

I n t e r n a t i o n a l T e l e c o m m u n i c a t i o n U n i o n

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.1142

(06/2006)

SERIES X: DATA NETWORKS, OPEN SYSTEM
COMMUNICATIONS AND SECURITY

Telecommunication security

eXtensible Access Control Markup Language (XACML 2.0)

ITU-T Recommendation X.1142

ITU-T X-SERIES RECOMMENDATIONS
DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY

PUBLIC DATA NETWORKS	
Services and facilities	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalling and switching	X.50–X.89
Network aspects	X.90–X.149
Maintenance	X.150–X.179
Administrative arrangements	X.180–X.199
OPEN SYSTEMS INTERCONNECTION	
Model and notation	X.200–X.209
Service definitions	X.210–X.219
Connection-mode protocol specifications	X.220–X.229
Connectionless-mode protocol specifications	X.230–X.239
PICS proformas	X.240–X.259
Protocol Identification	X.260–X.269
Security Protocols	X.270–X.279
Layer Managed Objects	X.280–X.289
Conformance testing	X.290–X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300–X.349
Satellite data transmission systems	X.350–X.369
IP-based networks	X.370–X.379
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600–X.629
Efficiency	X.630–X.639
Quality of service	X.640–X.649
Naming, Addressing and Registration	X.650–X.679
Abstract Syntax Notation One (ASN.1)	X.680–X.699
OSI MANAGEMENT	
Systems Management framework and architecture	X.700–X.709
Management Communication Service and Protocol	X.710–X.719
Structure of Management Information	X.720–X.729
Management functions and ODMA functions	X.730–X.799
SECURITY	X.800–X.849
OSI APPLICATIONS	
Commitment, Concurrency and Recovery	X.850–X.859
Transaction processing	X.860–X.879
Remote operations	X.880–X.889
Generic applications of ASN.1	X.890–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999
TELECOMMUNICATION SECURITY	X.1000–

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

eXtensible Access Control Markup Language (XACML 2.0)

Summary

XACML is an XML vocabulary for expressing access control policies. Access control consists of deciding if a requested resource access should be allowed and enforcing that decision. This Recommendation defines core XACML including syntax of the language, models, context with policy language model, syntax and processing rules. This Recommendation specifies XACML core and hierarchical role based access control profile. A multiple resource profile of XACML and a SAML 2.0 profile of XACML are specified. To improve on the security of exchanging XACML based policies, this Recommendation also specifies an XACML XML digital signature profile for securing data. A privacy profile is specified in order to provide guidelines for implementers.

This Recommendation is technically equivalent and compatible with the OASIS XACML 2.0 standard.

Source

ITU-T Recommendation X.1142 was approved on 13 June 2006 by ITU-T Study Group 17 (2005-2008) under the ITU-T Recommendation A.8 procedure.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. 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 2007

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

CONTENTS

	<i>Page</i>
1 Scope	1
2 References	1
3 Definitions	2
3.1 Imported definitions	2
3.2 Additional definitions	3
4 Abbreviations	5
5 Conventions	5
6 Overview	6
7 XACML core.....	6
7.1 Background	6
7.2 XACML models	10
7.3 XACML context	11
7.4 Policy syntax.....	14
7.5 Context syntax.....	34
7.6 XACML functional requirements	41
7.7 XACML extensibility points	49
7.8 Conformance.....	50
8 Core and hierarchical role based access control (RBAC) profile.....	57
8.1 RBAC background	57
8.2 RBAC example.....	59
8.3 Assigning and enabling role attributes	63
8.4 Implementing the RBAC model	65
8.5 Profile.....	67
8.6 Identifiers	67
9 Multiple resource profile of XACML	68
9.1 Requests for multiple resources.....	69
9.2 Requests for an entire hierarchy	71
9.3 New attribute identifiers	72
9.4 New profile identifiers	73
10 SAML 2.0 profile of XACML.....	73
10.1 Mapping SAML and XACML attributes.....	75
10.2 Authorization decisions	76
10.3 Policies	77
10.4 Element <saml:Assertion>	79
10.5 Element <samlp:RequestAbstractType>	80
10.6 Element <samlp:Response>	80
11 XML digital signature profile.....	81
11.1 Use of SAML	81
11.2 Canonicalization	81
11.3 Signing schemas	82
12 Hierarchical resource profile of XACML	82
12.1 Representing the identity of a node	83
12.2 Requesting access to a node	84
12.3 Stating policies that apply to nodes	87
12.4 New DataType: xpath-expression	87
12.5 New attribute identifiers	88
12.6 New profile identifiers	88
13 Privacy policy profile	89
13.1 Standard attributes.....	89
13.2 Standard rules: Matching purpose.....	89

