACML specifies behaviour for PDP in the following situations.

10.19.1 Unsupported functionality

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

10.19.2 Syntax and type errors

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

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

10.19.3 Missing attributes

The absence of matching attributes in the request context for any of the attribute designators attribute or selectors that are found in the policy will result in an enclosing `<AllOf>` element to return a value of "Indeterminate", if the designator or selector has the `MustBePresent` XML attribute set to true, as described in clauses 8.29 and 8.30 and may result in a `<Decision>` element containing the "Indeterminate" value. If, in this case, a status code is supplied, then the value

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

shall be used to indicate that more information is needed in order for a definitive decision to be rendered. In this case, the `<Status>` element may list the names and data-types of any attributes that are needed by PDP to refine its decision (see clause 8.58). PEP may resubmit a refined request context in response to a `<Decision>` element contents of "Indeterminate" with a status code of

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

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

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

it must not list the names and data-types of any attribute for which values were supplied in the original request.

NOTE – This requirement forces PDP to eventually return an authorization decision of "Permit", "Deny", or "Indeterminate" with some other status code, in response to successively-refined requests.

10.20 Identifier equality

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

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

The content of the element `<XPathVersion>`.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

11 Conformance

XACML defines a number of functions that have a somewhat special application, therefore they are not required to be supported in an implementation that claims to conform with this Recommendation.

This clause lists those portions of this Recommendation that must be included in an implementation of a PDP that claims to conform with XACML v3.0.

NOTE – "M" means mandatory-to-implement. "O" means optional.

11.1 Schema elements

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

Element name	M/O
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

11.2 Identifier prefixes

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

11.3 Algorithms

The implementation must include the rule- and policy-combining algorithms associated with the 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-applicable	M
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

11.4 Status codes

Implementation support for the <StatusCode> element is optional, but if the element is supported, then the following status codes must be supported and must be used in the way specified by XACML.

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

11.5 Attributes

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

11.6 Identifiers

The implementation must use the attributes associated with the following identifiers in the way defined by XACML. This requirement pertains primarily to implementations of PAP or PEP that uses XACML, since the semantics of the attributes are transparent to PDP [b-OASIS].

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:2.0:resource:target-namespace	O
urn:oasis:names:tc:xacml:1.0:action:action-id	O
urn:oasis:names:tc:xacml:1.0:action:action-namespace	O
urn:oasis:names:tc:xacml:1.0:action:implied-action	O

11.7 Data-types

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

11.8 Functions

The implementation must properly process those functions associated with the identifiers marked with "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

Function	M/O
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

Function	M/O
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

Function	M/O
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

Function	M/O
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

11.9 Identifiers planned for future deprecation

These identifiers are associated with previous versions of XACML and newer alternatives exist in this Recommendation. They are planned to be deprecated at some unspecified point in the future. It is recommended that these identifiers not be used in new polices and requests.

The implementation must properly process those features associated with the identifiers marked with "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

Function	M/O
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

Annex A

Data-types and functions

(This annex forms an integral part of this Recommendation.)

A.1 Introduction

This clause specifies the data-types and functions used in XACML to create predicates for conditions and target matches.

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

A.2 Data-types

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

- `http://www.w3.org/2001/XMLSchema#string`
- `http://www.w3.org/2001/XMLSchema#boolean`
- `http://www.w3.org/2001/XMLSchema#integer`
- `http://www.w3.org/2001/XMLSchema#double`
- `http://www.w3.org/2001/XMLSchema#time`
- `http://www.w3.org/2001/XMLSchema#date`
- `http://www.w3.org/2001/XMLSchema#dateTime`
- `http://www.w3.org/2001/XMLSchema#anyURI`
- `http://www.w3.org/2001/XMLSchema#hexBinary`
- `http://www.w3.org/2001/XMLSchema#base64Binary`
- `http://www.w3.org/2001/XMLSchema#dayTimeDuration`
- `http://www.w3.org/2001/XMLSchema#yearMonthDuration`
- `urn:oasis:names:tc:xacml:1.0:data-type:x500Name`
- `urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name`
- `urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`
- `urn:oasis:names:tc:xacml:2.0:data-type:dnsName`
- `urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression`

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

XACML PDP shall be capable of converting string representations into various primitive data-types. For doubles, XACML shall use the conversions described in [IEEE 754].

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

"`urn:oasis:names:tc:xacml:1.0:data-type:x500Name`",

"`urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name`"

"`urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`", and

"urn:oasis:names:tc:xacml:2.0:data-type:dnsName"

These types appear in several standard applications, such as transport layer security/secure socket layer (TLS/SSL) and electronic mail.

ITU-T X.500 directory name

The "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" primitive type represents ITU-T X.520 Distinguished Name. The valid syntax for such a name is described in [IETF RFC 4514], "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names".

IETF RFC 822 name

The "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" primitive type represents an electronic mail address. The valid syntax for such a name is described in [IETF RFC 5321], section 4.1.2, Command Argument Syntax, under the term "Mailbox".

IP address

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

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

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

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

Domain name system (DNS) name

The "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" primitive type represents a domain name System (DNS) host name, with optional port or port range. The syntax shall be:

dnsName = hostname [":" portrange]

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

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

portrange = portnumber | "-"portnumber | portnumber"-"[portnumber]

where "portnumber" is a decimal port number. If the port number is of the form "-x", where "x" is a port number, then the range is all ports numbered "x" and below. If the port number is of the form "x-", then the range is all ports numbered "x" and above.

XPath expression

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

The XPath expression must be evaluated in a context which is equivalent of a stand-alone XML document with the only child of the <Content> element as the document element. Namespace declarations which are not "visibly utilized", as defined by [W3C EXC-C14N], may not be present and must not be utilized by the XPath expression. The context node of the XPath expression is the document node of this stand-alone document.

A.3 Functions

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

NOTE – In each case an implementation is conformant as long as it produces the same result as is specified here, regardless of how and in what order the implementation behaves internally.

A.3.1 Equality predicates

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

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

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall return "True" if and only if the value of both of its arguments are of equal length and each string is determined to be equal. Otherwise, it shall return "False". The comparison shall use Unicode codepoint collation, as defined for the identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [W3C XF].

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

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

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

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

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

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

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

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

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

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#date" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall perform its evaluation according to the "op:date-equal" function [W3C XF], section 10.4.9.

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

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall perform its evaluation according to the "op:time-equal" function [W3C XF], section 10.4.12.

- urn:oasis:names:tc:xacml:1.0:function:dateTime-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall perform its evaluation according to the "op:dateTime-equal" function [W3C XF], section 10.4.6.

- urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dayTimeDuration" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation according to the "op:duration-equal" function [W3C XF], section 10.4.5.

NOTE – The lexical representation of each argument must be converted to a value expressed in fractional seconds [W3C XF], section 10.3.2.

- urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#yearMonthDuration" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation according to the "op:duration-equal" function [W3C XF], section 10.4.5.

NOTE – The lexical representation of each argument must be converted to a value expressed in fractional seconds [W3C XF], section 10.3.2.

- urn:oasis:names:tc:xacml:1.0:function:anyURI-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#anyURI" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall convert the arguments to strings with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI and return "True" if and only if the values of the two arguments are equal on a codepoint-by-codepoint basis.

- urn:oasis:names:tc:xacml:1.0:function:x500Name-equal

This function shall take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if each relative distinguished name (RDN) in the two arguments matches. Otherwise, it shall return "False". Two RDNs shall be said to match if and only if the result of the following operations is "True".

1) Normalize the two arguments according to [IETF RFC 2253], "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names".
2) If any RDN contains multiple attributeTypeAndValue pairs, re-order the AttributeValuePairs in that RDN in ascending order when compared as octet strings (described in clause 11.6 of [ITU-T X.690], "Set-of components").
3) Compare RDNs using the rules in [IETF RFC 5280], "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", section 4.1.2.4 "Issuer".

- urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal

This function shall take two arguments of data-type "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the two arguments are equal. Otherwise, it shall return "False". An IETF RFC 822 name consists of a local-part followed by "@" followed by a domain-part. The local-part is case-sensitive, while the domain-part (which is usually a DNS hostname) is not case-sensitive. Perform the following operations:

1) Normalize the domain-part of each argument to lower case.
2) Compare the expressions by applying the function "urn:oasis:names:tc:xacml:1.0:function:string-equal" to the normalized arguments.

- urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#hexBinary" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if the octet sequences represented by the value of both arguments have equal length and are equal in a conjunctive, point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function. Otherwise, it shall return "False". The conversion from the string representation to an octet sequence shall be as specified in [W3C XS], part 2, section 3.2.15.

- urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#base64Binary" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if the octet sequences represented by the value of both arguments have equal length and are equal in a conjunctive, point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function. Otherwise, it shall return "False". The conversion from the string representation to an octet sequence shall be as specified in [W3C XS], part 2, section 3.2.16.

A.3.2 Arithmetic functions

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

- urn:oasis:names:tc:xacml:1.0:function:integer-add

This function must accept two or more arguments.

- urn:oasis:names:tc:xacml:1.0:function:double-add

This function must accept two or more arguments.

- urn:oasis:names:tc:xacml:1.0:function:integer-subtract

The result is the second argument subtracted from the first argument.

- urn:oasis:names:tc:xacml:1.0:function:double-subtract

The result is the second argument subtracted from the first argument.

- urn:oasis:names:tc:xacml:1.0:function:integer-multiply

This function must accept two or more arguments.

- urn:oasis:names:tc:xacml:1.0:function:double-multiply

This function must accept two or more arguments.

- urn:oasis:names:tc:xacml:1.0:function:integer-divide

The result is the first argument divided by the second argument.

- urn:oasis:names:tc:xacml:1.0:function:double-divide

The result is the first argument divided by the second argument.

- urn:oasis:names:tc:xacml:1.0:function:integer-mod

The result is the remainder of the first argument divided by the second argument.

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

- urn:oasis:names:tc:xacml:1.0:function:integer-abs
- urn:oasis:names:tc:xacml:1.0:function:double-abs
- urn:oasis:names:tc:xacml:1.0:function:round
- urn:oasis:names:tc:xacml:1.0:function:floor

A.3.3 String conversion functions

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

- urn:oasis:names:tc:xacml:1.0:function:string-normalize-space

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string" and shall normalize the value by stripping off all leading and trailing whitespace characters. The whitespace characters are defined in the metasymbol S (Production 3) of [W3C XML].

- urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string" and shall normalize the value by converting each upper case character to its lower case equivalent. Case mapping shall be done as specified for the fn:lower-case function in [XF] with no tailoring for particular languages or environments.

A.3.4 Numeric data-type conversion functions

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

- urn:oasis:names:tc:xacml:1.0:function:double-to-integer

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#double" and shall truncate its numeric value to a whole number and return an element of data-type "http://www.w3.org/2001/XMLSchema#integer".

- urn:oasis:names:tc:xacml:1.0:function:integer-to-double

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#integer" and shall promote its value to an element of data-type "http://www.w3.org/2001/XMLSchema#double" with the same numeric value. If the integer argument is outside the range which can be represented by a double, the result shall be

"Indeterminate", with the status code "urn:oasis:names:tc:xacml:1.0:status:processing-error".

A.3.5 Logical functions

This clause contains the specification for logical functions that operate on arguments of data-type "http://www.w3.org/2001/XMLSchema#boolean".

- urn:oasis:names:tc:xacml:1.0:function:or

This function shall return "False" if it has no arguments and shall return "True" if at least one of its arguments evaluates to "True". The order of evaluation shall be from first argument to last. The evaluation shall stop with a result of "True" if any argument evaluates to "True", leaving the rest of the arguments unevaluated.

- urn:oasis:names:tc:xacml:1.0:function:and

This function shall return "True" if it has no arguments and shall return "False" if one of its arguments evaluates to "False". The order of evaluation shall be from first argument to last. The evaluation shall stop with a result of "False" if any argument evaluates to "False", leaving the rest of the arguments unevaluated.

- urn:oasis:names:tc:xacml:1.0:function:n-of

The first argument to this function shall be of data-type http://www.w3.org/2001/XMLSchema#integer. The remaining arguments shall be of data-type http://www.w3.org/2001/XMLSchema#boolean. The first argument specifies the minimum number of the remaining arguments that must evaluate to "True" for the expression to be considered "True". If the first argument is 0, the result shall be "True". If the number of arguments after the first one is less than the value of the first argument, then the expression shall result in "Indeterminate". The order of evaluation shall be: first evaluate the integer value, and then evaluate each subsequent argument. The evaluation shall stop and return "True" if the specified number of arguments evaluate to "True". The evaluation of arguments shall stop if it is determined that evaluating the remaining arguments will not satisfy the requirement.

- urn:oasis:names:tc:xacml:1.0:function:not

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

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

A.3.6 Numeric comparison functions

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

- urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
- urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal
- urn:oasis:names:tc:xacml:1.0:function:integer-less-than
- urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal
- urn:oasis:names:tc:xacml:1.0:function:double-greater-than
- urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal

- urn:oasis:names:tc:xacml:1.0:function:double-less-than
- urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal

A.3.7 Date and time arithmetic functions

These functions perform arithmetic operations with date and time.

- urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration

This function shall take two arguments, the first shall be of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and the second shall be of data-type "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It shall return a result of "http://www.w3.org/2001/XMLSchema#dateTime". This function shall return the value by adding the second argument to the first argument according to the specification of adding durations to date and time [W3C XS], part2, Appendix E.

- urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration

This function shall take two arguments, the first shall be "http://www.w3.org/2001/XMLSchema#dateTime" and the second shall be "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It shall return a result of "http://www.w3.org/2001/XMLSchema#dateTime". This function shall return the value by adding the second argument to the first argument according to the specification of adding durations to date and time [W3C XS], part 2, Appendix E.

- urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration

This function shall take two arguments, the first shall be "http://www.w3.org/2001/XMLSchema#dateTime" and the second shall be "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It shall return a result of "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration, then this function shall return the value by adding the corresponding negative duration, as per [W3C XS], part 2, Appendix E. If the second argument is a negative duration, then the result shall be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration" had been applied to the corresponding positive duration.

- urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration

This function shall take two arguments, the first shall be "http://www.w3.org/2001/XMLSchema#dateTime" and the second shall be "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It shall return a result of "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration, then this function shall return the value by adding the corresponding negative duration, as per [W3C XS], part 2, Appendix E. If the second argument is a negative duration, then the result shall be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration" had been applied to the corresponding positive duration.

- urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration

This function shall take two arguments, the first shall be "http://www.w3.org/2001/XMLSchema#date" and the second shall be "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It shall return a result of "http://www.w3.org/2001/XMLSchema#date". This function shall return the value by adding the second argument to the first argument according to the specification of adding duration to date [W3C XS], part 2, Appendix E.

- urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration

This function shall take two arguments, the first shall be "http://www.w3.org/2001/XMLSchema#date" and the second shall be

"http://www.w3.org/2001/XMLSchema#yearMonthDuration". It shall return a result of "http://www.w3.org/2001/XMLSchema#date". If the second argument is a positive duration, then this function shall return the value by adding the corresponding negative duration, as per [W3C XS], part 2, Appendix E. If the second argument is a negative duration, then the result shall be as if the function "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" had been applied to the corresponding positive duration.

A.3.8 Non-numeric comparison functions

These functions perform comparison operations on two arguments of non-numerical types.

- urn:oasis:names:tc:xacml:1.0:function:string-greater-than

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument is lexicographically strictly greater than the second argument. Otherwise, it shall return "False". The comparison shall use Unicode codepoint collation, as defined for the identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [W3C XF].

- urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument is lexicographically greater than or equal to the second argument. Otherwise, it shall return "False". The comparison shall use Unicode codepoint collation, as defined for the identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [W3C XF].

- urn:oasis:names:tc:xacml:1.0:function:string-less-than

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only the first argument is lexigraphically strictly less than the second argument. Otherwise, it shall return "False". The comparison shall use Unicode codepoint collation, as defined for the identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [W3C XF].

- urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only the first argument is lexigraphically less than or equal to the second argument. Otherwise, it shall return "False". The comparison shall use Unicode codepoint collation, as defined for the identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [W3C XF].

