

International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.1144

(10/2013)

SERIES X: DATA NETWORKS, OPEN SYSTEM
COMMUNICATIONS AND SECURITY

Secure applications and services – Web security

eXtensible Access Control Markup Language (XACML 3.0)

Recommendation ITU-T X.1144

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	
General security aspects	X.1000–X.1029
Network security	X.1030–X.1049
Security management	X.1050–X.1069
Telebiometrics	X.1080–X.1099
SECURE APPLICATIONS AND SERVICES	
Multicast security	X.1100–X.1109
Home network security	X.1110–X.1119
Mobile security	X.1120–X.1139
Web security	X.1140–X.1149
Security protocols	X.1150–X.1159
Peer-to-peer security	X.1160–X.1169
Networked ID security	X.1170–X.1179
IPTV security	X.1180–X.1199
CYBERSPACE SECURITY	
Cybersecurity	X.1200–X.1229
Countering spam	X.1230–X.1249
Identity management	X.1250–X.1279
SECURE APPLICATIONS AND SERVICES	
Emergency communications	X.1300–X.1309
Ubiquitous sensor network security	X.1310–X.1339
CYBERSECURITY INFORMATION EXCHANGE	
Overview of cybersecurity	X.1500–X.1519
Vulnerability/state exchange	X.1520–X.1539
Event/incident/heuristics exchange	X.1540–X.1549
Exchange of policies	X.1550–X.1559
Heuristics and information request	X.1560–X.1569
Identification and discovery	X.1570–X.1579
Assured exchange	X.1580–X.1589

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

Recommendation ITU-T X.1144

eXtensible Access Control Markup Language (XACML 3.0)

Summary

Recommendation ITU-T X.1144 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)).

This Recommendation 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 3.0 standard.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T X.1144	2013-10-14	17	11.1002/1000/12044

* To access the Recommendation, type the URL <http://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID. For example, <http://handle.itu.int/11.1002/1000/11830-en>.

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, 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 TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2014

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

		Page
1	Scope	1
2	References.....	1
3	Definitions	3
	3.1 Terms defined elsewhere	3
	3.2 Terms defined in this Recommendation.....	4
4	Abbreviations and acronyms	6
5	Conventions	7
6	Overview	7
	6.1 Requirements.....	8
	6.2 Rule and policy combining.....	8
	6.3 Combining algorithms	9
	6.4 Multiple subjects	9
	6.5 Policies based on subject and resource attributes.....	9
	6.6 Multi-valued attributes	10
	6.7 Policies based on resource contents	10
	6.8 Operators	10
	6.9 Policy distribution	11
	6.10 Policy indexing.....	11
	6.11 Abstraction layer	12
	6.12 Actions performed in conjunction with enforcement.....	12
	6.13 Supplemental information about a decision	12
7	XACML models	12
	7.1 Data-flow model.....	12
	7.2 XACML context.....	14
	7.3 Policy language model	14
8	Syntax	18
	8.1 Element <PolicySet>	18
	8.2 Element <Description>	21
	8.3 Element <PolicyIssuer>	21
	8.4 Element <PolicySetDefaults>	21
	8.5 Element <XPathVersion>.....	22
	8.6 Element <Target>	22
	8.7 Element <AnyOf>	22
	8.8 Element <AllOf>	23
	8.9 Element <Match>	23
	8.10 Element <PolicySetIdReference>	24
	8.11 Element <PolicyIdReference>	24

	Page
8.12 Simple type VersionType	24
8.13 Simple type VersionMatchType	25
8.14 Element <Policy>	25
8.15 Element <PolicyDefaults>	27
8.16 Element <CombinerParameters>	27
8.17 Element <CombinerParameter>	28
8.18 Element <RuleCombinerParameters>	28
8.19 Element <PolicyCombinerParameters>	29
8.20 Element <PolicySetCombinerParameters>	29
8.21 Element <Rule>	30
8.22 Simple type EffectType	31
8.23 Element <VariableDefinition>	31
8.24 Element <VariableReference>	32
8.25 Element <Expression>	32
8.26 Element <Condition>	32
8.27 Element <Apply>	32
8.28 Element <Function>	33
8.29 Element <AttributeDesignator>	33
8.30 Element <AttributeSelector>	34
8.31 Element <AttributeValue>	35
8.32 Element <Obligations>	36
8.33 Element <AssociatedAdvice>	36
8.34 Element <Obligation>	36
8.35 Element <Advice>	37
8.36 Element <AttributeAssignment>	37
8.37 Element <ObligationExpressions>	38
8.38 Element <AdviceExpressions>	38
8.39 Element <ObligationExpression>	38
8.40 Element <AdviceExpression>	39
8.41 Element <AttributeAssignmentExpression>	40
8.42 Element <Request>	40
8.43 Element <RequestDefaults>	41
8.44 Element <Attributes>	42
8.45 Element <Content>	42
8.46 Element <Attribute>	43
8.47 Element <Response>	43
8.48 Element <Result>	44
8.49 Element <PolicyIdentifierList>	45
8.50 Element <MultiRequests>	45
8.51 Element <RequestReference>	45