	<i>Page</i>
Annex A – Data-types and functions	90
A.1 Introduction	90
A.2 Data-types	90
A.3 Functions.....	92
Annex B – XACML identifiers	104
B.1 XACML namespaces	104
B.2 Access subject categories.....	104
B.3 Data-types	104
B.4 Subject attributes	105
B.5 Resource attributes	106
B.6 Action attributes	106
B.7 Environment attributes	106
B.8 Status codes	107
B.9 Combining algorithms.....	107
Annex C – Combining algorithms	108
C.1 Deny-overrides	108
C.2 Ordered-deny-overrides.....	109
C.3 Permit-overrides	109
C.4 Ordered-permit-overrides	111
C.5 First-applicable.....	111
C.6 Only-one-applicable	113
Annex D – XACML schema	114
D.1 XACML context schema	114
D.2 Policy schema	116
D.3 XACML SAML protocol schema	122
D.4 XACML SAML assertion schema	123
Appendix I – Security considerations	124
I.1 Threat model	124
I.2 Safeguards	126
Appendix II – XACML examples.....	128
II.1 Example one	128
II.2 Example two	131
Appendix III – Example description of higher order bag functions.....	145
III.1 Example of higher-order bag functions.....	145
BIBLIOGRAPHY	150

eXtensible Access Control Markup Language (XACML 2.0)

1 Scope

This Recommendation defines the eXtensible Access Control Markup Language (XACML) Version 2.0. 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 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 XACML solutions for each of these requirements are in this Recommendation. In particular, clause 7 develops the core XACML language including XACML models, policy language model, policy syntax and processing rules. Clause 8 develops XACML core and hierarchical Role Based Access Control (RBAC) profile. Clause 9 develops XACML multiple resource profile. Clause 10 discusses technologies for securing XACML communication through the development of SAML 2.0 profile of XACML. Clause 11 builds on the foundation of clause 10 by developing an XML digital signature profile for XACML. Clause 12 discusses the combination of XACML profiles as developed in clauses 7 to 11 by developing a hierarchical resource profile of XACML. Privacy issues are discussed in clause 13.

2 References

The following 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, and parties to agreements based on this Recommendation are encouraged to investigate the possibility of applying the most recent editions of the Recommendations and other references listed below. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations. The IETF maintains a list of RFCs, together with those that have been made obsolete by later RFCs. W3C maintains a list of the latest Recommendations and other publications.

- ITU-T Recommendation X.811 (1995) | ISO/IEC 10181-2:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Authentication framework*.
- ITU-T Recommendation X.812 (1995) | ISO/IEC 10181-3:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework*.
- ITU-T Recommendation X.1141 (2006), *Security Assertion Markup Language (SAML 2.0)*.
- IETF RFC 822 (1982), *Standard for the Format of ARPA Internet Text Messages*.
- IETF RFC 2119 (1997), *Key words for use in RFCs to Indicate Requirement Levels*.
- IETF RFC 2253 (1997), *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*.
- IETF RFC 2256 (1997), *A Summary of the X.500 (96) User Schema for use with LDAPv3*.

- IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax*.
- IETF RFC 2732 (1999), *Format for Literal IPv6 Addresses in URL's*.
- IETF RFC 2821 (2001), *Simple Mail Transfer Protocol*.
- IETF RFC 3280 (2002), *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.
- W3C Canonicalization:2002, *Exclusive XML Canonicalization Version 1.0*, W3C Recommendation, Copyright © [18 July 2002] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/xml-exc-c14n/>.
- W3C Datatypes:2001, *XML Schema Part 2: Datatypes*, W3C Recommendation, Copyright © [2 May 2001] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>.
- W3C Encryption:2002, *XML Encryption Syntax and Processing*, W3C Recommendation, Copyright © [10 December 2002] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>.
- W3C MathML:2003, *Mathematical Markup Language (MathML), Version 2.0*, W3C Recommendation, Copyright © [21 October 2003] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2003/REC-MathML2-20031021/>.
- W3C Signature:2002, *XML-Signature Syntax and Processing*, W3C Recommendation, Copyright © [12 February 2002] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/>.
- W3C XML:2004, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation, Copyright © [4 February 2004] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/REC-xml/>.
- W3C XPATH:1999, *XML Path Language, Version 1.0*, W3C Recommendation, Copyright © [16 November 1999] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/1999/REC-xpath-19991116/>.
- W3C XSLT:1999, *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation, Copyright © [16 November 1999] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/1999/REC-xslt-19991116/>.

NOTE – The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

3 Definitions

For the purposes of this Recommendation, the following definitions apply.

3.1 Imported definitions

3.1.1 This Recommendation uses the following terms defined in ITU-T Rec. X.811:

a) Principle.

3.1.2 This Recommendation uses the following terms defined in ITU-T Rec. X.812:

- a) Access control information;
- b) User.

3.1.3 This Recommendation uses the following terms defined in W3C Web Services Glossary:

- a) Namespace;
- b) XML Schema.