- urn:oasis:names:tc:xacml:1.0:function:time-greater-than

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument is greater than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" in [W3C XS], part 2, section 3.2.8. Otherwise, it shall return "False".

NOTE 1 – It is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

- urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument is greater than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" in [W3C XS] part 2, section 3.2.8. Otherwise, it shall return "False".

NOTE 2 – It is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

- urn:oasis:names:tc:xacml:1.0:function:time-less-than

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument is less than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" in [W3C XS], part 2, section 3.2.8. Otherwise, it shall return "False".

NOTE 3 – It is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

- urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument is less than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" in [W3C XS], part 2, section 3.2.8. Otherwise, it shall return "False".

NOTE 4 – It is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

- urn:oasis:names:tc:xacml:2.0:function:time-in-range

This function shall take three arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if the first argument falls in the range defined inclusively by the second and third arguments. Otherwise, it shall return "False". Regardless of its value, the third argument shall be interpreted as a time that is equal to, or later than by less than twenty-four hours, the second argument. If no time zone is provided for the first argument, it shall use the default time zone at the context handler. If no time zone is provided for the second or third arguments, then they shall use the time zone from the first argument.

- urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument is greater than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#dateTime" in [W3C XS], part 2, section 3.2.7. Otherwise, it shall return "False".

NOTE 5 – If a dateTime value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in [W3C XS].

- urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument is greater than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#dateTime" in [W3C XS], part 2, section 3.2.7. Otherwise, it shall return "False".

NOTE 6 – If a dateTime value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in [W3C XS].

- `urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than`

This function shall take two arguments of data-type "`http://www.w3.org/2001/XMLSchema#dateTime`" and shall return an "`http://www.w3.org/2001/XMLSchema#boolean`". It shall return "True" if and only if the first argument is less than the second argument according to the order relation specified for "`http://www.w3.org/2001/XMLSchema#dateTime`" in [W3C XS], part 2, section 3.2.7. Otherwise, it shall return "False".

NOTE 7 – If a dateTime value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in [W3C XS].

- `urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal`

This function shall take two arguments of data-type "`http://www.w3.org/2001/XMLSchema#dateTime`" and shall return an "`http://www.w3.org/2001/XMLSchema#boolean`". It shall return "True" if and only if the first argument is less than or equal to the second argument according to the order relation specified for "`http://www.w3.org/2001/XMLSchema#dateTime`" in [W3C XS], part 2, section 3.2.7. Otherwise, it shall return "False".

NOTE 8 – If a dateTime value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in [W3C XS].

- `urn:oasis:names:tc:xacml:1.0:function:date-greater-than`

This function shall take two arguments of data-type "`http://www.w3.org/2001/XMLSchema#date`" and shall return an "`http://www.w3.org/2001/XMLSchema#boolean`". It shall return "True" if and only if the first argument is greater than the second argument according to the order relation specified for "`http://www.w3.org/2001/XMLSchema#date`" in [W3C XS], part 2, section 3.2.9. Otherwise, it shall return "False".

NOTE 9 – If a date value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in [W3C XS].

- `urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal`

This function shall take two arguments of data-type "`http://www.w3.org/2001/XMLSchema#date`" and shall return an "`http://www.w3.org/2001/XMLSchema#boolean`". It shall return "True" if and only if the first argument is greater than or equal to the second argument according to the order relation specified for "`http://www.w3.org/2001/XMLSchema#date`" in [W3C XS], part 2, section 3.2.9. Otherwise, it shall return "False".

NOTE 10 – If a date value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in [W3C XS].

- `urn:oasis:names:tc:xacml:1.0:function:date-less-than`

This function shall take two arguments of data-type "`http://www.w3.org/2001/XMLSchema#date`" and shall return an "`http://www.w3.org/2001/XMLSchema#boolean`". It shall return "True" if and only if the first argument is less than the second argument according to the order relation specified for "`http://www.w3.org/2001/XMLSchema#date`" in [W3C XS], part 2, section 3.2.9. Otherwise, it shall return "False".

NOTE 11 – If a date value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in [W3C XS].

- `urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal`

This function shall take two arguments of data-type "`http://www.w3.org/2001/XMLSchema#date`" and shall return an "`http://www.w3.org/2001/XMLSchema#boolean`". It shall return "True" if and only if the first argument is less than or equal to the second argument according to the order relation

specified for "http://www.w3.org/2001/XMLSchema#date" in [W3C XS], part 2, section 3.2.9. Otherwise, it shall return "False".

NOTE 12 – If a date value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in [W3C XS].

A.3.9 String functions

The following functions operate on strings and convert to and from other data-types.

- urn:oasis:names:tc:xacml:2.0:function:string-concatenate

This function shall take two or more arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return "http://www.w3.org/2001/XMLSchema#string". The result shall be the concatenation, in order, of the arguments.

- urn:oasis:names:tc:xacml:3.0:function:boolean-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The result shall be the string converted to a Boolean. If the argument is not a valid lexical representation of a Boolean, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-boolean

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#boolean", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the Boolean converted to a string in the canonical form specified in [W3C XS].

- urn:oasis:names:tc:xacml:3.0:function:integer-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "http://www.w3.org/2001/XMLSchema#integer". The result shall be the string converted to an integer. If the argument is not a valid lexical representation of an integer, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-integer

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#integer", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the integer converted to a string in the canonical form specified in [W3C XS].

- urn:oasis:names:tc:xacml:3.0:function:double-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "http://www.w3.org/2001/XMLSchema#double". The result shall be the string converted to a double. If the argument is not a valid lexical representation of a double, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-double

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#double", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the double converted to a string in the canonical form specified in [W3C XS].

- urn:oasis:names:tc:xacml:3.0:function:time-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "http://www.w3.org/2001/XMLSchema#time". The result shall be the string converted to a time. If the argument is not a valid lexical representation of a time, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-time

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#time", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the time converted to a string in the canonical form specified in [W3C XS].

- urn:oasis:names:tc:xacml:3.0:function:date-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "http://www.w3.org/2001/XMLSchema#date". The result shall be the string converted to a date. If the argument is not a valid lexical representation of a date, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-date

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#date", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the date converted to a string in the canonical form specified in [W3C XS].

- urn:oasis:names:tc:xacml:3.0:function:dateTime-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "http://www.w3.org/2001/XMLSchema#dateTime". The result shall be the string converted to a dateTime. If the argument is not a valid lexical representation of a dateTime, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-dateTime

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#dateTime", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the dateTime converted to a string in the canonical form specified in [W3C XS].

- urn:oasis:names:tc:xacml:3.0:function:anyURI-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return "http://www.w3.org/2001/XMLSchema#anyURI". The result shall be URI constructed by converting the argument to URI. If the argument is not a valid lexical representation of URI, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#anyURI", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be URI converted to a string in the form it was originally represented in XML form.

- urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "http://www.w3.org/2001/XMLSchema#dayTimeDuration". The result shall be the string

converted to a dayTimeDuration. If the argument is not a valid lexical representation of a dayTimeDuration, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-dayTimeDuration

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#dayTimeDuration", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the dayTimeDuration converted to a string in the canonical form specified in [W3C XPathFunc].

- urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "http://www.w3.org/2001/XMLSchema#yearMonthDuration". The result shall be the string converted to a yearMonthDuration. If the argument is not a valid lexical representation of a yearMonthDuration, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-yearMonthDuration

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#yearMonthDuration", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the yearMonthDuration converted to a string in the canonical form specified in [W3C XPathFunc].

- urn:oasis:names:tc:xacml:3.0:function:x500Name-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". The result shall be the string converted to an x500Name. If the argument is not a valid lexical representation of a x500Name, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name

This function shall take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-type:x500Name", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the x500Name converted to a string in the form it was originally represented in XML form.

- urn:oasis:names:tc:xacml:3.0:function:rfc822Name-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". The result shall be the string converted to an rfc822Name. If the argument is not a valid lexical representation of an rfc822Name, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name

This function shall take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the rfc822Name converted to a string in the form it was originally represented in XML form.

- urn:oasis:names:tc:xacml:3.0:function:ipAddress-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". The result shall be the string converted to an ipAddress. If the argument is not a valid lexical representation of an ipAddress, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress

This function shall take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the ipAddress converted to a string in the form it was originally represented in XML form.

- urn:oasis:names:tc:xacml:3.0:function:dnsName-from-string

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string", and shall return an "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". The result shall be the string converted to a dnsName. If the argument is not a valid lexical representation of a dnsName, then the result shall be "Indeterminate" with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName

This function shall take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-type:dnsName", and shall return an "http://www.w3.org/2001/XMLSchema#string". The result shall be the dnsName converted to a string in the form it was originally represented in XML form.

- urn:oasis:names:tc:xacml:3.0:function:string-starts-with

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return "http://www.w3.org/2001/XMLSchema#boolean". The result shall be true if the second string begins with the first string, and false otherwise. Equality testing shall be done as defined for urn:oasis:names:tc:xacml:1.0:function:string-equal.

- urn:oasis:names:tc:xacml:3.0:function:anyURI-starts-with

This function shall take a first argument of data-type"http://www.w3.org/2001/XMLSchema#string" and a second argument of data-type "http://www.w3.org/2001/XMLSchema#anyURI" and shall return "http://www.w3.org/2001/XMLSchema#boolean". The result shall be true if URI converted to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI begins with the string, and false otherwise. Equality testing shall be done as defined for urn:oasis:names:tc:xacml:1.0:function:string-equal.

- urn:oasis:names:tc:xacml:3.0:function:string-ends-with

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return a "http://www.w3.org/2001/XMLSchema#boolean". The result shall be true if the second string ends with the first string, and false otherwise. Equality testing shall be done as defined for urn:oasis:names:tc:xacml:1.0:function:string-equal.

- urn:oasis:names:tc:xacml:3.0:function:anyURI-ends-with

This function shall take a first argument of data-type "http://www.w3.org/2001/XMLSchema#string" and a second argument of data-type "http://www.w3.org/2001/XMLSchema#anyURI" and shall return "http://www.w3.org/2001/XMLSchema#boolean". The result shall be true if URI converted to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI ends with the

string, and false otherwise. Equality testing shall be done as defined for urn:oasis:names:tc:xacml:1.0:function:string-equal.

- urn:oasis:names:tc:xacml:3.0:function:string-contains

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return "http://www.w3.org/2001/XMLSchema#boolean". The result shall be true if the second string contains the first string, and false otherwise. Equality testing shall be done as defined for urn:oasis:names:tc:xacml:1.0:function:string-equal.

- urn:oasis:names:tc:xacml:3.0:function:anyURI-contains

This function shall take a first argument of data-type "http://www.w3.org/2001/XMLSchema#string" and a second argument of data-type "http://www.w3.org/2001/XMLSchema#anyURI" and shall return "http://www.w3.org/2001/XMLSchema#boolean". The result shall be true if URI converted to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI contains the string, and false otherwise. Equality testing shall be done as defined for urn:oasis:names:tc:xacml:1.0:function:string-equal.

- urn:oasis:names:tc:xacml:3.0:function:string-substring

This function shall take a first argument of data-type "http://www.w3.org/2001/XMLSchema#string" and a second and a third argument of data-type "http://www.w3.org/2001/XMLSchema#integer" and shall return "http://www.w3.org/2001/XMLSchema#string". The result shall be the substring of the first argument beginning at the position given by the second argument and ending at the position before the position given by the third argument. The first character of the string has position zero. The negative integer value -1 given for the third arguments indicates the end of the string. If the second or third arguments are out of bounds, then the function must evaluate to "Indeterminate" with a status code of urn:oasis:names:tc:xacml:1.0:status:processing-error.

- urn:oasis:names:tc:xacml:3.0:function:anyURI-substring

This function shall take a first argument of data-type "http://www.w3.org/2001/XMLSchema#anyURI" and a second and a third argument of data-type "http://www.w3.org/2001/XMLSchema#integer" and shall return "http://www.w3.org/2001/XMLSchema#string". The result shall be the substring of the first argument converted to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI beginning at the position given by the second argument and ending at the position before the position given by the third argument. The first character of URI converted to a string has position zero. The negative integer value -1 given for the third arguments indicates the end of the string. If the second or third arguments are out of bounds, then the function must evaluate to "Indeterminate" with a status code of urn:oasis:names:tc:xacml:1.0:status:processing-error. If the resulting substring is not syntactically a valid URI, then the function must evaluate to "Indeterminate" with a status code of urn:oasis:names:tc:xacml:1.0:status:processing-error.

A.3.10 Bag functions

These functions operate on a bag of 'type' values, where type is one of the primitive data-types, and x.x is a version of XACML where the function has been defined. Some additional conditions defined for each function below shall cause the expression to evaluate to "Indeterminate".

- urn:oasis:names:tc:xacml:x.x:function:type-one-and-only

This function shall take a bag of 'type' values as an argument and shall return a value of 'type'. It shall return the only value in the bag. If the bag does not have one and only one value, then the expression shall evaluate to "Indeterminate".

- urn:oasis:names:tc:xacml:x.x:function:type-bag-size

This function shall take a bag of 'type' values as an argument and shall return an "http://www.w3.org/2001/XMLSchema#integer" indicating the number of values in the bag.

- urn:oasis:names:tc:xacml:x.x:function:type-is-in

This function shall take an argument of 'type' as the first argument and a bag of 'type' values as the second argument and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall evaluate to "True" if and only if the first argument matches by the "urn:oasis:names:tc:xacml:x.x:function:type-equal" any value in the bag. Otherwise, it shall return "False".

- urn:oasis:names:tc:xacml:x.x:function:type-bag

This function shall take any number of arguments of 'type' and return a bag of 'type' values containing the values of the arguments. An application of this function to zero arguments shall produce an empty bag of the specified data-type.

A.3.11 Set functions

These functions operate on bags mimicking sets by eliminating duplicate elements from a bag.

- urn:oasis:names:tc:xacml:x.x:function:type-intersection

This function shall take two arguments that are both a bag of 'type' values. It shall return a bag of 'type' values such that it contains only elements that are common between the two bags, which is determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal". No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal", shall exist in the result.

- urn:oasis:names:tc:xacml:x.x:function:type-at-least-one-member-of

This function shall take two arguments that are both a bag of 'type' values. It shall return "http://www.w3.org/2001/XMLSchema#boolean". The function shall evaluate to "True" if and only if at least one element of the first argument is contained in the second argument as determined by "urn:oasis:names:tc:xacml:x.x:function:type-is-in".

- urn:oasis:names:tc:xacml:x.x:function:type-union

This function shall take two or more arguments that are both a bag of 'type' values. The expression shall return a bag of 'type' such that it contains all elements of all the argument bags. No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal", shall exist in the result.

- urn:oasis:names:tc:xacml:x.x:function:type-subset

This function shall take two arguments that are both a bag of 'type' values. It shall return "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument is a subset of the second argument. Each argument shall be considered to have had its duplicates removed, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal", before the subset calculation.

- urn:oasis:names:tc:xacml:x.x:function:type-set-equals

This function shall take two arguments that are both a bag of 'type' values. It shall return "http://www.w3.org/2001/XMLSchema#boolean". It shall return the result of applying "urn:oasis:names:tc:xacml:1.0:function:and" to the application of "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the first and second arguments and the application of "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the second and first arguments.

A.3.12 Higher-order bag functions

This clause describes functions in XACML that perform operations on bags such that functions may be applied to the bags in general.

- `urn:oasis:names:tc:xacml:3.0:function:any-of`

This function applies a Boolean function between specific primitive values and a bag of values, and shall return "True" if and only if the predicate is "True" for at least one element of the bag.

This function shall take n+1 arguments, where n is one or greater. The first argument shall be a `<Function>` element that names a Boolean function that takes n arguments of primitive types. Under the remaining n arguments, n-1 parameters shall be values of primitive data-types and one shall be a bag of a primitive data-type. The expression shall be evaluated as if the function named in the `<Function>` argument were applied to the n-1 non-bag arguments and each element of the bag argument and the results are combined with "`urn:oasis:names:tc:xacml:1.0:function:or`".