	Page
8.52 Element <AttributesReference>	46
8.53 Element <Decision>	46
8.54 Element <Status>	46
8.55 Element <StatusCode>	47
8.56 Element <StatusMessage>	47
8.57 Element <StatusDetail>	47
8.58 Element <MissingAttributeDetail>	48
9 XPath 2.0 definitions	49
10 Functional requirements	50
10.1 Unicode issues	50
10.2 Policy enforcement point.....	51
10.3 Attribute evaluation	52
10.4 Expression evaluation.....	55
10.5 Arithmetic evaluation	55
10.6 Match evaluation	55
10.7 Target evaluation	56
10.8 VariableReference evaluation	57
10.9 Condition evaluation	58
10.10 Extended "indeterminate"	58
10.11 Rule evaluation.....	58
10.12 Policy evaluation	58
10.13 Policy set evaluation.....	59
10.14 Policy and policy set value for i "Indeterminate" target.....	59
10.15 PolicySetIdReference and PolicyIdReference evaluation	60
10.16 Hierarchical resources	60
10.17 Authorization decision.....	60
10.18 Obligations and advice	60
10.19 Exception handling.....	61
10.20 Identifier equality	61
11 Conformance	63
Annex A – Data-types and functions	72
A.1 Introduction	72
A.2 Data-types.....	72
A.3 Functions	74
A.4 Functions, data-types, attributes and algorithms planned for deprecation	97
Annex B – XACML identifiers.....	100
B.1 XACML namespaces.....	100
B.2 Attribute categories	100
B.3 Data-types.....	101

	Page
B.4 Subject attributes	101
B.5 Resource attributes	102
B.6 Action attributes	102
B.7 Environment attributes	103
B.8 Status codes	103
B.9 Combining algorithms	103
Annex C – Combining algorithms	106
C.1 Extended "Indeterminate" values	106
C.2 Deny-overrides	106
C.3 Ordered-deny-overrides.....	108
C.4 Permit-overrides	108
C.5 Ordered-permit-overrides	110
C.6 Deny-unless-permit	110
C.7 Permit-unless-deny	111
C.8 First-applicable	111
C.9 Only-one-applicable	113
C.10 Legacy Deny-overrides	114
C.11 Legacy Ordered-deny-overrides.....	116
C.12 Legacy Permit-overrides	116
C.13 Legacy Ordered-permit-overrides	118
Appendix I – Example	119
I.1 Example one	119
I.2 Example two.....	122
Appendix II – XACML extensibility points	137
II.1 Extensible XML attribute types.....	137
II.2 Structured attributes.....	137
Appendix III – Security and privacy considerations.....	138
III.1 Threat model.....	138
III.2 Safeguards	140
III.3 Unicode security issues	143
III.4 Identifier equality	143
Appendix IV – Schema	144
Bibliography.....	152

Recommendation ITU-T X.1144

eXtensible Access Control Markup Language (XACML 3.0)

1 Scope

This Recommendation defines the eXtensible Access Control Markup Language (XACML) Version 3.0. 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, <i>Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services</i> . |
| [ITU-T X.509] | Recommendation ITU-T X.509 (2012) ISO/IEC 9594-8:2014, <i>Information technology – Open Systems Interconnection – The Directory: Public key and attribute certificate frameworks</i> . |

- [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)*.
- [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*.
- [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*.
- [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*.
<<http://www.ietf.org/rfc/rfc2798.txt>>
- [IETF RFC 2822] IETF RFC 2822 (2001), *Internet Message Format*.
<<http://www.ietf.org/rfc/rfc2822.txt>>
- [IETF RFC 3198] IETF RFC 3198 (2001), *Terminology for Policy-Based Management*.
<<http://www.ietf.org/rfc/rfc3198.txt>>
- [IETF RFC 3986] IETF RFC 3986 (2005), *Uniform Resource Identifiers (URI): Generic Syntax*.
<<http://www.ietf.org/rfc/rfc3986.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 4949] IETF RFC 4949 (2007), *Internet Security Glossary, Version 2*.
<<http://www.ietf.org/rfc/rfc4949.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*.
- [W3C DS] W3C Recommendation (10 June 2008), *XML-Signature Syntax and Processing*.
<<http://www.w3.org/TR/xmlsig-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 Glossary] W3C Glossary and Dictionary (2003), *Glossary and Dictionary*.
<<http://www.w3.org/2003/glossary/>>