- 3.1.4** This Recommendation uses the following terms defined in IETF RFC 2828:
- a) Access;
 - b) Access control;
 - c) Policy administration point;
 - d) Policy decision point;
 - e) Policy enforcement point;
 - f) Security architecture;
 - g) Security policy;
 - h) Security service.
- 3.1.5** This Recommendation uses the following terms defined in IETF RFC 2396:
- a) Uniform resource identifier (URI);
 - b) URI reference.
- 3.1.6** This Recommendation uses the following terms defined in W3C XML Signature:
- a) Data object.

3.2 Additional definitions

- 3.2.1 access:** Performing an action.
- 3.2.2 access control:** Controlling access in accordance with a policy.
- 3.2.3 action:** An operation on a resource.
- 3.2.4 applicable policy:** The set of policies and policy sets that govern access for a specific decision request.
- 3.2.5 attribute:** Characteristic of a subject, resource, action or environment that may be referenced in a predicate or target.
- 3.2.6 attribute authority (AA):** An entity that binds attributes to identities. Such a binding may be expressed using a SAML attribute assertion with the attribute Authority as the issuer.
- 3.2.7 authorization decision:** The result of evaluating applicable policy, returned by the PDP to the PEP. A function that evaluates to "Permit", "Deny", "Indeterminate" or "NotApplicable", and (optionally) a set of obligations.
- 3.2.8 bag:** An unordered collection of values, in which there may be duplicate values.
- 3.2.9 condition:** An expression of predicates. A function that evaluates to "True", "False" or "Indeterminate".
- 3.2.10 conjunctive sequence:** A sequence of predicates combined using the logical 'AND' operation.
- 3.2.11 context:** The canonical representation of a decision request and an authorization decision.
- 3.2.12 context handler:** The system entity that converts decision requests in the native request format to the XACML canonical form and converts authorization decisions in the XACML canonical form to the native response format.
- 3.2.13 custodian:** The entity to which personally-identifiable information is entrusted.
- 3.2.14 data object:** Refer to a digital object that is being signed. A data object is referenced inside an <Reference> element using a URI.
- 3.2.15 decision:** The result of evaluating a rule, policy or policy set.
- 3.2.16 decision request:** The request by a PEP to a PDP to render an authorization decision.
- 3.2.17 disjunctive sequence:** A sequence of predicates combined using the logical 'OR' operation.
- 3.2.18 effect:** The intended consequence of a satisfied rule (either "Permit" or "Deny").
- 3.2.19 environment:** The set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource or action.
- 3.2.20 HasPrivilegesOfRole policy:** An optional type of <Policy> that can be included in a permission <PolicySet> to allow support queries asking if a subject "has the privileges of" a specific role.

- 3.2.21 hierarchical resource:** A resource that is organized as a tree or forest (directed acyclic graph) of individual resources called nodes.
- 3.2.22 junior role:** In a role hierarchy, role A is *junior* to role B if role B inherits all the permissions associated with role A.
- 3.2.23 multi-role permissions:** A set of permissions for which a user must hold more than one role simultaneously in order to gain access.
- 3.2.24 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.
- 3.2.25 node:** An individual resource that is part of a hierarchical resource.
- 3.2.26 obligation:** An operation specified in a policy or policy set that should be performed by the PEP in conjunction with the enforcement of an authorization decision.
- 3.2.27 owner:** The subject of personally-identifiable information.
- 3.2.28 permission:** The ability or right to perform some action on some resource, possibly only under certain specified conditions.
- 3.2.29 permission <policy set> (PPS):** A <PolicySet> that contains the actual permissions associated with a given role.
- 3.2.30 policy information point (PIP):** The system entity that acts as a source of attribute values.
- 3.2.31 policy set:** A set of policies, other policy sets, a policy-combining algorithm and (optionally) a set of obligations. May be a component of another policy set.
- 3.2.32 predicate:** A statement about attributes whose truth can be evaluated.
- 3.2.33 resource:** Data, service or system component.
- 3.2.34 role:** A job function within the context of an organization that has associated semantics regarding the authority and responsibility conferred on the user assigned to the role.
- 3.2.35 role based access control (RBAC):** A model for controlling access to resources where permitted actions on resources are identified with roles rather than with individual subject identities.
- 3.2.36 role enablement authority:** An entity that assigns role attributes and values to users or enables role attributes and values during a user's session.
- 3.2.37 role <PolicySet> (RPS):** A <PolicySet> that associates holders of a given role attribute and value with a Permission <PolicySet> that contains the actual permissions associated with the given role.
- 3.2.38 senior role:** In a role hierarchy, role A is *senior* to role B if role A inherits all the permissions associated with role B.
- 3.2.39 rule:** A target, an effect and a condition. A component of a policy.
- 3.2.40 rule-combining algorithm:** The procedure for combining decisions from multiple rules.
- 3.2.41 subject:** An actor whose attributes may be referenced by a predicate.
- 3.2.42 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.
- 3.2.43 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**".