For example, the following expression shall return "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
      <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal"/>
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
      </Apply>
</Apply>
```

This expression is "True" because the first argument is equal to at least one of the elements of the bag, according to the function.

- `urn:oasis:names:tc:xacml:3.0:function:all-of`

This function applies a Boolean function between a specific primitive value and a bag of values, and returns "True" if and only if the predicate is "True" for every element of the bag.

This function shall take n+1 arguments, where n is one or greater. The first argument shall be a `<Function>` element that names a Boolean function that takes n arguments of primitive types. Under the remaining n arguments, n-1 parameters shall be values of primitive data-types and one shall be a bag of a primitive data-type. The expression shall be evaluated as if the function named in the `<Function>` argument were applied to the n-1 non-bag arguments and each element of the bag argument and the results are combined with "`urn:oasis:names:tc:xacml:1.0:function:and`".

For example, the following expression shall evaluate to "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:all-of">
      <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
greater-than"/>
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
```

```
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">9</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
      </Apply>
</Apply>
```

This expression is "True" because the first argument (10) is greater than all of the elements of the bag (9, 3, 4 and 2).

- `urn:oasis:names:tc:xacml:3.0:function:any-of-any`

This function applies a Boolean function on each tuple from the cross product on all bags arguments, and returns "True" if and only if the predicate is "True" for at least one inside-function call.

This function shall take n+1 arguments, where n is one or greater. The first argument shall be a `<Function>` element that names a Boolean function that takes n arguments. The remaining arguments are either primitive data-types or bags of primitive types. The expression shall be evaluated as if the function named in the `<Function>` argument was applied between every tuple of the cross product on all bags and the primitive values, and the results were combined using "`urn:oasis:names:tc:xacml:1.0:function:or`". The semantics are that the result of the expression shall be "True" if and only if the applied predicate is "True" for at least one function call on the tuples from the bags and primitive values.

For example, the following expression shall evaluate to "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of-any">
      <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal"/>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Mary</AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
      </Apply>
</Apply>
```

This expression is "True" because at least one of the elements of the first bag, namely "Ringo", is equal to at least one of the elements of the second bag.

- `urn:oasis:names:tc:xacml:1.0:function:all-of-any`

This function applies a Boolean function between the elements of two bags. The expression shall be "True" if and only if the supplied predicate is "True" between each element of the first bag and any element of the second bag.

This function shall take three arguments. The first argument shall be a `<Function>` element that names a Boolean function that takes two arguments of primitive types. The second argument shall be a bag of a primitive data-type. The third argument shall be a bag of a primitive data-type. The expression shall be evaluated as if the "`urn:oasis:names:tc:xacml:3.0:function:any-of`" function had been applied to each value of the first bag and the whole of the second bag using the supplied xacml:Function, and the results were then combined using "`urn:oasis:names:tc:xacml:1.0:function:and`".

For example, the following expression shall evaluate to "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-any">
      <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
greater-than"/>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">20</AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">19</AttributeValue>
      </Apply>
</Apply>
```

This expression is "True" because each of the elements of the first bag is greater than at least one of the elements of the second bag.

- `urn:oasis:names:tc:xacml:1.0:function:any-of-all`

This function applies a Boolean function between the elements of two bags. The expression shall be "True" if and only if the supplied predicate is "True" between each element of the second bag and any element of the first bag.

This function shall take three arguments. The first argument shall be an `<Function>` element that names a Boolean function that takes two arguments of primitive types. The second argument shall be a bag of a primitive data-type. The third argument shall be a bag of a primitive data-type. The expression shall be evaluated as if the "`urn:oasis:names:tc:xacml:3.0:function:any-of`" function had been applied to each value of the second bag and the whole of the first bag using the supplied xacml:Function, and the results were then combined using "`urn:oasis:names:tc:xacml:1.0:function:and`".

For example, the following expression shall evaluate to "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-all">
      <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
greater-than"/>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
      </Apply>
</Apply>
```

This expression is "True" because, for all of the values in the second bag, there is a value in the first bag that is greater.

- `urn:oasis:names:tc:xacml:1.0:function:all-of-all`

This function applies a Boolean function between the elements of two bags. The expression shall be "True" if and only if the supplied predicate is "True" between each and every element of the first bag collectively against all the elements of the second bag.

This function shall take three arguments. The first argument shall be a `<Function>` element that names a Boolean function that takes two arguments of primitive types. The second argument shall be a bag of a primitive data-type. The third argument shall be a bag of a primitive data-type. The expression is evaluated as if the function named in the `<Function>` element were applied between every element of the second argument and every element of the third argument and the results were combined using "`urn:oasis:names:tc:xacml:1.0:function:and`". The semantics are that the result of the expression is "True" if and only if the applied predicate is "True" for all elements of the first bag compared to all the elements of the second bag.

For example, the following expression shall evaluate to "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-all">
      <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
greater-than"/>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">6</AttributeValue>
         <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
```

```
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
       </Apply>
</Apply>
```

This expression is "True" because all elements of the first bag, "5" and "6", are each greater than all of the integer values "1", "2", "3", "4" of the second bag.

- `urn:oasis:names:tc:xacml:3.0:function:map`

This function converts a bag of values to another bag of values.

This function shall take n+1 arguments, where n is one or greater. The first argument shall be a `<Function>` element naming a function that takes a n arguments of a primitive data-type and returns a value of a primitive data-type. Under the remaining n arguments, n-1 parameters shall be the values of primitive data-types and one shall be a bag of a primitive data-type. The expression shall be evaluated as if the function named in the `<Function>` argument were applied to the n-1 non-bag arguments and each element of the bag argument and resulting in a bag of the converted value. The result shall be a bag of the primitive data-type that is returned by the function named in the `<xacml:Function>` element.

For example, the following expression,

```
<Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:map">
       <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
normalize-to-lower-case">
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Hello</AttributeValue>
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">World!</AttributeValue>
       </Apply>
</Apply>
```

evaluates to a bag containing "hello" and "world!".

A.3.13 Regular-expression-based functions

These functions operate on various types using regular expressions and evaluate to "`http://www.w3.org/2001/XMLSchema#boolean`".

- `urn:oasis:names:tc:xacml:1.0:function:string-regexp-match`

This function decides a regular expression match. It shall take two arguments of "`http://www.w3.org/2001/XMLSchema#string`" and shall return an "`http://www.w3.org/2001/XMLSchema#boolean`". The first argument shall be a regular expression and the second argument shall be a general string. The function specification shall be that of the "`xf:matches`" function with the arguments reversed in [W3C XF], section 7.6.2.

- `urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match`

This function decides a regular expression match. It shall take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "http://www.w3.org/2001/XMLSchema#anyURI". It shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be URI. The function shall convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI, then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

- urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match

This function decides a regular expression match. It shall take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". It shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be an IPv4 or IPv6 address. The function shall convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress, then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

- urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match

This function decides a regular expression match. It shall take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". It shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be a DNS name. The function shall convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName, then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

- urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match

This function decides a regular expression match. It shall take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". It shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be an IETF RFC 822 name. The function shall convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name, then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

- urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match

This function decides a regular expression match. It shall take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". It shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be an ITU-T X.500 directory name. The function shall convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name, then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

A.3.14 Special match functions

These functions operate on various types and evaluate to "http://www.w3.org/2001/XMLSchema#boolean" based on the specified standard matching algorithm.

- urn:oasis:names:tc:xacml:1.0:function:x500Name-match

This function shall take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument matches some terminal sequence of RDNs from the second argument when compared using x500Name-equal.

As an example (non-normative), if the first argument is "O=Medico Corp,C=US" and the second argument is "cn=John Smith,o=Medico Corp, c=US", then the function will return "True".

- urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match

This function shall take two arguments; the first is of data-type "http://www.w3.org/2001/XMLSchema#string" and the second is of data-type "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". This function shall evaluate to "True" if the first argument matches the second argument according to the following specification.

An IETF RFC 822 name consists of a local-part followed by "@" followed by a domain-part. The local-part is case-sensitive, while the domain-part (which is usually a DNS name) is not case-sensitive.

The second argument contains a complete rfc822Name. The first argument is a complete or partial rfc822Name used to select appropriate values in the second argument as follows.

In order to match a particular address in the second argument, the first argument must specify the complete e-mail address to be matched. For example, if the first argument is "Anderson@sun.com", this matches a value in the second argument of "Anderson@sun.com" and "Anderson@SUN.COM", but not "Anne.Anderson@sun.com", "anderson@sun.com" or "Anderson@east.sun.com".

In order to match any address at a particular domain in the second argument, the first argument must specify only a domain name (usually a DNS name). For example, if the first argument is "sun.com", this matches a value in the second argument of "Anderson@sun.com" or "Baxter@SUN.COM", but not "Anderson@east.sun.com".

In order to match any address in a particular domain in the second argument, the first argument must specify the desired domain-part with a leading ".". For example, if the first argument is ".east.sun.com", this matches a value in the second argument of "Anderson@east.sun.com" and "anne.anderson@ISRG.EAST.SUN.COM" but not "Anderson@sun.com".

A.3.15 XPath-based functions

This clause specifies functions that take XPath expressions for arguments. An XPath expression evaluates to a node-set, which is a set of XML nodes that match the expression. A node or node-set is not in the formal data-type system of XACML. All comparisons or other operations on node-sets are performed in isolation of the particular function specified. The context nodes and namespace mappings of the XPath expressions are defined by the XPath data-type, see clause B.3. The following functions are defined:

- urn:oasis:names:tc:xacml:3.0:function:xpath-node-count

This function shall take an "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" as an argument and evaluates to an "http://www.w3.org/2001/XMLSchema#integer". The value returned from the function shall be the count of the nodes within the node-set that matches the given

XPath expression. If the <Content> element of the category to which the XPath expression applies to is not present in the request, this function shall return a value of zero.

- urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal

This function shall take two "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" arguments and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall return "True" if any of the XML nodes in the node-set matched by the first argument equals any of the XML nodes in the node-set matched by the second argument. Two nodes are considered equal if they have the same identity. If the <Content> element of the category to which either XPath expression applies to is not present in the request, this function shall return a value of "False".

- urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

This function shall take two "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" arguments and shall return an "http://www.w3.org/2001/XMLSchema#boolean". This function shall evaluate to "True" if one of the following two conditions is satisfied: (1) Any of the XML nodes in the node-set matched by the first argument is equal to any of the XML nodes in the node-set matched by the second argument; (2) any node below any of the XML nodes in the node-set matched by the first argument is equal to any of the XML nodes in the node-set matched by the second argument. Two nodes are considered equal if they have the same identity. If the <Content> element of the category to which either XPath expression applies to is not present in the request, this function shall return a value of "False".

NOTE – The first condition is equivalent to "xpath-node-equal", and guarantees that "xpath-node-equal" is a special case of "xpath-node-match".

A.3.16 Other functions

- urn:oasis:names:tc:xacml:3.0:function:access-permitted

This function shall take an "http://www.w3.org/2001/XMLSchema#anyURI" and an "http://www.w3.org/2001/XMLSchema#string" as arguments. The first argument shall be interpreted as an attribute category. The second argument shall be interpreted as the XML content of an <Attributes> element with Category equal to the first argument. The function evaluates to an "http://www.w3.org/2001/XMLSchema#boolean". This function shall return "True" if and only if the policy evaluation described below returns the value of "Permit".

The following evaluation is described as if the context is actually instantiated, but it is only required that an equivalent result be obtained.

The function shall construct a new context, by copying all the information from the current context, omitting any <Attributes> element with Category equal to the first argument. The second function argument shall be added to the context as the content of an <Attributes> element with Category equal to the first argument.

The function shall invoke a complete policy evaluation using the newly constructed context. This evaluation shall be completely isolated from the evaluation which invoked the function, but shall use all current policies and combining algorithms, including any per request policies.

PDP shall detect any loop which may occur if successive evaluations invoke this function by counting the number of total invocations of any instance of this function during any single initial invocation of PDP. If the total number of invocations exceeds the bound for such invocations, the initial invocation of this function evaluates to "Indeterminate" with a "urn:oasis:names:tc:xacml:1.0:status:processing-error" status code. See also the security considerations in Appendix III.1.8.

A.3.17 Extension functions and primitive types

Functions and primitive types are specified by string identifiers allowing for the introduction of functions in addition to those specified by XACML. This approach allows one to extend the XACML module with special functions and special primitive data-types.

In order to preserve the integrity of the XACML evaluation strategy, the result of an extension function shall depend only on the values of its arguments. Global and hidden parameters shall NOT affect the evaluation of an expression. Functions shall NOT have side effects, as evaluation order cannot be guaranteed in a standard way.

A.4 Functions, data-types, attributes and algorithms planned for deprecation

The following functions, data-types and algorithms have been defined by previous versions of XACML and newer and better alternatives are defined in this Recommendation. Their use is discouraged for new use and they are candidates for deprecation in future versions of XACML.

The following xpath-based functions have been replaced with equivalent functions which use the new `urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression` datatype instead of strings.

- `urn:oasis:names:tc:xacml:1.0:function:xpath-node-count`
- Replaced with `urn:oasis:names:tc:xacml:3.0:function:xpath-node-count`
- `urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal`
- Replaced with `urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal`
- `urn:oasis:names:tc:xacml:1.0:function:xpath-node-match`
- Replaced with `urn:oasis:names:tc:xacml:3.0:function:xpath-node-match`

The following URI and string concatenation function has been replaced with a string to URI conversion function, which allows the use of the general string functions with URI through string conversion.

- `urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate`
- Replaced by `urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI`

The following identifiers have been replaced with official identifiers defined by W3C.

- `http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration`
- Replaced with `http://www.w3.org/2001/XMLSchema#dayTimeDuration`
- `http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration`
- Replaced with `http://www.w3.org/2001/XMLSchema#yearMonthDuration`

The following functions have been replaced with functions which use the updated `dayTimeDuration` and `yearMonthDuration` data-types.

- `urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal`
- Replaced with `urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal`
- `urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal`
- Replaced with `urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal`
- `urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration`
- Replaced with `urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration`
- `urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration`
- Replaced with `urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration`

- urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration
- Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
- urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration
- Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
- urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration
- Replaced with urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
- urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration
- Replaced with urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration

The following attribute identifiers have been replaced with new identifiers.

- urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address
- Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-address
- urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name
- Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-locality:dns-name

The following combining algorithms have been replaced with new variants which allow for better handling of "Indeterminate" results.

- urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides
- Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides
- urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides
- Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides
- urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides
- Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides
- urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides
- Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides
- urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides
- Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides
- urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides
- Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides
- urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides
- Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-overrides
- urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides

- Replaced with `urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-overrides`

Annex B

XACML identifiers

(This annex forms an integral part of this Recommendation.)

This annex defines standard identifiers for commonly used entities.

B.1 XACML namespaces

XACML is defined using this identifier.

`urn:oasis:names:tc:xacml:3.0:core:schema`

B.2 Attribute categories

The following attribute category identifiers must be used when XACML 2.0 or earlier policy or request is translated into XACML 3.0.

Attributes previously placed in the Resource, Action, and Environment sections of a request are placed in an attribute category with the following identifiers respectively. It is recommended that they be used to list attributes of resources, actions, and the environment respectively when authoring XACML 3.0 policies or requests.

`urn:oasis:names:tc:xacml:3.0:attribute-category:resource`

`urn:oasis:names:tc:xacml:3.0:attribute-category:action`

`urn:oasis:names:tc:xacml:3.0:attribute-category:environment`

Attributes previously placed in the Subject section of a request are placed in an attribute category which is identical to the subject category in XACML 2.0, as defined below. It is recommended that they be used to list attributes of subjects when authoring XACML 3.0 policies or requests.

This identifier indicates the system entity that initiated the access request. That is, the initial entity in a request chain. If the subject category is not specified in XACML 2.0, this is the default translation value.

`urn:oasis:names:tc:xacml:1.0:subject-category:access-subject`

This identifier indicates the system entity that will receive the results of the request (used when it is distinct from the access-subject).

`urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject`

This identifier indicates a system entity through which the access request was passed.

`urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject`

This identifier indicates a system entity associated with a local or remote codebase that generated the request. Corresponding subject attributes might include URL from which it was loaded and/or the identity of the code-signer.

`urn:oasis:names:tc:xacml:1.0:subject-category:codebase`

This identifier indicates a system entity associated with the computer that initiated the access request. An example would be an IPSec identity.

`urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine`

B.3 Data-types

The following identifiers indicate data-types that are defined in clause A.2.

urn:oasis:names:tc:xacml:1.0:data-type:x500Name.

urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name

urn:oasis:names:tc:xacml:2.0:data-type:ipAddress

urn:oasis:names:tc:xacml:2.0:data-type:dnsName

urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression

The following data-type identifiers are defined by XML Schema [W3C XS].

http://www.w3.org/2001/XMLSchema#string

http://www.w3.org/2001/XMLSchema#boolean

http://www.w3.org/2001/XMLSchema#integer

http://www.w3.org/2001/XMLSchema#double

http://www.w3.org/2001/XMLSchema#time

http://www.w3.org/2001/XMLSchema#date

http://www.w3.org/2001/XMLSchema#dateTime

http://www.w3.org/2001/XMLSchema#anyURI

http://www.w3.org/2001/XMLSchema#hexBinary

http://www.w3.org/2001/XMLSchema#base64Binary

The following data-type identifiers correspond to the dayTimeDuration and yearMonthDuration data-types defined in [W3C XF], sections 10.3.2 and 10.3.1, respectively.

http://www.w3.org/2001/XMLSchema#dayTimeDuration

http://www.w3.org/2001/XMLSchema#yearMonthDuration

B.4 Subject attributes

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 clause B.2).

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.

This identifier indicates the name of the subject.

urn:oasis:names:tc:xacml:1.0:subject:subject-id

This identifier indicates the security domain of the subject. It identifies the administrator and policy that manages the namespace in which the subject id is administered.

urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier

This identifier indicates a public key used to confirm the subject's identity.

urn:oasis:names:tc:xacml:1.0:subject:key-info

This identifier indicates the time at which the subject was authenticated.

urn:oasis:names:tc:xacml:1.0:subject:authentication-time

This identifier indicates the method used to authenticate the subject.

urn:oasis:names:tc:xacml:1.0:subject:authentication-method

This identifier indicates the time at which the subject initiated the access request, according to PEP.

urn:oasis:names:tc:xacml:1.0:subject:request-time

This identifier indicates the time at which the subject's current session began, according to PEP.

urn:oasis:names:tc:xacml:1.0:subject:session-start-time

The following identifiers indicate the location where authentication credentials were activated.

This identifier indicates that the location is expressed as an IP address.

urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-address

The corresponding attribute shall be of data-type "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress".

This identifier indicates that the location is expressed as a DNS name.

urn:oasis:names:tc:xacml:3.0:subject:authn-locality:dns-name

The corresponding attribute shall be of data-type "urn:oasis:names:tc:xacml:2.0:data-type:dnsName".

Where a suitable attribute is already defined in LDAP [IETF RFC 2256] and [IETF RFC 2798], the XACML identifier shall be formed by adding the attribute name to the URI of the LDAP specification. For example, the attribute name for the userPassword defined in [IETF RFC 2256] shall be:

http://www.ietf.org/rfc/rfc2256.txt#userPassword

B.5 Resource attributes

These identifiers indicate attributes of the resource. When used, it is recommended they appear within the <Attributes> element of the request context with Category urn:oasis:names:tc:xacml:3.0:attribute-category:resource.

This attribute identifies the resource to which access is requested.

urn:oasis:names:tc:xacml:1.0:resource:resource-id

This attribute identifies the namespace of the top element(s) of the contents of the <Content> element. In the case where the resource content is supplied in the request context and the resource namespaces are defined in the resource, PEP may provide this attribute in the request to indicate the namespaces of the resource content. In this case, there shall be one value of this attribute for each unique namespace of the top level elements in the <Content> element. The type of the corresponding attribute shall be "http://www.w3.org/2001/XMLSchema#anyURI".

urn:oasis:names:tc:xacml:2.0:resource:target-namespace

B.6 Action attributes

These identifiers indicate attributes of the action being requested. When used, it is recommended they appear within the <Attributes> element of the request context with Category urn:oasis:names:tc:xacml:3.0:attribute-category:action.

This attribute identifies the action for which access is requested.

urn:oasis:names:tc:xacml:1.0:action:action-id

Where the action is implicit, the value of the action-id attribute shall be

urn:oasis:names:tc:xacml:1.0:action:implied-action

This attribute identifies the namespace in which the action-id attribute is defined. urn:oasis:names:tc:xacml:1.0:action:action-namespace

B.7 Environment attributes

These identifiers indicate attributes of the environment within which the decision request is to be evaluated. When used in the decision request, it is recommended they appear in the `<Attributes>` element of the request context with `Category urn:oasis:names:tc:xacml:3.0:attribute-category:environment`.

This identifier indicates the current time at the context handler. In practice, it is the time at which the request context was created. For this reason, if these identifiers appear in multiple places within a `<Policy>` or `<PolicySet>`, then the same value shall be assigned to each occurrence in the evaluation procedure, regardless of how much time elapses between the processing of the occurrences.

`urn:oasis:names:tc:xacml:1.0:environment:current-time`

The corresponding attribute shall be of data-type "`http://www.w3.org/2001/XMLSchema#time`".

`urn:oasis:names:tc:xacml:1.0:environment:current-date`

The corresponding attribute shall be of data-type "`http://www.w3.org/2001/XMLSchema#date`".

`urn:oasis:names:tc:xacml:1.0:environment:current-dateTime`

The corresponding attribute shall be of data-type "`http://www.w3.org/2001/XMLSchema#dateTime`".

B.8 Status codes

The following status code values are defined.

This identifier indicates success.

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

This identifier indicates that all the attributes necessary to make a policy decision were not available (see clause 8.58).

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

This identifier indicates that some attribute value contained a syntax error, such as a letter in a numeric field.

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

This identifier indicates that an error occurred during policy evaluation. An example would be division by zero.

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

B.9 Combining algorithms

The deny-overrides rule-combining algorithm has the following value for the `ruleCombiningAlgId` attribute:

`urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides`

The deny-overrides policy-combining algorithm has the following value for the `policyCombiningAlgId` attribute:

`urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides`

The permit-overrides rule-combining algorithm has the following value for the `ruleCombiningAlgId` attribute:

`urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides`

The permit-overrides policy-combining algorithm has the following value for the `policyCombiningAlgId` attribute:

urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

The first-applicable rule-combining algorithm has the following value for the ruleCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

The first-applicable policy-combining algorithm has the following value for the policyCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

The only-one-applicable-policy policy-combining algorithm has the following value for the policyCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

The ordered-deny-overrides rule-combining algorithm has the following value for the ruleCombiningAlgId attribute:

urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

The ordered-deny-overrides policy-combining algorithm has the following value for the policyCombiningAlgId attribute:

urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides

The ordered-permit-overrides rule-combining algorithm has the following value for the ruleCombiningAlgId attribute:

urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-overrides

The ordered-permit-overrides policy-combining algorithm has the following value for the policyCombiningAlgId attribute:

urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-overrides

The deny-unless-permit rule-combining algorithm has the following value for the ruleCombiningAlgId attribute:

urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit

The permit-unless-deny rule-combining algorithm has the following value for the ruleCombiningAlgId attribute:

urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny

The deny-unless-permit policy-combining algorithm has the following value for the policyCombiningAlgId attribute:

urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit

The permit-unless-deny policy-combining algorithm has the following value for the policyCombiningAlgId attribute:

urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny

The legacy deny-overrides rule-combining algorithm has the following value for the ruleCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

The legacy deny-overrides policy-combining algorithm has the following value for the policyCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides

The legacy permit-overrides rule-combining algorithm has the following value for the ruleCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides

The legacy permit-overrides policy-combining algorithm has the following value for the policyCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides

The legacy ordered-deny-overrides rule-combining algorithm has the following value for the ruleCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides

The legacy ordered-deny-overrides policy-combining algorithm has the following value for the policyCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides

The legacy ordered-permit-overrides rule-combining algorithm has the following value for the ruleCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides

The legacy ordered-permit-overrides policy-combining algorithm has the following value for the policyCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides

Annex C

Combining algorithms

(This annex forms an integral part of this Recommendation.)

This annex contains a description of the rule- and policy-combining algorithms specified by XACML. Pseudo code is normative, descriptions in English are non-normative.

The legacy combining algorithms are defined in previous versions of XACML, and are retained for compatibility reasons. It is recommended that the new combining algorithms be used instead of the legacy combining algorithms for new use.

NOTE – In each case an implementation is conformant as long as it produces the same result as is specified here, regardless of how and in what order the implementation behaves internally.

C.1 Extended "Indeterminate" values

Some combining algorithms are defined in terms of an extended set of "Indeterminate" values. See clause 10.10 for the definition of the Extended "Indeterminate" values. For these algorithms, PDP must keep track of the extended set of "Indeterminate" values during rule and policy combining.

The output of a combining algorithm which does not track the extended set of "Indeterminate" values must be treated as "Indeterminate{DP}" for the value "Indeterminate" by a combining algorithm which tracks the extended set of "Indeterminate" values.

A combining algorithm which does not track the extended set of "Indeterminate" values must treat the output of a combining algorithm which tracks the extended set of "Indeterminate" values as an "Indeterminate" for any of the possible values of the extended set of "Indeterminate".

C.2 Deny-overrides

This clause defines the "Deny-overrides" rule-combining algorithm of a policy and policy-combining algorithm of a policy set.

This combining algorithm makes use of the extended "Indeterminate".

The rule-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides`

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides`

The following is a non-normative informative description of this combining algorithm.

The deny overrides combining algorithm is intended for those cases where a deny decision should have priority over a permit decision. This algorithm has the following behaviour.

1) If any decision is "Deny", the result is "Deny".
2) Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
3) Otherwise, if any decision is "Indeterminate{D}" and another decision is "Indeterminate{P}" or "Permit", the result is "Indeterminate{DP}".
4) Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
5) Otherwise, if any decision is "Permit", the result is "Permit".
6) Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
7) Otherwise, the result is "NotApplicable".

The following pseudo-code represents the normative specification of this combining algorithm. The algorithm is presented here in a form where the input to it is an array with children (the policies, policy sets or rules) of the policy or policy set. The children may be processed in any order, so the set of obligations or advice provided by this algorithm is not deterministic.

```
Decision denyOverridesCombiningAlgorithm(Node[] children)
{
   Boolean atLeastOneErrorD  = false;
   Boolean atLeastOneErrorP  = false;
   Boolean atLeastOneErrorDP  = false;
   Boolean atLeastOnePermit = false;
   for( i=0 ; i < lengthOf(children) ; i++ )
   {
          Decision decision = children[i].evaluate();
          if (decision == Deny)
          {
                 return Deny;
          }
          if (decision == Permit)
          {
                 atLeastOnePermit = true;
                 continue;
          }
          if (decision == NotApplicable)
          {
                 continue;
          }
          if (decision == Indeterminate{D})
          {
                 atLeastOneErrorD = true;
                 continue;
          }
          if (decision == Indeterminate{P})
          {
                 atLeastOneErrorP = true;
                 continue;
          }
          if (decision == Indeterminate{DP})
          {
                 atLeastOneErrorDP = true;
                 continue;
          }
   }
   if (atLeastOneErrorDP)
   {
          return Indeterminate{DP};
   }
   if (atLeastOneErrorD && (atLeastOneErrorP || atLeastOnePermit))
   {
          return Indeterminate{DP};
   }
   if (atLeastOneErrorD)
   {
          return Indeterminate{D};
   }
   if (atLeastOnePermit)
   {
          return Permit;
   }
   if (atLeastOneErrorP)
   {
          return Indeterminate{P};
   }
   return NotApplicable;
}
```

Obligations and advice shall be combined as described in clause 10.18.

C.3 Ordered-deny-overrides

The following specification defines the "Ordered-deny-overrides" rule-combining algorithm of a policy.

The behaviour of this algorithm is identical to that of the "Deny-overrides" rule-combining algorithm with one exception. The order in which the collection of rules is evaluated shall match the order as listed in the policy.

The rule-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides`

The following specification defines the "Ordered-deny-overrides" policy-combining algorithm of a policy set.

The behaviour of this algorithm is identical to that of the "Deny-overrides" policy-combining algorithm with one exception. The order in which the collection of policies is evaluated shall match the order as listed in the policy set.

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides`

C.4 Permit-overrides

This clause defines the "Permit-overrides" rule-combining algorithm of a policy and policy-combining algorithm of a policy set.

This combining algorithm makes use of the extended "Indeterminate".

The rule-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides`

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides`

The following is a non-normative informative description of this combining algorithm.

The permit overrides combining algorithm is intended for those cases where a permit decision should have priority over a deny decision. This algorithm has the following behaviour.

1) If any decision is "Permit", the result is "Permit".
2) Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
3) Otherwise, if any decision is "Indeterminate{P}" and another decision is "Indeterminate{D}" or Deny, the result is "Indeterminate{DP}".
4) Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
5) Otherwise, if any decision is "Deny", the result is "Deny".
6) Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
7) Otherwise, the result is "NotApplicable".

The following pseudo-code represents the normative specification of this combining algorithm. The algorithm is presented here in a form where the input to it is an array with all children (the policies, policy sets or rules) of the policy or policy set. The children may be processed in any order, so the set of obligations or advice provided by this algorithm is not deterministic.

```
Decision permitOverridesCombiningAlgorithm(Node[] children)
{
   Boolean atLeastOneErrorD  = false;
   Boolean atLeastOneErrorP  = false;
   Boolean atLeastOneErrorDP  = false;
   Boolean atLeastOneDeny = false;
   for( i=0 ; i < lengthOf(children) ; i++ )
   {
          Decision decision = children[i].evaluate();
          if (decision == Deny)
          {
                 atLeastOneDeny = true;
                 continue;
          }
          if (decision == Permit)
          {
                 return Permit;
          }
          if (decision == NotApplicable)
          {
                 continue;
          }
          if (decision == Indeterminate{D})
          {
                 atLeastOneErrorD = true;
                 continue;
          }
          if (decision == Indeterminate{P})
          {
                 atLeastOneErrorP = true;
                 continue;
          }
          if (decision == Indeterminate{DP})
          {
                 atLeastOneErrorDP = true;
                 continue;
          }
   }
   if (atLeastOneErrorDP)
   {
          return Indeterminate{DP};
   }
   if (atLeastOneErrorP && (atLeastOneErrorD || atLeastOneDeny))
   {
          return Indeterminate{DP};
   }
   if (atLeastOneErrorP)
   {
          return Indeterminate{P};
   }
   if (atLeastOneDeny)
   {
          return Deny;
   }
   if (atLeastOneErrorD)
   {
          return Indeterminate{D};
   }
   return NotApplicable;
}
```

C.5 Ordered-permit-overrides

The following specification defines the "Ordered-permit-overrides" rule-combining algorithm of a policy.

The behaviour of this algorithm is identical to that of the "Permit-overrides" rule-combining algorithm with one exception. The order in which the collection of rules is evaluated shall match the order as listed in the policy.

The rule-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-overrides`

The following specification defines the "Ordered-permit-overrides" policy-combining algorithm of a policy set.

The behaviour of this algorithm is identical to that of the "Permit-overrides" policy-combining algorithm with one exception. The order in which the collection of policies is evaluated shall match the order as listed in the policy set.

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-overrides`

C.6 Deny-unless-permit

This clause defines the "Deny-unless-permit" rule-combining algorithm of a policy or policy-combining algorithm of a policy set.

The rule-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit`

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit`

The following is a non-normative informative description of this combining algorithm.

The "Deny-unless-permit" combining algorithm is intended for those cases where a permit decision should have priority over a deny decision, and an "Indeterminate" or "NotApplicable" must never be the result. It is particularly useful at the top level in a policy structure to ensure that PDP will always return a definite "Permit" or "Deny" result. This algorithm has the following behaviour.

1) If any decision is "Permit", the result is "Permit".

2) Otherwise, the result is "Deny".