- [W3C MathML] W3C Recommendation (21 October 2003), *Mathematical Markup Language (MathML), Version 2.0*.
<<http://www.w3.org/TR/2003/REC-MathML2-20031021/>>
- [W3C Technology] W3C XMLTechnology.
<<http://www.w3.org/standards/xml>>
- [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 XMLid] W3C Recommendation (9 September 2005), *xml:id Version 1.0*.
<<http://www.w3.org/TR/2005/REC-xml-id-20050909/>>
- [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 [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 [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 [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 [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 [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 [ITU-T X.812].

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

3.1.9 security architecture [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 [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 [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) [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 [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 [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 [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: The set of decision requests, identified by definitions for resource, subject and action that a rule, policy, or policy set is intended to evaluate.

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.

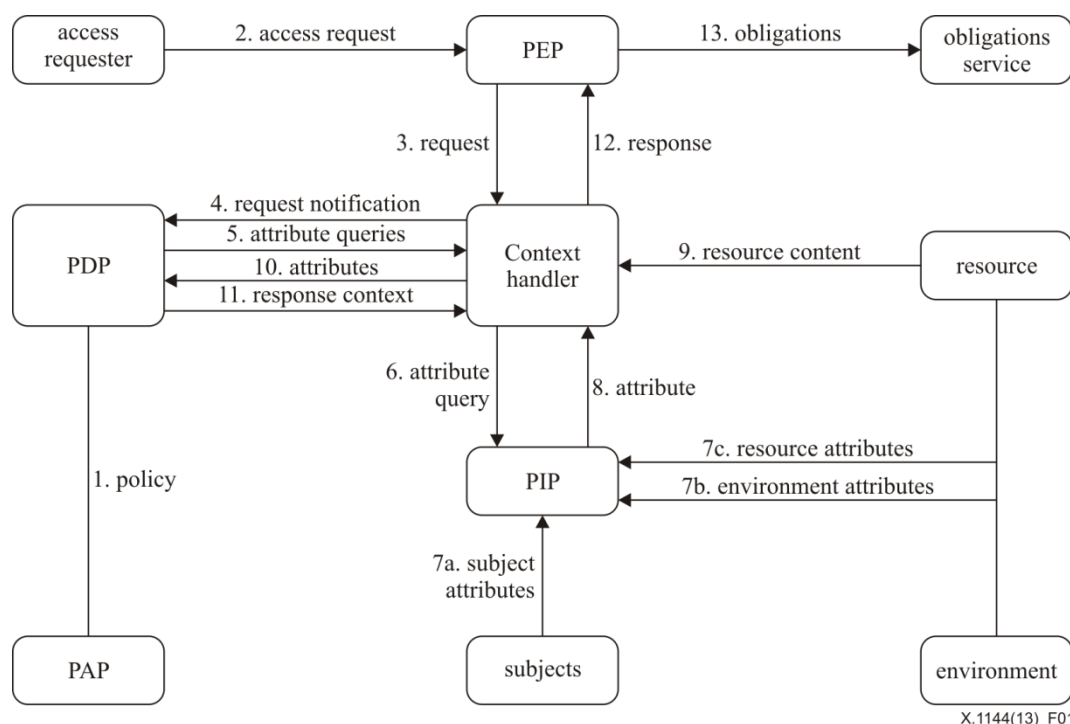


Figure 1 – Data-flow diagram

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).

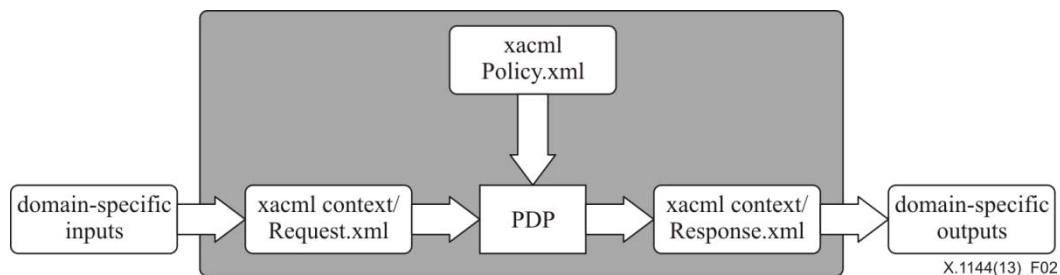


Figure 2 – XACML context

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.