4 Abbreviations

For the purposes of this Recommendation, the following abbreviations apply:

AA	Attribute Authority
ASP	Application Service Provider
CA	Certification Authority
CMP	Certificate Management Protocol
CRL	Certificate Revocation List
ECP	Enhanced Client/Proxy
HTTP	Hypertext Transfer Protocol
ID	IDentifier
IPSEC	IP SECurity protocol
LDAP	Lightweight Directory Access Protocol
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PKI	Public-Key Infrastructure
POP	Proof of Possession
PPS	Permission <Policy Set>
RA	Registration Authority
RBAC	Role Based Access Control
RPS	Role <PolicySet>
RSA	Rivest, Shamir, Adleman (public key algorithm)
SAML	Security Assertion Markup Language
SP	Service Provider
SSO	Single Sign On
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URN	Uniform Resource Name
XML	eXtensible Markup Language
XPath	XML Path Language
XSLT	eXtensible Stylesheet Language

5 Conventions

This Recommendation uses the keywords "must", "must not", "required", "shall", "shall not", "should", "should not", "recommended", "may", and "optional". In this Recommendation, these terms are to be interpreted as described in RFC 2119.

In this Recommendation, the phrase "Normative, but optional" means that the described functionality is optional for compliant XACML implementations, but, if the functionality is claimed as being supported according to this profile, then it shall be supported in the way described.

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 very 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. And the policy may be enforced by the extranet, mail, 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. And, 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 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, analyzing, modifying, withdrawing, retrieving and enforcing policy.

7 XACML core

This clause develops the foundation of XACML including general policy requirements, models, general context, policy syntax and provides some examples.

7.1 Background

This clause is informative.

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 an application, and the widespread support that it enjoys from all the main platform and tool vendors.

7.1.1 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 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.

7.1.2 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 PDP. So, 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 PAP, where it may form the basic unit of management, and be reused in multiple policies.

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 the 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.

7.1.3 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. Standard combining algorithms are defined for:

- 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 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.

7.1.4 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. An attribute called "subject-category" is used to differentiate between subjects acting in different capacities. Some standard values for this attribute are specified, and users may define additional ones.

7.1.5 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. XACML provides facilities to support this approach. Attributes of subjects contained in the request context may be identified by the `<SubjectAttributeDesignator>` element. This element contains a URN that identifies the attribute. Alternatively, the `<AttributeSelector>` element may contain an XPath expression over the request context to identify a particular subject attribute value by its location in the context.

XACML provides a standard way to reference the attributes defined in IETF RFC 2253. 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 `<ResourceAttributeDesignator>` element. This element contains a URN that identifies the attribute. Alternatively, the `<AttributeSelector>` element may contain an XPath expression over the request context to identify a particular resource attribute value by its location in the context.

7.1.6 Multi-valued attributes

The most common techniques for communicating attributes (LDAP, XPath, 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 the PDP should handle the case of multiple attribute values. These are the "higher-order" functions.

7.1.7 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 request context to identify data in the information resource to be used in the policy evaluation.

7.1.8 Operator

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. And, 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 X.500 name-forms, strings, 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.

7.1.9 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. 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.

7.1.10 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, the 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, the PDP should evaluate the <Target> element of the retrieved policy or policy set statements.
- 2) Alternatively, the 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.

7.1.11 Abstraction layer

PEPs come in many forms. For instance, a PEP may be part of a remote-access gateway, part of a Web server or part of an email user-agent. It is unrealistic to expect that all PEPs in an enterprise do currently, or will in the future, issue decision requests to a 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., 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 the PEP and the XACML context format understood by the PDP.

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

In the case where the native request/response format is specified in XML Schema (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.

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 in the policy, values in the resource may be included in the policy evaluation.

7.1.12 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. XACML provides facilities to specify actions that must be performed in conjunction with policy evaluation in the <Obligations> element. There are no standard definitions for these actions in version 2.0 of XACML. Therefore, bilateral agreement between a PAP and the PEP that will enforce its policies is required for correct interpretation. PEPs that conform with v2.0 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 the PEP for enforcement.