The following pseudo-code represents the normative specification of this combining algorithm. The algorithm is presented here in a form where the input to it is an array with all the children (the policies, policy sets or rules) of the policy or policy set. The children may be processed in any order, so the set of obligations or advice provided by this algorithm is not deterministic.

```
Decision denyUnlessPermitCombiningAlgorithm(Node[] children)
{
   for( i=0 ; i < lengthOf(children) ; i++ )
   {
          if (children[i].evaluate() == Permit)
          {
                 return Permit;
          }
   }
   return Deny;
}
```

Obligations and advice shall be combined as described in clause 10.18.

C.7 Permit-unless-deny

This clause defines the "Permit-unless-deny" rule-combining algorithm of a policy or policy-combining algorithm of a policy set.

The rule-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny`

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny`

The following is a non-normative informative description of this combining algorithm.

The "Permit-unless-deny" combining algorithm is intended for those cases where a deny decision should have priority over a permit decision, and an "Indeterminate" or "NotApplicable" must never be the result. It is particularly useful at the top level in a policy structure to ensure that PDP will always return a definite "Permit" or "Deny" result. This algorithm has the following behaviour.

1) If any decision is "Deny", the result is "Deny".

2) Otherwise, the result is "Permit".

The following pseudo-code represents the normative specification of this combining algorithm. The algorithm is presented here in a form where the input to it is an array with all the children (the policies, policy sets or rules) of the policy or policy set. The children may be processed in any order, so the set of obligations or advice provided by this algorithm is not deterministic.

```
Decision permitUnlessDenyCombiningAlgorithm(Node[] children)
{
   for( i=0 ; i < lengthOf(children) ; i++ )
   {
          if (children[i].evaluate() == Deny)
          {
                 return Deny;
          }
   }
   return Permit;
}
```

Obligations and advice shall be combined as described in clause 10.18.

C.8 First-applicable

This clause defines the "First-applicable" rule-combining algorithm of a policy and policy-combining algorithm of a policy set.

The rule combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable`

The following is a non-normative informative description of the "First-applicable" rule-combining algorithm of a policy.

Each rule shall be evaluated in the order in which it is listed in the policy. For a particular rule, if the target matches and the condition evaluates to "True", then the evaluation of the policy shall halt and the corresponding effect of the rule shall be the result of the evaluation of the policy (i.e., "Permit" or "Deny"). For a particular rule selected in the evaluation, if the target evaluates to "False" or the condition evaluates to "False", then the next rule in the order shall be evaluated. If no further rule in the order exists, then the policy shall evaluate to "NotApplicable".

If an error occurs while evaluating the target or condition of a rule, then the evaluation shall halt, and the policy shall evaluate to "Indeterminate", with the appropriate error status.

The following pseudo-code represents the normative specification of this rule-combining algorithm.

```
Decision firstApplicableEffectRuleCombiningAlgorithm(Rule[] rules)
{
   for( i = 0 ; i < lengthOf(rules) ; i++ )
   {
          Decision decision = evaluate(rules[i]);
          if (decision == Deny)
          {
                return Deny;
          }
          if (decision == Permit)
          {
                return Permit;
          }
          if (decision == NotApplicable)
          {
                continue;
          }
          if (decision == Indeterminate)
          {
                return Indeterminate;
          }
   }
   return NotApplicable;
}
```

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable`

The following is a non-normative informative description of the "First-applicable" policy-combining algorithm of a policy set.

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".

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.

The following pseudo-code represents the normative specification of this policy-combining algorithm.

```
Decision firstApplicableEffectPolicyCombiningAlgorithm(Policy[] policies)
{
    for( i = 0 ; i < lengthOf(policies) ; i++ )
    {
        Decision decision = evaluate(policies[i]);
        if(decision == Deny)
        {
            return Deny;
        }
        if(decision == Permit)
        {
            return Permit;
        }
        if (decision == NotApplicable)
        {
            continue;
```

```
        }
        if (decision == Indeterminate)
        {
            return Indeterminate;
        }
    }
    return NotApplicable;
}
```

Obligations and advice of the individual policies shall be combined as described in clause 10.18.

C.9 Only-one-applicable

This clause defines the "Only-one-applicable" policy-combining algorithm of a policy set.

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable`

The following is a non-normative informative description of the "Only-one-applicable" policy-combining algorithm of a policy set.

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-combining algorithm shall be "NotApplicable". If more than one policy is considered applicable by virtue of its target, then the result of the policy-combining algorithm shall be "Indeterminate".

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.

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.

The following pseudo-code represents the normative specification of this policy-combining algorithm.

```
Decision onlyOneApplicablePolicyPolicyCombiningAlogrithm(Policy[] policies)
{
  Boolean          atLeastOne     = false;
  Policy           selectedPolicy = null;
  ApplicableResult appResult;

  for ( i = 0; i < lengthOf(policies) ; i++ )
  {
     appResult = isApplicable(policies[I]);

     if ( appResult == Indeterminate )
     {
         return Indeterminate;
     }
     if( appResult == Applicable )
     {
         if ( atLeastOne )
         {
             return Indeterminate;
         }
         else
         {
             atLeastOne     = true;
             selectedPolicy = policies[i];
         }
     }
     if ( appResult == NotApplicable )
     {
         continue;
     }
```

```
  }
  if ( atLeastOne )
  {
      return evaluate(selectedPolicy);
  }
  else
  {
      return NotApplicable;
  }
}
```

Obligations and advice of the individual rules shall be combined as described in clause 10.18.

C.10 Legacy Deny-overrides

This clause defines the legacy "Deny-overrides" rule-combining algorithm of a policy and policy-combining algorithm of a policy set.

The rule-combining algorithm defined here has the following identifier:

urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

The following is a non-normative informative description of this combining algorithm.

The "Deny-overrides" rule-combining algorithm is intended for those cases where a "Deny" decision should have priority over a "Permit" decision. This algorithm has the following behaviour.

1) If any rule evaluates to "Deny", the result is "Deny".
2) Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is "Indeterminate".
3) Otherwise, if any rule evaluates to "Permit", the result is "Permit".
4) Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate", the result is "Indeterminate".
5) Otherwise, the result is "NotApplicable".

The following pseudo-code represents the normative specification of this rule-combining algorithm.

```
Decision denyOverridesRuleCombiningAlgorithm(Rule[] rules)
{
   Boolean atLeastOneError  = false;
   Boolean potentialDeny    = false;
   Boolean atLeastOnePermit = false;
   for( i=0 ; i < lengthOf(rules) ; i++ )
   {
          Decision decision = evaluate(rules[i]);
          if (decision == Deny)
          {
                return Deny;
          }
          if (decision == Permit)
          {
                atLeastOnePermit = true;
                continue;
          }
          if (decision == NotApplicable)
          {
                continue;
          }
          if (decision == Indeterminate)
          {
                atLeastOneError = true;

                if (effect(rules[i]) == Deny)
                {
                      potentialDeny = true;
                }
                continue;
```

```
            }
      }
      if (potentialDeny)
      {
            return Indeterminate;
      }
      if (atLeastOnePermit)
      {
            return Permit;
      }
      if (atLeastOneError)
      {
            return Indeterminate;
      }
      return NotApplicable;
}
```

Obligations and advice of the individual rules shall be combined as described in clause 10.18.

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides`

The following is a non-normative informative description of this combining algorithm.

The "Deny-overrides" policy combining algorithm is intended for those cases where a "Deny" decision should have priority over a "Permit" decision. This algorithm has the following behaviour.

1) If any policy evaluates to "Deny", the result is "Deny".
2) Otherwise, if any policy evaluates to "Indeterminate", the result is "Deny".
3) Otherwise, if any policy evaluates to "Permit", the result is "Permit".
4) Otherwise, the result is "NotApplicable".

The following pseudo-code represents the normative specification of this policy-combining algorithm.

```
Decision denyOverridesPolicyCombiningAlgorithm(Policy[] policies)
{
   Boolean atLeastOnePermit = false;
   for( i=0 ; i < lengthOf(policies) ; i++ )
   {
         Decision decision = evaluate(policies[i]);
         if (decision == Deny)
         {
               return Deny;
         }
         if (decision == Permit)
         {
               atLeastOnePermit = true;
               continue;
         }
         if (decision == NotApplicable)
         {
               continue;
         }
         if (decision == Indeterminate)
         {
               return Deny;
         }
   }
   if (atLeastOnePermit)
   {
         return Permit;
   }
   return NotApplicable;
}
```

Obligations and advice of the individual policies shall be combined as described in clause 10.18.

C.11 Legacy Ordered-deny-overrides

The following specification defines the legacy "Ordered-deny-overrides" rule-combining algorithm of a policy.

The behaviour of this algorithm is identical to that of the "Deny-overrides" rule-combining algorithm with one exception. The order in which the collection of rules is evaluated shall match the order as listed in the policy.

The rule-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides`

The following specification defines the legacy "Ordered-deny-overrides" policy-combining algorithm of a policy set.

The behaviour of this algorithm is identical to that of the "Deny-overrides" policy-combining algorithm with one exception. The order in which the collection of policies is evaluated shall match the order as listed in the policy set.

The rule-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides`

C.12 Legacy Permit-overrides

This clause defines the legacy "Permit-overrides" rule-combining algorithm of a policy and policy-combining algorithm of a policy set.

The rule-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides`

The following is a non-normative informative description of this combining algorithm.

The "Permit-overrides" rule-combining algorithm is intended for those cases where a "Permit" decision should have priority over a "Deny" decision. This algorithm has the following behaviour.

1) If any rule evaluates to "Permit", the result is "Permit".
2) Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate" the result is "Indeterminate".
3) Otherwise, if any rule evaluates to "Deny", the result is "Deny".
4) Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is "Indeterminate".
5) Otherwise, the result is "NotApplicable".

The following pseudo-code represents the normative specification of this rule-combining algorithm.

```
Decision permitOverridesRuleCombiningAlgorithm(Rule[] rules)
{
   Boolean atLeastOneError  = false;
   Boolean potentialPermit  = false;
   Boolean atLeastOneDeny   = false;
   for( i=0 ; i < lengthOf(rules) ; i++ )
   {
          Decision decision = evaluate(rules[i]);
          if (decision == Deny)
          {
                 atLeastOneDeny = true;
                 continue;
          }
          if (decision == Permit)
          {
                 return Permit;
          }
```

```
          if (decision == NotApplicable)
          {
                continue;
          }
          if (decision == Indeterminate)
          {
                atLeastOneError = true;

                if (effect(rules[i]) == Permit)
                {
                      potentialPermit = true;
                }
                continue;
          }
   }
   if (potentialPermit)
   {
          return Indeterminate;
   }
   if (atLeastOneDeny)
   {
          return Deny;
   }
   if (atLeastOneError)
   {
          return Indeterminate;
   }
   return NotApplicable;
}
```

Obligations and advice of the individual rules shall be combined as described in clause 10.18.

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides`

The following is a non-normative informative description of this combining algorithm.

The "Permit-overrides" policy-combining algorithm is intended for those cases where a "Permit" decision should have priority over a "Deny" decision. This algorithm has the following behaviour.

1) If any policy evaluates to "Permit", the result is "Permit".
2) Otherwise, if any policy evaluates to "Deny", the result is "Deny".
3) Otherwise, if any policy evaluates to "Indeterminate", the result is "Indeterminate".
4) Otherwise, the result is "NotApplicable".

The following pseudo-code represents the normative specification of this policy-combining algorithm.

```
Decision permitOverridesPolicyCombiningAlgorithm(Policy[] policies)
{
   Boolean atLeastOneError = false;
   Boolean atLeastOneDeny  = false;
   for( i=0 ; i < lengthOf(policies) ; i++ )
   {
          Decision decision = evaluate(policies[i]);
          if (decision == Deny)
          {
                atLeastOneDeny = true;
                continue;
          }
          if (decision == Permit)
          {
                return Permit;
          }
          if (decision == NotApplicable)
          {
```

```
                continue;
          }
          if (decision == Indeterminate)
          {
                atLeastOneError = true;
                continue;
          }
    }
    if (atLeastOneDeny)
    {
          return Deny;
    }
    if (atLeastOneError)
    {
          return Indeterminate;
    }
    return NotApplicable;
}
```

Obligations and advice of the individual policies shall be combined as described in clause 10.18.

C.13 Legacy Ordered-permit-overrides

The following specification defines the legacy "Ordered-permit-overrides" rule-combining algorithm of a policy.

The behaviour of this algorithm is identical to that of the "Permit-overrides" rule-combining algorithm with one exception. The order in which the collection of rules is evaluated shall match the order as listed in the policy.

The rule-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides`

The following specification defines the legacy "Ordered-permit-overrides" policy-combining algorithm of a policy set.

The behaviour of this algorithm is identical to that of the "Permit-overrides" policy-combining algorithm with one exception. The order in which the collection of policies is evaluated shall match the order as listed in the policy set.

The policy-combining algorithm defined here has the following identifier:

`urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides`

Appendix I

Example

(This appendix does not form an integral part of this Recommendation.)

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

I.1 Example one

I.1.1 Example policy

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

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

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

```
[a1]    <?xml version="1.0" encoding="UTF-8"?>
[a2]    <Policy
[a3]      xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
[a4]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a5]      xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
[a6]      http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
[a7]      PolicyId="urn:oasis:names:tc:xacml:3.0:example:SimplePolicy1"
[a8]      Version="1.0"
[a9]      RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
[a10]     <Description>
[a11]       Medi Corp access control policy
[a12]     </Description>
[a13]     <Target/>
[a14]     <Rule
[a15]       RuleId= "urn:oasis:names:tc:xacml:3.0:example:SimpleRule1"
[a16]       Effect="Permit">
[a17]       <Description>
[a18]         Any subject with an e-mail name in the med.example.com domain
[a19]         can perform any action on any resource.
[a20]       </Description>
[a21]       <Target>
[a22]         <AnyOf>
[a23]           <AllOf>
[a24]             <Match
[a25]               MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
[a26]             <AttributeValue
[a27]               DataType="http://www.w3.org/2001/XMLSchema#string"
[a28]                 >med.example.com</AttributeValue>
[a29]             <AttributeDesignator
[a30]               MustBePresent="false"
[a31]               Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
[a32]               AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[a33]               DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
[a34]             </Match>
[a35]           </AllOf>
[a36]         </AnyOf>
[a37]       </Target>
[a38]     </Rule>
[a39]   </Policy>
```

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

[a2] introduces the XACML Policy itself.

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

[a3] gives URN for the XACML policies schema.