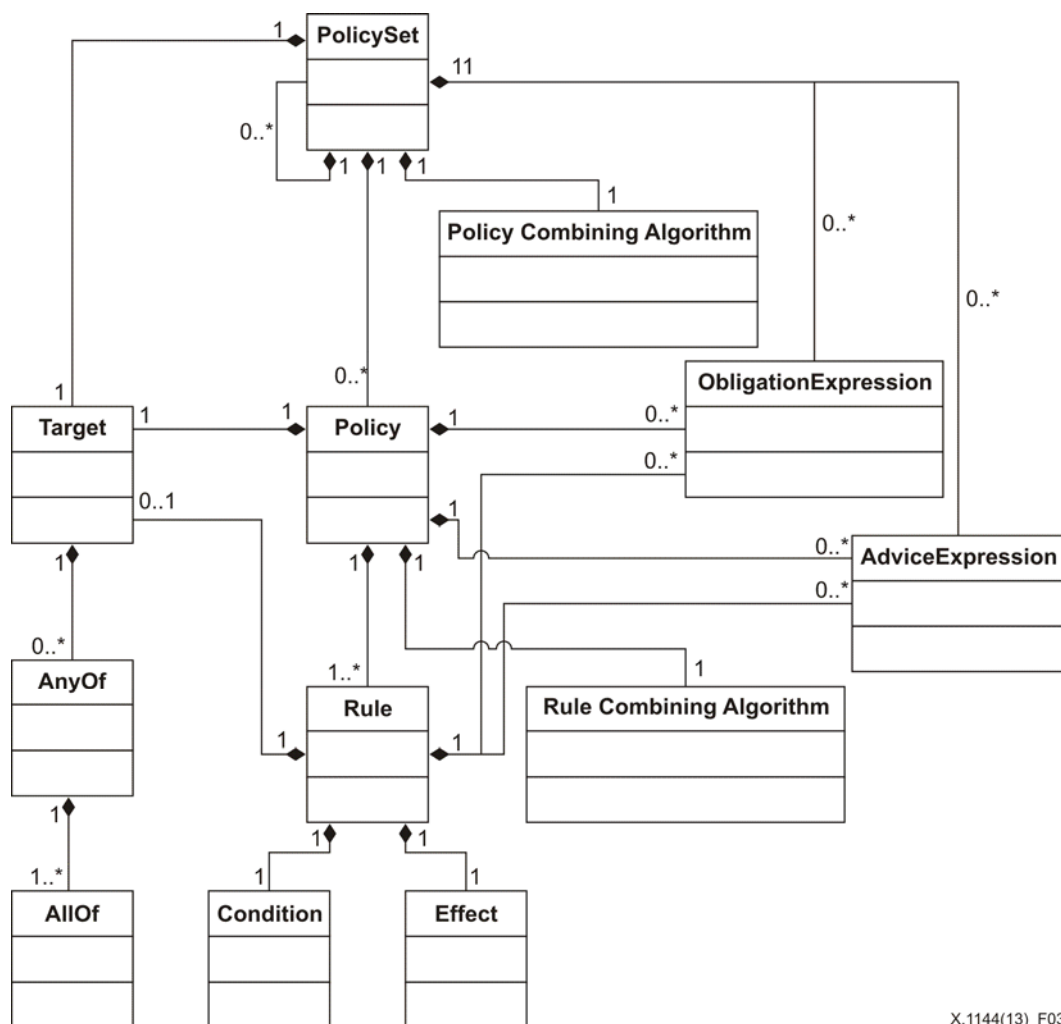
See clause 10.3.5 for a more detailed discussion of the request **context**.

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.



X.1144(13)_F03

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**.

<AdviceExpressions> [Optional]

A **conjunctive sequence** of **advice** expressions which must be 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.

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.

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 `<PolicyCombinerParameters>` 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 `<PolicySetCombinerParameters>` 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 be 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.

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 the XPath expression selects no node. See clause 10.3.5. See also clauses 10.19.2 and 10.19.3.

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:extension>
  </xs:complexContent>
</xs:complexType>
```

```

        </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 be 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 document 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.

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

NOTE 1 – 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).

```
<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 2 – 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.

`<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><xs:complexType name="SubjectType">

```

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>

```