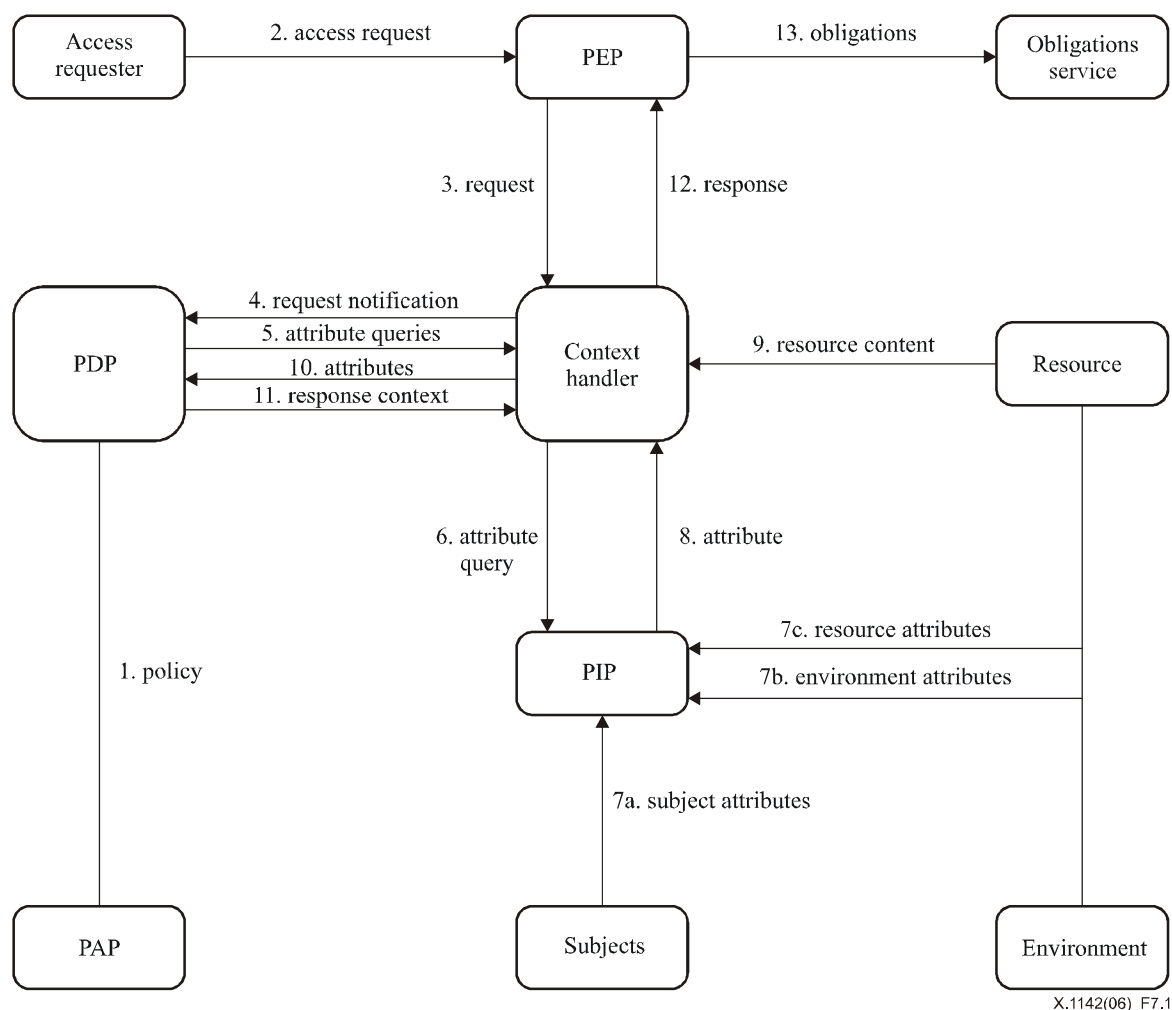
7.2 XACML models

This clause is informative.

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

7.2.1 Data-flow model

The major actors in the XACML domain are shown in the data-flow diagram of Figure 7-1.



X.1142(06)_F7.1

Figure 7-1/X.1142 – 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 by the following steps.

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

- 3) The PEP sends the request for access to the context handler in its native request format, optionally including attributes of the subjects, resource, action and environment.
- 4) The context handler constructs an XACML Request context and sends it to the PDP.
- 5) The PDP requests any additional subject, resource, action and environment attributes from the context handler.
- 6) The context handler requests the attributes from a PIP.
- 7) The PIP obtains the requested attributes.
- 8) The 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 the PDP. The PDP evaluates the policy.
- 11) The 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 the PEP. The context handler returns the response to the PEP.
- 13) The PEP fulfils the obligations.
- 14) (Not shown) If access is permitted, then the PEP permits access to the resource; otherwise, it denies access.

7.3 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 7-2, in which the scope of XACML is indicated by the shaded area. The XACML context is defined in XML schema, describing a canonical representation for the inputs and outputs of the PDP. Attributes referenced by an instance of XACML policy may be in the form of XPath expressions over the context, or attribute designators that identify the attribute by subject, resource, action or environment and its identifier, data-type and (optionally) its issuer. Implementations must convert between the attribute representations in the application environment (e.g., SAML) 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 XSLT transformation (see W3C XSLT:1999).