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

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

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

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

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

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

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

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

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

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

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

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

I.1.2 Example request context

Let us examine a hypothetical decision request that might be submitted to PDP that executes the policy above. In English, the access request that generates the decision request may be stated as follows:

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

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

```
[b1]    <?xml version="1.0" encoding="UTF-8"?>
[b2]    <Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
[b3]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[b4]      xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
        http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
[b5]      ReturnPolicyIdList="false">
[b6]      <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
        subject">
[b7]        <Attribute IncludeInResult="false"
[b8]          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
[b9]          <AttributeValue
[b10]           DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"
[b11]            >bs@simpsons.com</AttributeValue>
[b12]       </Attribute>
[b13]     </Attributes>
[b14]     <Attributes
[b15]       Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
[b16]       <Attribute IncludeInResult="false"
[b17]         AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
[b18]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[b19]           >file://example/med/record/patient/BartSimpson</AttributeValue>
[b20]       </Attribute>
[b21]     </Attributes>
[b22]     <Attributes
[b23]       Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
[b24]       <Attribute IncludeInResult="false"
[b25]           AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
[b26]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
[b27]           >read</AttributeValue>
[b28]       </Attribute>
[b29]     </Attributes>
[b30]   </Request>
```

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

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

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

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

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

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

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

I.1.3 Example response context

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

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

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

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

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

I.2 Example two

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

I.2.1 Example medical record instance

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

```
 [d1]    <?xml version="1.0" encoding="UTF-8"?>
 [d2]    <record xmlns="urn:example:med:schemas:record"
 [d3]    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 [d4]      <patient>
 [d5]        <patientName>
 [d6]          <first>Bartholomew</first>
 [d7]          <last>Simpson</last>
 [d8]        </patientName>
 [d9]        <patientContact>
[d10]          <street>27 Shelbyville Road</street>
[d11]          <city>Springfield</city>
[d12]          <state>MA</state>
[d13]          <zip>12345</zip>
[d14]          <phone>555.123.4567</phone>
[d15]          <fax/>
[d16]          <email/>
[d17]        </patientContact>
[d18]        <patientDoB>1992-03-21</patientDoB>
[d19]        <patientGender>male</patientGender>
[d20]        <patient-number>555555</patient-number>
[d21]      </patient>
[d22]      <parentGuardian>
[d23]        <parentGuardianId>HS001</parentGuardianId>
[d24]        <parentGuardianName>
[d25]          <first>Homer</first>
[d26]          <last>Simpson</last>
```

```
[d27]         </parentGuardianName>
[d28]         <parentGuardianContact>
[d29]           <street>27 Shelbyville Road</street>
[d30]           <city>Springfield</city>
[d31]           <state>MA</state>
[d32]           <zip>12345</zip>
[d33]           <phone>555.123.4567</phone>
[d34]           <fax/>
[d35]           <email>homers@aol.com</email>
[d36]         </parentGuardianContact>
[d37]       </parentGuardian>
[d38]       <primaryCarePhysician>
[d39]         <physicianName>
[d40]           <first>Julius</first>
[d41]           <last>Hibbert</last>
[d42]         </physicianName>
[d43]         <physicianContact>
[d44]           <street>1 First St</street>
[d45]           <city>Springfield</city>
[d46]           <state>MA</state>
[d47]           <zip>12345</zip>
[d48]           <phone>555.123.9012</phone>
[d49]           <fax>555.123.9013</fax>
[d50]           <email/>
[d51]         </physicianContact>
[d52]         <registrationID>ABC123</registrationID>
[d53]       </primaryCarePhysician>
[d54]       <insurer>
[d55]         <name>Blue Cross</name>
[d56]         <street>1234 Main St</street>
[d57]         <city>Springfield</city>
[d58]         <state>MA</state>
[d59]         <zip>12345</zip>
[d60]         <phone>555.123.5678</phone>
[d61]         <fax>555.123.5679</fax>
[d62]         <email/>
[d63]       </insurer>
[d64]       <medical>
[d65]         <treatment>
[d66]           <drug>
[d67]             <name>methylphenidate hydrochloride</name>
[d68]             <dailyDosage>30mgs</dailyDosage>
[d69]             <startDate>1999-01-12</startDate>
[d70]           </drug>
[d71]           <comment>
[d72]             patient exhibits side-effects of skin coloration and carpal degeneration
[d73]           </comment>
[d74]         </treatment>
[d75]         <result>
[d76]           <test>blood pressure</test>
[d77]           <value>120/80</value>
[d78]           <date>2001-06-09</date>
[d79]           <performedBy>Nurse Betty</performedBy>
[d80]         </result>
[d81]       </medical>
[d82]     </record>
```

I.2.2 Example request context

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

```
 [e1]    <?xml version="1.0" encoding="UTF-8"?>
 [e2]    <Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
 [e3]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 [e4]      xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
        http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
 [e5]       ReturnPolicyIdList="false">
 [e6]      <Attributes
 [e7]        Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
 [e8]        <Attribute IncludeInResult="false"
 [e9]            AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[e10]          Issuer="med.example.com">
[e11]          <AttributeValue
[e12]            DataType="http://www.w3.org/2001/XMLSchema#string">CN=Julius
        Hibbert</AttributeValue>
[e13]        </Attribute>
```

```
[e14]         <Attribute IncludeInResult="false"
[e15]           AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
[e16]           Issuer="med.example.com">
[e17]           <AttributeValue
[e18]             DataType="http://www.w3.org/2001/XMLSchema#string"
[e19]             >physician</AttributeValue>
[e20]           </Attribute>
[e21]         <Attribute IncludeInResult="false"
[e22]           AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id"
[e23]           Issuer="med.example.com">
[e24]           <AttributeValue
[e25]           DataType="http://www.w3.org/2001/XMLSchema#string">jh1234</AttributeValue>
[e26]         </Attribute>
[e27]       </Attributes>
[e28]       <Attributes
[e29]         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
[e30]         <Content>
[e31]           <md:record xmlns:md="urn:example:med:schemas:record"
[e32]             xsi:schemaLocation="urn:example:med:schemas:record
[e33]             http://www.med.example.com/schemas/record.xsd">
[e34]             <md:patient>
[e35]               <md:patientDoB>1992-03-21</md:patientDoB>
[e36]               <md:patient-number>555555</md:patient-number>
[e37]               <md:patientContact>
[e38]                 <md:email>b.simpson@example.com</md:email>
[e39]               </md:patientContact>
[e40]             </md:patient>
[e41]           </md:record>
[e42]         </Content>
[e43]         <Attribute IncludeInResult="false"
[e44]             AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector" >
[e45]           <AttributeValue
[e46]             XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[e47]             DataType=" urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
[e48]             >md:record/md:patient/md:patientDoB</AttributeValue>
[e49]         </Attribute>
[e50]         <Attribute IncludeInResult="false"
[e51]             AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace" >
[e52]           <AttributeValue
[e53]             DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[e54]             >urn:example:med:schemas:record</AttributeValue>
[e55]         </Attribute>
[e56]       </Attributes>
[e57]       <Attributes
[e58]         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
[e59]         <Attribute IncludeInResult="false"
[e60]              AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" >
[e61]           <AttributeValue
[e62]             DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
[e63]         </Attribute>
[e64]       </Attributes>
[e65]       <Attributes
[e66]         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment">
[e67]         <Attribute IncludeInResult="false"
[e68]              AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date" >
[e69]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date"
[e70]              >2010-01-11</AttributeValue>
[e71]         </Attribute>
[e72]       </Attributes>
[e73]     </Request>
```

[e2]-[e4] Standard namespace declarations.

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

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

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

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

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

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

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

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

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

I.2.3 Example plain-language rules

The following plain-language rules are to be enforced:

Rule 1: A person, identified by his or her patient number, may read any record for which he or she is the designated patient.

Rule 2: A person may read any record for which he or she is the designated parent or guardian, and for which the patient is under 16 years of age.

Rule 3: A physician may write to any medical element for which he or she is the designated primary care physician, provided an e-mail is sent to the patient.

Rule 4: An administrator shall not be permitted to read or write to medical elements of a patient record.

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

I.2.4 Example XACML rule instances

I.2.4.1 Rule 1

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

```
[f1]    <?xml version="1.0" encoding="UTF-8"?>
[f2]    <Policy
[f3]      xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
[f4]      xmlns:xacml ="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
[f5]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[f6]      xmlns:md="http://www.med.example.com/schemas/record.xsd"
[f7]      PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:1"
[f8]      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
        overrides"
[f9]      Version="1.0">
[f10]     <PolicyDefaults>
[f11]       <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
[f12]     </PolicyDefaults>
[f13]     <Target/>
[f14]     <VariableDefinition VariableId="17590034">
[f15]       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[f16]         <Apply
[f17]           FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[f18]           <AttributeDesignator
[f19]             MustBePresent="false"
[f20]             Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
[f21]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:patient-
        number"
[f22]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
[f23]         </Apply>
[f24]         <Apply
[f25]           FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[f26]           <AttributeSelector
[f27]               MustBePresent="false"
[f28]               Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
```

```
[f29]                Path="md:record/md:patient/md:patient-number/text()"
[f30]              DataType="http://www.w3.org/2001/XMLSchema#string"/>
[f31]          </Apply>
[f32]        </Apply>
[f33]      </VariableDefinition>
[f34]      <Rule
[f35]        RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
[f36]        Effect="Permit">
[f37]        <Description>
[f38]          A person may read any medical record in the
[f39]          http://www.med.example.com/schemas/record.xsd namespace
[f40]          for which he or she is the designated patient
[f41]        </Description>
[f42]        <Target>
[f43]          <AnyOf>
[f44]            <AllOf>
[f45]              <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
[f46]                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[f47]                 >urn:example:med:schemas:record</AttributeValue>
[f48]                <AttributeDesignator
[f49]                  MustBePresent="false"
[f50]                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[f51]                AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
[f52]                DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
[f53]              </Match>
[f54]              <Match
[f55]                MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
[f56]                <AttributeValue
[f57]                  DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
[f58]           XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[f59]                    >md:record</AttributeValue>
[f60]                <AttributeDesignator
[f61]                  MustBePresent="false"
[f62]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[f63]                 AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
[f64]                 DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
[f65]              </Match>
[f66]            </AllOf>
[f67]          </AnyOf>
[f68]          <AnyOf>
[f69]            <AllOf>
[f70]              <Match
[f71]                MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[f72]                <AttributeValue
[f73]                  DataType="http://www.w3.org/2001/XMLSchema#string"
[f74]                   >read</AttributeValue>
[f75]                <AttributeDesignator
[f76]                  MustBePresent="false"
[f77]                  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
[f78]                  AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[f79]                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
[f80]              </Match>
[f81]            </AllOf>
[f82]          </AnyOf>
[f83]        </Target>
[f84]        <Condition>
[f85]          <VariableReference VariableId="17590034"/>
[f86]        </Condition>
[f87]      </Rule>
[f88]    </Policy>
```

[f3]-[f6] XML namespace declarations.

[f11] XPath expressions in the ***policy*** are to be interpreted according to 1.0 version [W3C XPath].

[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***.

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

[f17] The first argument of the variable definition is a function specified by the `FunctionId` attribute. Since urn:oasis:names:tc:xacml:1.0:function:string-equal takes arguments of type "`http://www.w3.org/2001/XMLSchema#string`" and `AttributeDesignator` selects a ***bag*** of type

"http://www.w3.org/2001/XMLSchema#string",
"urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" is used. This function guarantees that its argument evaluates to a ***bag*** containing exactly one value.

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

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

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

[f35] ***Rule*** identifier.

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

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

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

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

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

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

[f45]The MatchId attribute names the matching function.

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

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

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

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

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

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

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

I.2.4.2 Rule 2

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

```
 [g1]    <?xml version="1.0" encoding="UTF-8"?>
 [g2]    <Policy
 [g3]      xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
 [g4]      xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
 [g5]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 [g6]      xmlns:xf="http://www.w3.org/2005/xpath-functions"
 [g7]      xmlns:md="http:www.med.example.com/schemas/record.xsd"
 [g8]      PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
 [g9]      Version="1.0"
[g10]      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
          overrides">
[g11]      <PolicyDefaults>
[g12]        <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
[g13]      </PolicyDefaults>
[g14]      <Target/>
[g15]      <VariableDefinition VariableId="17590035">
[g16]        <Apply
[g17]          FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal">
[g18]          <Apply
[g19]            FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
[g20]            <AttributeDesignator
[g21]              MustBePresent="false"
[g22]              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
[g23]              AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
[g24]              DataType="http://www.w3.org/2001/XMLSchema#date"/>
[g25]          </Apply>
[g26]          <Apply
[g27]      FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration">
[g28]            <Apply
[g29]              FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
[g30]              <AttributeSelector
[g31]                MustBePresent="false"
[g32]                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[g33]                Path="md:record/md:patient/md:patientDoB/text()"
[g34]                DataType="http://www.w3.org/2001/XMLSchema#date"/>
[g35]            </Apply>
[g36]            <AttributeValue
[g37]              DataType="http://www.w3.org/2001/XMLSchema#yearMonthDuration"
[g38]              >P16Y</AttributeValue>
[g39]          </Apply>
[g40]        </Apply>
[g41]      </VariableDefinition>
[g42]      <Rule
[g43]        RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
[g44]        Effect="Permit">
[g45]        <Description>
[g46]          A person may read any medical record in the
[g47]          http://www.med.example.com/records.xsd namespace
[g48]          for which he or she is the designated parent or guardian,
[g49]          and for which the patient is under 16 years of age
[g50]        </Description>
[g51]        <Target>
[g52]          <AnyOf>
[g53]            <AllOf>
[g54]              <Match
```

```
[g55]               MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
[g56]               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[g57]                 >urn:example:med:schemas:record</AttributeValue>
[g58]               <AttributeDesignator
[g59]                 MustBePresent="false"
[g60]                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[g61]              AttributeId= "urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
[g62]                 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
[g63]             </Match>
[g64]             <Match
[g65]               MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
[g66]               <AttributeValue
[g67]                 DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
[g68]          XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[g69]                 >md:record</AttributeValue>
[g70]               <AttributeDesignator
[g71]                 MustBePresent="false"
[g72]                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[g73]                 AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
[g74]                DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
[g75]             </Match>
[g76]           </AllOf>
[g77]         </AnyOf>
[g78]         <AnyOf>
[g79]           <AllOf>
[g80]             <Match
[g81]               MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[g82]               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
[g83]                  >read</AttributeValue>
[g84]               <AttributeDesignator
[g85]                 MustBePresent="false"
[g86]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
[g87]                 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[g88]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
[g89]             </Match>
[g90]           </AllOf>
[g91]         </AnyOf>
[g92]       </Target>
[g93]       <Condition>
[g94]         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
[g95]           <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[g96]             <Apply
[g97]              FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[g98]                <AttributeDesignator
[g99]                  MustBePresent="false"
[g100]             Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
[g101]                AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:parent-
    guardian-id"
[g102]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
[g103]             </Apply>
[g104]             <Apply
[g105]            FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[g106]               <AttributeSelector
[g107]                MustBePresent="false"
[g108]                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[g109]         Path="md:record/md:parentGuardian/md:parentGuardianId/text()"
[g110]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
[g111]             </Apply>
[g112]           </Apply>
[g113]           <VariableReference VariableId="17590035"/>
[g114]         </Apply>
[g115]       </Condition>
[g116]     </Rule>
[g117]   </Policy>
```

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

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

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

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

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

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

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

[g51]-[g92] ***Rule*** declaration and ***rule target***. See ***Rule*** 1 in clause I.2.4.1 for the detailed explanation of these elements.