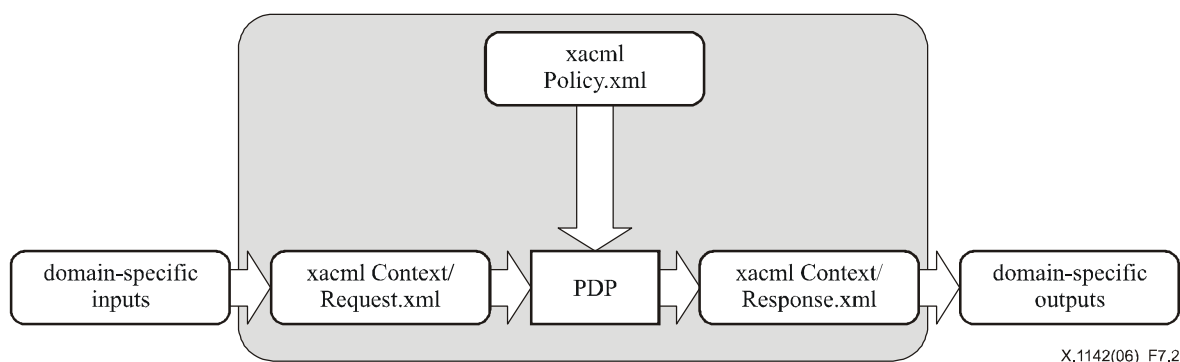


Figure 7-2/X.1142 – XACML context

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

7.3.1 Policy language model

The policy language model is shown in Figure 7-3. The main components of the model are:

- Rule;
- Policy; and
- Policy set.

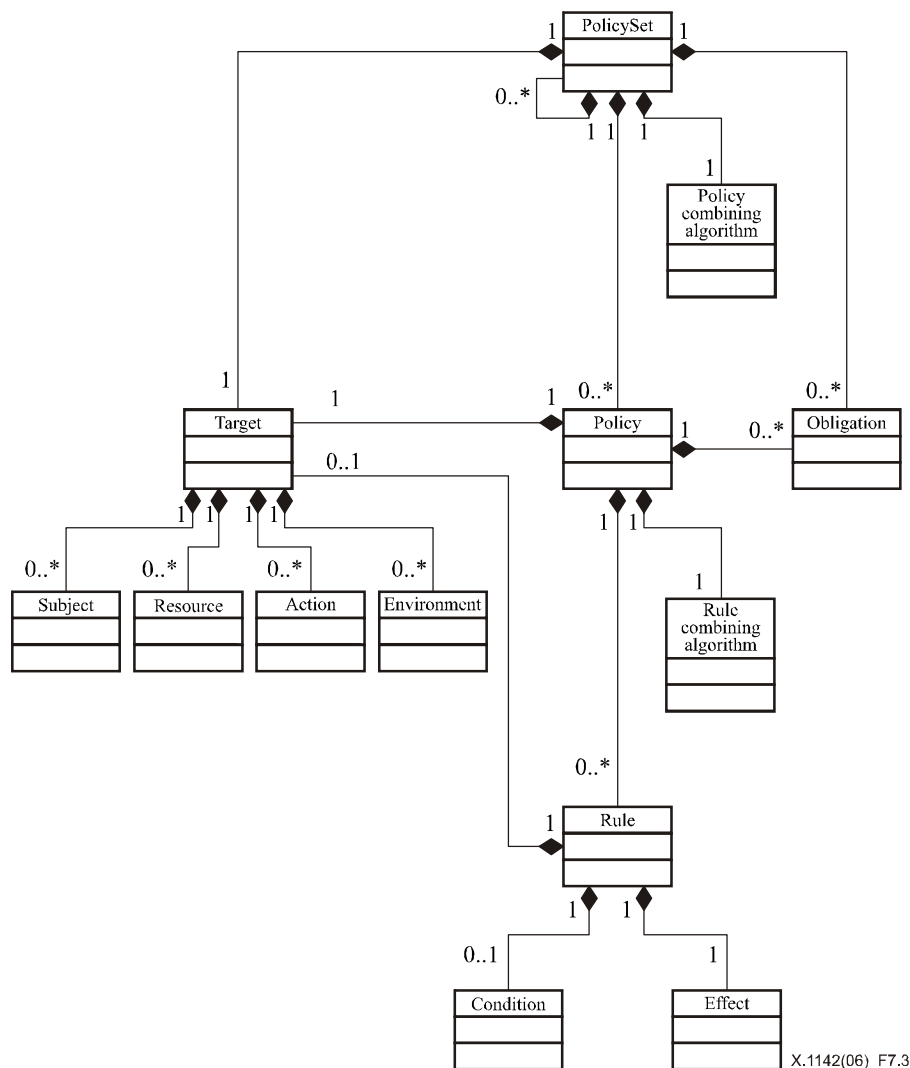


Figure 7-3/X.1142 – Policy language model

7.3.1.1 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; and
- a condition.

7.3.1.1.1 Rule target

The target defines the set of:

- resources;
- subjects;
- actions; and
- environment,

to which the rule is intended to apply. 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 subjects, resource, action and environment attributes in the request context. Target definitions are discrete, in order that applicable rules may be efficiently identified by the PDP.

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 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. And 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 mail addresses hosted by the med.example.com mail server. And the XPath/XPointer value `//xacml-context:Request/xacml-context:Resource/xacml-context:ResourceContent/md:record/md:patient/` is a legal XPath/XPointer 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 the 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. So, 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"`.

7.3.1.1.2 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".

7.3.1.1.3 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.

7.3.1.2 Policy

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

- a target;
- a rule-combining algorithm-identifier;
- a set of rules; and
- obligations.

7.3.1.2.1 Policy target

An XACML <PolicySet>, <Policy> or <Rule> element contains a <Target> element that specifies the set of subjects, resources, actions and environments 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 likely 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.

7.3.1.2.2 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 the 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.

7.3.1.2.3 Obligations

Obligations may be added by the writer of the policy.

When a PDP evaluates a policy containing obligations, it returns certain of those obligations to the PEP in the response context.

7.3.1.3 Policy set

A policy set comprises four main components:

- a target;
- a policy-combining algorithm-identifier;
- a set of policies; and
- obligations.

7.3.1.3.1 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 the 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.

7.3.1.3.2 Obligations

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

When a PDP evaluates a policy set containing obligations, it returns certain of those obligations to the PEP in its response context.

7.4 Policy syntax

Schema fragments in the following clauses are non-normative.

7.4.1 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 this Recommendation shall be used.

If a `<PolicySet>` element contains references to other policy sets or policies in the form of URLs, then these references may be resolvable.

Policy sets and policies included in a `<PolicySet>` element must be combined using the algorithm identified by the `PolicyCombiningAlgId` attribute. `<PolicySet>` is treated exactly like a `<Policy>` in all policy combining algorithms.

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

The `<Obligations>` element contains a set of obligations that must be fulfilled by the PEP in conjunction with the authorization decision. If the PEP does not understand, or cannot fulfil, any of the obligations, then it must act as if the PDP had returned a "Deny" authorization decision value.

```

<xs:element name="PolicySet" type="xacml:PolicySetType"/>
<xs:complexType name="PolicySetType">
  <xs:sequence>
    <xs:element ref="xacml:Description" 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:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
  <xs:attribute name="Version" type="xacml:VersionType" default="1.0"/>
  <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
</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 the PAP to ensure that no two policies visible to the 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 a URL, then it may be resolvable.
- Version [Default 1.0]
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.
- <Description> [Optional]
A free-form description 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 a 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 a URL, then it may be resolvable.
- `<Obligations>` [Optional]
Contains the set of `<Obligation>` elements.
- `<CombinerParameters>` [Optional]
Contains a sequence of `<CombinerParameter>` elements.
- `<PolicyCombinerParameters>` [Optional]
Contains a sequence of `<CombinerParameter>` elements that are associated with a particular `<Policy>` or `<PolicyIdReference>` element within the `<PolicySet>`.
- `<PolicySetCombinerParameters>` [Optional]
Contains a sequence of `<CombinerParameter>` elements that are associated with a particular `<PolicySet>` or `<PolicySetIdReference>` element within the `<PolicySet>`.

7.4.2 Element `<Description>`

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

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

7.4.3 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" minOccurs="0"/>
    </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.

7.4.4 Element `<XPathVersion>`

The `<XPathVersion>` element shall specify the version of W3C XPath:1999 to be used by `<AttributeSelector>` elements and XPath-based functions in the policy set or policy.

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

The URI for W3C XPath:1999 is "http://www.w3.org/TR/1999/Rec-xpath-19991116". The `<XPathVersion>` element is required if the XACML enclosing policy set or policy contains `<AttributeSelector>` elements or XPath-based functions.

7.4.5 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. It contains definitions for subjects, resources, actions and environments.

The <Target> element shall contain a conjunctive sequence of <Subjects>, <Resources> <Actions> and <Environments> 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 section of the <Target> element and the corresponding section of the <xacml-context:Request> element.

```
<xs:element name="Target" type="xacml:TargetType"/>
<xs:complexType name="TargetType">
  <xs:sequence>
    <xs:element ref="xacml:Subjects" minOccurs="0"/>
    <xs:element ref="xacml:Resources" minOccurs="0"/>
    <xs:element ref="xacml:Actions" minOccurs="0"/>
    <xs:element ref="xacml:Environments" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

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

The <Target> element contains the following elements:

- <Subjects> [Optional]
Matching specification for the subject attributes in the context. If this element is missing, then the target shall match all subjects.
- <Resources> [Optional]
Matching specification for the resource attributes in the context. If this element is missing, then the target shall match all resources.
- <Actions> [Optional]
Matching specification for the action attributes in the context. If this element is missing, then the target shall match all actions.
- <Environments> [Optional]
Matching specification for the environment attributes in the context. If this element is missing, then the target shall match all environments.