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

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

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

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

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

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

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

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

I.2.4.3 Rule 3

Rule 3 illustrates the use of an obligation expression.

```
[h1]    <?xml version="1.0" encoding="UTF-8"?>
[h2]    <Policy
[h3]      xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
[h4]      xmlns:xacml ="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
[h5]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[h6]      xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
        http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
[h7]      xmlns:md="http:www.med.example.com/schemas/record.xsd"
[h8]      PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:3"
[h9]      Version="1.0"
[h10]     RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
        overrides">
[h11]     <Description>
[h12]       Policy for any medical record in the
[h13]       http://www.med.example.com/schemas/record.xsd namespace
[h14]     </Description>
[h15]     <PolicyDefaults>
[h16]       <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
[h17]     </PolicyDefaults>
[h18]     <Target>
[h19]       <AnyOf>
[h20]         <AllOf>
[h21]           <Match
[h22]             MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
[h23]             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[h24]               >urn:example:med:schemas:record</AttributeValue>
[h25]             <AttributeDesignator
[h26]               MustBePresent="false"
[h27]               Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[h28]               AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
[h29]               DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
[h30]           </Match>
[h31]         </AllOf>
[h32]       </AnyOf>
[h33]     </Target>
[h34]     <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:3"
[h35]       Effect="Permit">
[h36]       <Description>
[h37]         A physician may write any medical element in a record
[h38]         for which he or she is the designated primary care
[h39]         physician, provided an email is sent to the patient
[h40]       </Description>
[h41]       <Target>
[h42]         <AnyOf>
[h43]           <AllOf>
[h44]             <Match
[h45]               MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[h46]               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
[h47]                 >physician</AttributeValue>
[h48]               <AttributeDesignator
[h49]                 MustBePresent="false"
[h50]              Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
[h51]                 AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
[h52]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
[h53]             </Match>
[h54]           </AllOf>
[h55]         </AnyOf>
[h56]         <AnyOf>
[h57]           <AllOf>
[h58]             <Match
[h59]               MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
[h60]               <AttributeValue
[h61]                 DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
[h62]            XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[h63]                 >md:record/md:medical</AttributeValue>
[h64]               <AttributeDesignator
[h65]                 MustBePresent="false"
[h66]               Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[h67]                 AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
[h68]               DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
[h69]             </Match>
[h70]           </AllOf>
[h71]         </AnyOf>
[h72]         <AnyOf>
[h73]           <AllOf>
```

```
[h74]              <Match
[h75]                MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[h76]                <AttributeValue
[h77]                  DataType="http://www.w3.org/2001/XMLSchema#string"
[h78]                  >write</AttributeValue>
[h79]                <AttributeDesignator
[h80]                  MustBePresent="false"
[h81]                  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
[h82]                  AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[h83]                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
[h84]              </Match>
[h85]            </AllOf>
[h86]          </AnyOf>
[h87]        </Target>
[h88]        <Condition>
[h89]          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[h90]            <Apply
[h91]             FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[h92]              <AttributeDesignator
[h93]                MustBePresent="false"
[h94]               Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
[h95]          AttributeId="urn:oasis:names:tc:xacml:3.0:example: attribute:physician-id"
[h96]                DataType="http://www.w3.org/2001/XMLSchema#string"/>
[h97]            </Apply>
[h98]            <Apply
[h99]             FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[h100]             <AttributeSelector
[h101]                MustBePresent="false"
[h102]               Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[h103]     Path="md:record/md:primaryCarePhysician/md:registrationID/text()"
[h104]               DataType="http://www.w3.org/2001/XMLSchema#string"/>
[h105]           </Apply>
[h106]         </Apply>
[h107]       </Condition>
[h108]     </Rule>
[h109]     <ObligationExpressions>
[h110]       <ObligationExpression
           ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
[h111]         FulfillOn="Permit">
[h112]         <AttributeAssignmentExpression
[h113]           AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:mailto">
[h114]           <AttributeSelector
[h115]             MustBePresent="true"
[h116]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[h117]             Path="md:record/md:patient/md:patientContact/md:email"
[h118]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
[h119]         </AttributeAssignmentExpression>
[h120]         <AttributeAssignmentExpression
[h121]           AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
[h122]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
[h123]           >Your medical record has been accessed by:</AttributeValue>
[h124]         </AttributeAssignmentExpression>
[h125]         <AttributeAssignmentExpression
[h126]           AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
[h127]           <AttributeDesignator
[h128]             MustBePresent="false"
[h129]            Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
[h130]             AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[h131]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
[h132]         </AttributeAssignmentExpression>
[h133]       </ObligationExpression>
[h134]     </ObligationExpressions>
[h135]   </Policy>
```

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

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

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

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

[h18]-[h33] Policy target. The policy target defines a set of applicable decision requests. The structure of the <Target> element in the <Policy> is identical to the structure of the <Target> element in the <Rule>. In this case, the policy target is the set of all XML resources that conform to the namespace "urn:example:med:schemas:record".

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

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

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

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

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

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

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

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

[h110] The ObligationId attribute identifies the obligation. In this case, PEP is required to send an e-mail.

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

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

[h120]-[h123] The second parameter contains literal text for the e-mail body.

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

I.2.4.4 Rule 4

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

```
[i1]    <?xml version="1.0" encoding="UTF-8"?>
[i2]    <Policy
[i3]      xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
[i4]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[i5]      xmlns:md="http:www.med.example.com/schemas/record.xsd"
[i6]      PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:4"
[i7]      Version="1.0"
[i8]      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
        algorithm:deny-overrides">
[i9]      <PolicyDefaults>
[i10]       <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
```

```
[i11]      </PolicyDefaults>
[i12]      <Target/>
[i13]      <Rule
[i14]        RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
[i15]        Effect="Deny">
[i16]        <Description>
[i17]          An Administrator shall not be permitted to read or write
[i18]          medical elements of a patient record in the
[i19]          http://www.med.example.com/records.xsd namespace.
[i20]        </Description>
[i21]        <Target>
[i22]          <AnyOf>
[i23]            <AllOf>
[i24]              <Match
[i25]                MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[i26]               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
[i27]                >administrator</AttributeValue>
[i28]                <AttributeDesignator
[i29]                  MustBePresent="false"
[i30]          Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
[i31]                AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
[i32]                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
[i33]              </Match>
[i34]            </AllOf>
[i35]          </AnyOf>
[i36]          <AnyOf>
[i37]            <AllOf>
[i38]              <Match
[i39]                MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
[i40]               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[i41]                >urn:example:med:schemas:record</AttributeValue>
[i42]                <AttributeDesignator
[i43]                  MustBePresent="false"
[i44]              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[i45]             AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
[i46]                  DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
[i47]              </Match>
[i48]              <Match
[i49]                MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
[i50]                <AttributeValue
[i51]               DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
[i52]          XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[i53]                     >md:record/md:medical</AttributeValue>
[i54]                <AttributeDesignator
[i55]                   MustBePresent="false"
[i56]              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[i57]              AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
[i58]              DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
[i59]              </Match>
[i60]            </AllOf>
[i61]          </AnyOf>
[i62]          <AnyOf>
[i63]            <AllOf>
[i64]              <Match
[i65]                MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[i66]              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
[i67]                 >read</AttributeValue>
[i68]                <AttributeDesignator
[i69]                   MustBePresent="false"
[i70]                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
[i71]                  AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[i72]                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
[i73]              </Match>
[i74]            </AllOf>
[i75]            <AllOf>
[i76]              <Match
[i77]                MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[i78]               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
[i79]                >write</AttributeValue>
[i80]                <AttributeDesignator
[i81]                   MustBePresent="false"
[i82]                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
[i83]                  AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[i84]                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
[i85]              </Match>
[i86]            </AllOf>
[i87]          </AnyOf>
[i88]        </Target>
[i89]      </Rule>
[i90]    </Policy>
```

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

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

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

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

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

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

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

[i48]-[i59] The second `<Match>` element targets the rule at XML elements that match the XPath expression "/md:record/md:medical".

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

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

This rule does not have a `<Condition>` element.

I.2.4.5 Example PolicySet

This clause uses the examples of the previous clauses to illustrate the process of combining policies. The policy governing read access to medical elements of a record is formed from each of the four rules described in clause I.2.3. In plain language, the combined rule is:

Either the requestor is the patient; or

the requestor is the parent or guardian and the patient is under 16; or

the requestor is the primary care physician and a notification is sent to the patient; and

the requestor is not an administrator.

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

```
[j1]    <?xml version="1.0" encoding="UTF-8"?>
[j2]    <PolicySet
[j3]      xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
[j4]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[j5]      PolicySetId="urn:oasis:names:tc:xacml:3.0:example:policysetid:1"
[j6]      Version="1.0"
[j7]      PolicyCombiningAlgId=
[j8]      "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides">
[j9]      <Description>
[j10]       Example policy set.
[j11]     </Description>
[j12]     <Target>
[j13]       <AnyOf>
[j14]         <AllOf>
[j15]           <Match
```

```
[j16]              MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[j17]              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
[j18]                >urn:example:med:schema:records</AttributeValue>
[j19]              <AttributeDesignator
[j20]                MustBePresent="false"
[j21]                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[j22]                AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
[j23]                DataType="http://www.w3.org/2001/XMLSchema#string"/>
[j24]            </Match>
[j25]          </AllOf>
[j26]        </AnyOf>
[j27]      </Target>
[j28]      <PolicyIdReference>
[j29]        urn:oasis:names:tc:xacml:3.0:example:policyid:3
[j30]      </PolicyIdReference>
[j31]      <Policy
[j32]        PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
[j33]        RuleCombiningAlgId=
[j34]          "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides"
[j35]        Version="1.0">
[j36]        <Target/>
[j37]        <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
[j38]          Effect="Permit">
[j39]        </Rule>
[j40]        <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
[j41]          Effect="Permit">
[j42]        </Rule>
[j43]        <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
[j44]          Effect="Deny">
[j45]        </Rule>
[j46]      </Policy>
[j47]    </PolicySet>
```

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

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

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

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

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

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

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

Appendix II

XACML extensibility points

(This appendix does not form an integral part of this Recommendation.)

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

II.1 Extensible XML attribute types

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

`Category`,

`AttributeId`,

`DataType`,

`FunctionId`,

`MatchId`,

`ObligationId`,

`AdviceId`,

`PolicyCombiningAlgId`,

`RuleCombiningAlgId`,

`StatusCode`,

See clause 8 for definitions of these ***attribute*** types.

II.2 Structured attributes

`<AttributeValue>` elements may contain an instance of a structured XML data-type. Clause 10.3.1 describes a number of standard techniques to identify data items within such a structured attribute. Listed here are some additional techniques that require XACML extensions.

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

Appendix III

Security and privacy considerations

(This appendix does not form an integral part of this Recommendation.)

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

III.1 Threat model

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

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

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

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

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

III.1.1 Unauthorized disclosure

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

Unauthorized disclosure is addressed by confidentiality safeguards.

III.1.2 Message replay

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

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

NOTE – Encryption of the message does not mitigate a replay attack since the message is simply replayed and does not have to be understood by the adversary.

III.1.3 Message insertion

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

The solution to a message insertion attack is to use mutual authentication and message sequence integrity safeguards between the actors. It should be noted that just using SSL mutual authentication is not sufficient. This only proves that the other party is the one identified by the subject of the ITU-T X.509 certificate. In order to be effective, it is necessary to confirm that the certificate subject is authorized to send the message.

III.1.4 Message deletion

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

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

III.1.5 Message modification

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

III.1.6 NotApplicable results

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

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

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

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

It may be safe to treat "NotApplicable" as "Permit" only in a closed environment where all applications that formulate a decision request can be guaranteed to use the exact syntax expected by the policies. In a more open environment, where decision requests may be received from applications that use any legal syntax, it is strongly recommended that "NotApplicable" NOT be treated as "Permit" unless matching rules have been very carefully designed to match all possible applicable inputs, regardless of syntax or type variations. Note, however, that according to clause 10.2, PEP must deny access unless it receives an explicit "Permit" authorization decision.

III.1.7 Negative rules

A negative rule is one that is based on a predicate not being "True". If not used with care, negative rules can lead to policy violations, therefore some authorities recommend that they not be used.

However, negative rules can be extremely efficient in certain cases, so XACML has chosen to include them. Nevertheless, it is recommended that they be used with care and avoided if possible.

A common use for negative rules is to deny access to an individual or subgroup when their membership in a larger group would otherwise permit them access. For example, to write a rule that allows all vice presidents to see the unpublished financial data, except for Joe, who is only a ceremonial vice president and can be indiscreet in his communications. If to complete control over the administration of subject attributes, a superior approach would be to define "Vice President" and "Ceremonial Vice President" as distinct groups and then define rules accordingly. However, in some environments this approach may not be feasible. (It is worth noting in passing that referring to individuals in rules does not scale well. Generally, shared attributes are preferred.)

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

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

III.1.8 Denial of service

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

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

III.2 Safeguards

III.2.1 Authentication

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

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

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

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

III.2.2 Policy administration

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

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

III.2.3 Confidentiality

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

Communication confidentiality

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

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

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

Statement level confidentiality

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

The suitable Recommendation [b-XMLEncSynPro] for this can be used to encrypt all or parts of an XML document. [b-XMLEncSynPro] is recommended for use with XACML.

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

III.2.4 Policy integrity

The XACML policy used by PDP to evaluate the request context is the heart of the system. Therefore, maintaining its integrity is essential. There are two aspects to maintaining the integrity of the policy. One is to ensure that `<Policy>` elements have not been altered since they were originally created by PAP. The other is to ensure that `<Policy>` elements have not been inserted or deleted from the set of policies.

In many cases, both aspects can be achieved by ensuring the integrity of the actors and implementing session-level mechanisms to secure the communication between actors. The selection of the

appropriate mechanisms is left to the implementers. However, when policy is distributed between organizations to be acted on at a later time, or when the policy travels with the protected resource, it would be useful to sign the policy. In these cases, [W3C DS] is recommended to be used with XACML.

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

III.2.5 Policy identifiers

Since policies can be referenced by their identifiers, it is the responsibility of PAP to ensure that these are unique. Confusion between identifiers could lead to misidentification of the applicable policy. This Recommendation is neutral on whether PAP must generate a new identifier when a policy is modified or may use the same identifier in the modified policy. This is a matter of administrative practice. However, care must be taken in either case. If the identifier is reused, there is a danger that other policies or policy sets that reference it may be adversely affected. Conversely, if a new identifier is used, these other policies may continue to use the previous policy, unless it is deleted. In either case, the results may not be what the policy administrator intends.

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

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

III.2.6 Trust model

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

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

None of the entities of the authorization system are dependent on PEP. They may collect data from it, (for example authentication data) but are responsible for verifying it themselves.

The correct operation of the system depends on the ability of PEP to actually enforce policy decisions.

PEP depends on PDP to correctly evaluate policies. This in turn implies that PDP is supplied with the correct inputs. Other than that, PDP does not depend on PEP.

PDP depends on PAP to supply appropriate policies. PAP is not dependent on other components.

III.2.7 Privacy

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

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

III.3 Unicode security issues

There are many security considerations related to the use of Unicode. An XACML implementation should follow the advice given in the relevant version of [b-UTR36].

III.4 Identifier equality

Clause 10.20 defines the identifier equality operation for XACML. This definition of equality does not do any kind of canonicalization or escaping of characters. The identifiers defined in this Recommendation have been selected to exclude any ambiguity regarding these aspects. It is recommended that identifiers defined by extensions also do not introduce any identifiers which might be mistaken for being subject to processing, for instance, URL character encoding using "%".

Appendix IV

Schema

(This appendix does not form an integral part of this Recommendation.)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright OASIS Open 2010. All Rights Reserved. -->
<xs:schema xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    elementFormDefault="qualified" attributeFormDefault="unqualified">

    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
            schemaLocation="http://www.w3.org/2001/xml.xsd"/>

    <xs:element name="Request" type="xacml:RequestType"/>
    <xs:complexType name="RequestType">
            <xs:sequence>
                    <xs:element ref="xacml:RequestDefaults" minOccurs="0"/>
                    <xs:element ref="xacml:Attributes" maxOccurs="unbounded"/>
                    <xs:element ref="xacml:MultiRequests" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="ReturnPolicyIdList" type="xs:boolean" use="required" />
            <xs:attribute name="CombinedDecision" type="xs:boolean" use="required" />
    </xs:complexType>

    <xs:element name="RequestDefaults" type="xacml:RequestDefaultsType"/>
    <xs:complexType name="RequestDefaultsType">
            <xs:sequence>
                    <xs:choice>
                            <xs:element ref="xacml:XPathVersion"/>
                    </xs:choice>
            </xs:sequence>
    </xs:complexType>

    <xs:element name="Response" type="xacml:ResponseType"/>
    <xs:complexType name="ResponseType">
            <xs:sequence>
                    <xs:element ref="xacml:Result" maxOccurs="unbounded"/>
            </xs:sequence>
    </xs:complexType>

    <xs:element name="Content" type="xacml:ContentType"/>
    <xs:complexType name="ContentType" mixed="true">
            <xs:sequence>
                    <xs:any namespace="##any" processContents="lax"/>
            </xs:sequence>
    </xs:complexType>

    <xs:element name="Result" type="xacml:ResultType"/>
    <xs:complexType name="ResultType">
            <xs:sequence>
                    <xs:element ref="xacml:Decision"/>
                    <xs:element ref="xacml:Status" minOccurs="0"/>
                    <xs:element ref="xacml:Obligations" minOccurs="0"/>
                    <xs:element ref="xacml:AssociatedAdvice" minOccurs="0"/>
                    <xs:element ref="xacml:Attributes" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="xacml:PolicyIdentifierList" minOccurs="0"/>
            </xs:sequence>
    </xs:complexType>

    <xs:element name="PolicyIdentifierList" type="xacml:PolicyIdentifierListType"/>
    <xs:complexType name="PolicyIdentifierListType">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element ref="xacml:PolicyIdReference"/>
                    <xs:element ref="xacml:PolicySetIdReference"/>
```

```
        </xs:choice>
</xs:complexType>

<xs:element name="Decision" type="xacml:DecisionType"/>
<xs:simpleType name="DecisionType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="Permit"/>
                <xs:enumeration value="Deny"/>
                <xs:enumeration value="Indeterminate"/>
                <xs:enumeration value="NotApplicable"/>
        </xs:restriction>
</xs:simpleType>

<xs:element name="Status" type="xacml:StatusType"/>
<xs:complexType name="StatusType">
        <xs:sequence>
                <xs:element ref="xacml:StatusCode"/>
                <xs:element ref="xacml:StatusMessage" minOccurs="0"/>
                <xs:element ref="xacml:StatusDetail" minOccurs="0"/>
        </xs:sequence>
</xs:complexType>

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

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

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

<xs:element name="MissingAttributeDetail" type="xacml:MissingAttributeDetailType"/>
<xs:complexType name="MissingAttributeDetailType">
        <xs:sequence>
                <xs:element ref="xacml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Category" type="xs:anyURI" use="required"/>
        <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
        <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
        <xs:attribute name="Issuer" type="xs:string" use="optional"/>
</xs:complexType>

<xs:element name="Attributes" type="xacml:AttributesType"/>
<xs:complexType name="AttributesType">
        <xs:sequence>
                <xs:element ref="xacml:Content" minOccurs="0"/>
                <xs:element ref="xacml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Category" type="xs:anyURI" use="required"/>
        <xs:attribute ref="xml:id" use="optional"/>
</xs:complexType>

<xs:element name="Attribute" type="xacml:AttributeType"/>
<xs:complexType name="AttributeType">
        <xs:sequence>
                <xs:element ref="xacml:AttributeValue" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
        <xs:attribute name="Issuer" type="xs:string" use="optional"/>
        <xs:attribute name="IncludeInResult" type="xs:boolean" use="required"/>
</xs:complexType>
```

```
<xs:element name="MultiRequests" type="xacml:MultiRequestsType"/>
<xs:complexType name="MultiRequestsType">
	<xs:sequence>
		<xs:element ref="xacml:RequestReference" maxOccurs="unbounded"/>
	</xs:sequence>
</xs:complexType>

<xs:element name="RequestReference" type="xacml:RequestReferenceType"/>
<xs:complexType name="RequestReferenceType">
	<xs:sequence>
		<xs:element ref="xacml:AttributesReference" maxOccurs="unbounded"/>
	</xs:sequence>
</xs:complexType>

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

<xs:element name="Obligations" type="xacml:ObligationsType"/>
<xs:complexType name="ObligationsType">
	<xs:sequence>
		<xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
	</xs:sequence>
</xs:complexType>

<xs:element name="AssociatedAdvice" type="xacml:AssociatedAdviceType"/>
<xs:complexType name="AssociatedAdviceType">
	<xs:sequence>
		<xs:element ref="xacml:Advice" maxOccurs="unbounded"/>
	</xs:sequence>
</xs:complexType>

<xs:element name="Obligation" type="xacml:ObligationType"/>
<xs:complexType name="ObligationType">
	<xs:sequence>
		<xs:element ref="xacml:AttributeAssignment" minOccurs="0" maxOccurs="unbounded"/>
	</xs:sequence>
	<xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
</xs:complexType>

<xs:element name="Advice" type="xacml:AdviceType"/>
<xs:complexType name="AdviceType">
	<xs:sequence>
		<xs:element ref="xacml:AttributeAssignment" minOccurs="0" maxOccurs="unbounded"/>
	</xs:sequence>
	<xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>
</xs:complexType>

<xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
<xs:complexType name="AttributeAssignmentType" mixed="true">
	<xs:complexContent mixed="true">
		<xs:extension base="xacml:AttributeValueType">
			<xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
			<xs:attribute name="Category" type="xs:anyURI" use="optional"/>
			<xs:attribute name="Issuer" type="xs:string" use="optional"/>
		</xs:extension>
	</xs:complexContent>
</xs:complexType>

<xs:element name="ObligationExpressions" type="xacml:ObligationExpressionsType"/>
<xs:complexType name="ObligationExpressionsType">
	<xs:sequence>
		<xs:element ref="xacml:ObligationExpression" maxOccurs="unbounded"/>
	</xs:sequence>
</xs:complexType>

<xs:element name="AdviceExpressions" type="xacml:AdviceExpressionsType"/>
<xs:complexType name="AdviceExpressionsType">
	<xs:sequence>
```

```
                        <xs:element ref="xacml:AdviceExpression" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>

        <xs:element name="ObligationExpression" type="xacml:ObligationExpressionType"/>
        <xs:complexType name="ObligationExpressionType">
                <xs:sequence>
                        <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
                <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
        </xs:complexType>

        <xs:element name="AdviceExpression" type="xacml:AdviceExpressionType"/>
        <xs:complexType name="AdviceExpressionType">
                <xs:sequence>
                        <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>
                <xs:attribute name="AppliesTo" type="xacml:EffectType" use="required"/>
        </xs:complexType>

        <xs:element name="AttributeAssignmentExpression" type="xacml:AttributeAssignmentExpressionType"/>
        <xs:complexType name="AttributeAssignmentExpressionType">
                <xs:sequence>
                        <xs:element ref="xacml:Expression"/>
                </xs:sequence>
                <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
                <xs:attribute name="Category" type="xs:anyURI" use="optional"/>
                <xs:attribute name="Issuer" type="xs:string" use="optional"/>
        </xs:complexType>

        <xs:simpleType name="EffectType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Permit"/>
                        <xs:enumeration value="Deny"/>
                </xs:restriction>
        </xs:simpleType>

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

        <xs:element name="PolicyIssuer" type="xacml:PolicyIssuerType"/>
        <xs:complexType name="PolicyIssuerType">
                <xs:sequence>
           <xs:element ref="xacml:Content" minOccurs="0"/>
```

```
                        <xs:element ref="xacml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>

        <xs:element name="CombinerParameters" type="xacml:CombinerParametersType"/>
        <xs:complexType name="CombinerParametersType">
                <xs:sequence>
                        <xs:element ref="xacml:CombinerParameter" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>

        <xs:element name="CombinerParameter" type="xacml:CombinerParameterType"/>
        <xs:complexType name="CombinerParameterType">
                <xs:sequence>
                        <xs:element ref="xacml:AttributeValue"/>
                </xs:sequence>
                <xs:attribute name="ParameterName" type="xs:string" use="required"/>
        </xs:complexType>

        <xs:element name="RuleCombinerParameters" type="xacml:RuleCombinerParametersType"/>
        <xs:complexType name="RuleCombinerParametersType">
                <xs:complexContent>
                        <xs:extension base="xacml:CombinerParametersType">
                                <xs:attribute name="RuleIdRef" type="xs:string" use="required"/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>

        <xs:element name="PolicyCombinerParameters" type="xacml:PolicyCombinerParametersType"/>
        <xs:complexType name="PolicyCombinerParametersType">
                <xs:complexContent>
                        <xs:extension base="xacml:CombinerParametersType">
                                <xs:attribute name="PolicyIdRef" type="xs:anyURI" use="required"/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>

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

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

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

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

        <xs:complexType name="IdReferenceType">
                <xs:simpleContent>
                        <xs:extension base="xs:anyURI">
                                <xs:attribute name="Version" type="xacml:VersionMatchType" use="optional"/>
                                <xs:attribute name="EarliestVersion" type="xacml:VersionMatchType"
use="optional"/>
                                <xs:attribute name="LatestVersion" type="xacml:VersionMatchType"
use="optional"/>
                        </xs:extension>
```

```
		</xs:simpleContent>
</xs:complexType>

<xs:simpleType name="VersionType">
		<xs:restriction base="xs:string">
			<xs:pattern value="(\d+\.)*\d+"/>
		</xs:restriction>
</xs:simpleType>

<xs:simpleType name="VersionMatchType">
		<xs:restriction base="xs:string">
			<xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)"/>
		</xs:restriction>
</xs:simpleType>

<xs:element name="Policy" type="xacml:PolicyType"/>
<xs:complexType name="PolicyType">
		<xs:sequence>
			<xs:element ref="xacml:Description" minOccurs="0"/>
			<xs:element ref="xacml:PolicyIssuer" minOccurs="0"/>
			<xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
			<xs:element ref="xacml:Target"/>
			<xs:choice maxOccurs="unbounded">
				<xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
				<xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>
				<xs:element ref="xacml:VariableDefinition"/>
				<xs:element ref="xacml:Rule"/>
			</xs:choice>
			<xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
			<xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
		</xs:sequence>
		<xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
		<xs:attribute name="Version" type="xacml:VersionType" use="required"/>
		<xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
		<xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
</xs:complexType>

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

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

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

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

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

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

    <xs:element name="Match" type="xacml:MatchType"/>
    <xs:complexType name="MatchType">
            <xs:sequence>
                    <xs:element ref="xacml:AttributeValue"/>
                    <xs:choice>
                            <xs:element ref="xacml:AttributeDesignator"/>
                            <xs:element ref="xacml:AttributeSelector"/>
                    </xs:choice>
            </xs:sequence>
            <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
    </xs:complexType>

    <xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>
    <xs:complexType name="VariableDefinitionType">
            <xs:sequence>
                    <xs:element ref="xacml:Expression"/>
            </xs:sequence>
            <xs:attribute name="VariableId" type="xs:string" use="required"/>
    </xs:complexType>

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

    <xs:element name="VariableReference" type="xacml:VariableReferenceType"
substitutionGroup="xacml:Expression"/>
    <xs:complexType name="VariableReferenceType">
            <xs:complexContent>
                    <xs:extension base="xacml:ExpressionType">
                            <xs:attribute name="VariableId" type="xs:string" use="required"/>
                    </xs:extension>
            </xs:complexContent>
    </xs:complexType>

    <xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"
substitutionGroup="xacml:Expression"/>
    <xs:complexType name="AttributeSelectorType">
            <xs:complexContent>
                    <xs:extension base="xacml:ExpressionType">
                            <xs:attribute name="Category" type="xs:anyURI" use="required"/>
                            <xs:attribute name="ContextSelectorId" type="xs:anyURI" use="optional"/>
                            <xs:attribute name="Path" type="xs:string" use="required"/>
                            <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
                            <xs:attribute name="MustBePresent" type="xs:boolean" use="required"/>
                    </xs:extension>
            </xs:complexContent>
    </xs:complexType>

    <xs:element name="AttributeDesignator" type="xacml:AttributeDesignatorType"
substitutionGroup="xacml:Expression"/>
    <xs:complexType name="AttributeDesignatorType">
            <xs:complexContent>
                    <xs:extension base="xacml:ExpressionType">
                            <xs:attribute name="Category" type="xs:anyURI" use="required"/>
                            <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
                            <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
                            <xs:attribute name="Issuer" type="xs:string" use="optional"/>
                            <xs:attribute name="MustBePresent" type="xs:boolean" use="required"/>
                    </xs:extension>
            </xs:complexContent>
    </xs:complexType>

    <xs:element name="AttributeValue" type="xacml:AttributeValueType" substitutionGroup="xacml:Expression"/>
    <xs:complexType name="AttributeValueType" mixed="true">
            <xs:complexContent mixed="true">
                    <xs:extension base="xacml:ExpressionType">
                            <xs:sequence>
```

```
                                        <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                                </xs:sequence>
                                <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
                                <xs:anyAttribute namespace="##any" processContents="lax"/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>

        <xs:element name="Function" type="xacml:FunctionType" substitutionGroup="xacml:Expression"/>
        <xs:complexType name="FunctionType">
                <xs:complexContent>
                        <xs:extension base="xacml:ExpressionType">
                                <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>

        <xs:element name="Condition" type="xacml:ConditionType"/>
        <xs:complexType name="ConditionType">
                <xs:sequence>
                        <xs:element ref="xacml:Expression"/>
                </xs:sequence>
        </xs:complexType>

        <xs:element name="Apply" type="xacml:ApplyType" substitutionGroup="xacml:Expression"/>
        <xs:complexType name="ApplyType">
                <xs:complexContent>
                        <xs:extension base="xacml:ExpressionType">
                                <xs:sequence>
                                        <xs:element ref="xacml:Description" minOccurs="0"/>
                                        <xs:element ref="xacml:Expression" minOccurs="0"
maxOccurs="unbounded"/>
                                </xs:sequence>
                                <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:schema>
```

Bibliography

[b-ITU-T X.520] Recommendation ITU-T X.520 (2012), *Information technology – Open Systems Interconnection – The Directory: Selected attribute types*.

[b-ITU-T X.811] Recommendation ITU-T X.811 (1995)| ISO/IEC 10181-2:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Authentication framework.*

[b-ITU-T X.812] Recommendation ITU-T X.812 (1995)| ISO/IEC 10181-3:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework.*

[b-ITU-T X.1142] Recommendation ITU-T X.1142 (2006), *eXtensible Access Control Markup Language (XACML 2.0)*.

[b-IETF RFC 3986] IETF RFC 3986 (2005), *Uniform Resource Identifiers (URI): Generic Syntax*. <http://www.ietf.org/rfc/rfc3986.txt>

[b-IETF RFC 4949] IETF RFC 4949 (2007), *Internet Security Glossary, Version 2*. <http://www.ietf.org/rfc/rfc4949.txt>

[b-W3C Glossary] W3C Glossary and Dictionary (2003), *Glossary and Dictionary*. <http://www.w3.org/2003/glossary/>

[b-W3C Technology] W3C XMLTechnology. <http://www.w3.org/standards/xml>

[b-W3C XMLid] W3C Recommendation (9 September 2005), *xml:id Version 1.0*. <http://www.w3.org/TR/2005/REC-xml-id-20050909/>

[b-XACMLAdmin] OASIS Committee Draft 03 (11 March 2010), *XACML v3.0 Administration and Delegation Profile Version 1.0*. <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-03-en.doc>

[b-OASIS] OASIS (12 July 2017), eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01 <https://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-complete.pdf >

[b-CM] W3C Working Draft (27 October 2005), *Character model for the World Wide Web 1.0: Normalization*. <http://www.w3.org/TR/2005/WD-charmod-norm-20051027/>

[b-CMF] W3C Recommendation (15 February 2005), *Character Model for the World Wide Web 1.0: Fundamentals*. <http://www.w3.org/TR/2005/REC-charmod-20050215/>

[b-Hancock] Hancock (1987), *Polymorphic Type Checking, in Simon L. Peyton Jones, Implementation of Functional Programming Languages, Section 8, Prentice-Hall International*.

[b-Hier] OASIS Committee Draft 03 (11 March 2010), *XACML v3.0 Hierarchical Resource Profile Version 1.0*.

[b-Hinton] Heather M. Hinton, E. Stewart Lee (1994), *The Compatibility of Policies, Proceedings 2nd ACM Conference on Computer and Communications Security*, Nov., Fairfax, Virginia, USA.

[b-Kudo] Kudo, M., and Hada, S. (2000), *XML document security based on provisional authorization, Proceedings of the Seventh ACM Conference on Computer and Communications Security*, Nov., Athens, Greece, pp. 87-96.

[b-Multi] OASIS Committee Draft 03 (11 March 2010), *XACML v3.0 Multiple Decision Profile Version 1.0*.
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-multiple-v1-spec-cd-03-en.doc>

[b-Perritt] Perritt, H. Knowbots (1993), *Permissions Headers and Contract Law, Conference on Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment*, April.
<http://www.ifla.org/documents/infopol/copyright/perh2.txt>

[b-RBAC] ANSI INCITS 359-2004, *Information technology – Role Based Access Control*.
<http://csrc.nist.gov/rbac/>

[b-Sloman] Sloman, M. (1994), *Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, Volume 2, part 4. Plenum Press*.

[b-UAX15] Mark Davis and Ken Whistler, *Unicode Standard Annex #15: Unicode Normalization Forms, Unicode 5.1*.
<http://unicode.org/reports/tr15/>

[b-UTR36] Mark Davis and Michel Suignard, *Unicode Technocal Report #36: Unicode Security Considerations*.
<http://www.unicode.org/reports/tr36/>

[b-XMLEncSynPro] W3C Candidate Recommendation (04 March 2002), *XML Encryption Syntax and Processing*.
<http://www.w3.org/TR/2002/CR-xmlenc-core-20020304/>

[b-XSLT] W3C Recommendation (16 November 1999), *XSL Transformations (XSLT) Version 1.0*.
<http://www.w3.org/TR/xslt>

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	**Data networks, open system communications and security**
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems

Published in Switzerland
Geneva, 2024