7.4.6 Element <Subjects>

The <Subjects> element shall contain a disjunctive sequence of <Subject> elements.

```
<xs:element name="Subjects" type="xacml:SubjectsType"/>
<xs:complexType name="SubjectsType">
  <xs:sequence>
    <xs:element ref="xacml:Subject" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <Subjects> element is of **SubjectsType** complex type.

The <Subjects> element contains the following elements:

- <Subject> [One to Many, Required]
As defined in 7.4.7.

7.4.7 Element <Subject>

The <Subject> element shall contain a conjunctive sequence of <SubjectMatch> elements.

```
<xs:element name="Subject" type="xacml:SubjectType"/>
<xs:complexType name="SubjectType">
  <xs:sequence>
    <xs:element ref="xacml:SubjectMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <Subject> element is of **SubjectType** complex type.

The <Subject> element contains the following elements:

- `<SubjectMatch>` [One to Many]
A conjunctive sequence of individual matches of the subject attributes in the request context and the embedded attribute values.

7.4.8 Element `<SubjectMatch>`

The `<SubjectMatch>` element shall identify a set of subject-related entities by matching attribute values in a `<xacml-context:Subject>` element of the request context with the embedded attribute value.

```
<xs:element name="SubjectMatch" type="xacml:SubjectMatchType"/>
<xs:complexType name="SubjectMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:SubjectAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

The `<SubjectMatch>` element is of **SubjectMatchType** complex type.

The `<SubjectMatch>` 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 as documented in 7.6.5.
- `<xacml:AttributeValue>` [Required]
Embedded attribute value.
- `<SubjectAttributeDesignator>` [Required choice]
may be used to identify one or more attribute values in a `<Subject>` element of the request context.
- `<AttributeSelector>` [Required choice]
may be used to identify one or more attribute values in the request context. The XPath expression should resolve to an attribute in a `<Subject>` element of the request context.

7.4.9 Element `<Resources>`

The `<Resources>` element shall contain a disjunctive sequence of `<Resource>` elements.

```
<xs:element name="Resources" type="xacml:ResourcesType"/>
<xs:complexType name="ResourcesType">
  <xs:sequence>
    <xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The `<Resources>` element is of **ResourcesType** complex type.

The `<Resources>` element contains the following elements:

- `<Resource>` [One to Many, Required]
See 7.4.10.

7.4.10 Element `<Resource>`

The `<Resource>` element shall contain a conjunctive sequence of `<ResourceMatch>` elements.

```
<xs:element name="Resource" type="xacml:ResourceType"/>
<xs:complexType name="ResourceType">
  <xs:sequence>
    <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <Resource> element is of **ResourceType** complex type.

The <Resource> element contains the following elements:

- <ResourceMatch> [One to Many]
A conjunctive sequence of individual matches of the resource attributes in the request context and the embedded attribute values.

7.4.11 Element <ResourceMatch>

The <ResourceMatch> element shall identify a set of resource-related entities by matching attribute values in the <xacml-context:Resource> element of the request context with the embedded attribute value.

```
<xs:element name="ResourceMatch" type="xacml:ResourceMatchType"/>
<xs:complexType name="ResourceMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:ResourceAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

The <ResourceMatch> element is of **ResourceMatchType** complex type.

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

- MatchId [Required]
Specifies a matching function. Values of this attribute must be of type **xs:anyURI**, with legal values as documented in 7.6.5.
- <xacml:AttributeValue> [Required]
Embedded attribute value.
- <ResourceAttributeDesignator> [Required Choice]
may be used to identify one or more attribute values in the <Resource> element of the request context.
- <AttributeSelector> [Required Choice]
may be used to identify one or more attribute values in the request context. The XPath expression should resolve to an attribute in the <Resource> element of the request context.

7.4.12 Element <Actions>

The <Actions> element shall contain a disjunctive sequence of <Action> elements.

```
<xs:element name="Actions" type="xacml:ActionsType"/>
<xs:complexType name="ActionsType">
  <xs:sequence>
    <xs:element ref="xacml:Action" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <Actions> element is of **ActionsType** complex type.

The <Actions> element contains the following elements:

- <Action> [One to Many, Required]
See 7.4.13.

7.4.13 Element <Action>

The <Action> element shall contain a conjunctive sequence of <ActionMatch> elements.

```
<xs:element name="Action" type="xacml:ActionType"/>
<xs:complexType name="ActionType">
```

```

<xs:sequence>
  <xs:element ref="xacml:ActionMatch" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

```

The <Action> element is of **ActionType** complex type.

The <Action> element contains the following elements:

- <ActionMatch> [One to Many]
A conjunctive sequence of individual matches of the action attributes in the request context and the embedded attribute values, see 7.4.14.

7.4.14 Element <ActionMatch>

The <ActionMatch> element shall identify a set of action-related entities by matching attribute values in the <xacml-context:Action> element of the request context with the embedded attribute value.

```

<xs:element name="ActionMatch" type="xacml:ActionMatchType"/>
<xs:complexType name="ActionMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:ActionAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>

```

The <ActionMatch> element is of **ActionMatchType** complex type.

The <ActionMatch> 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 as documented in 7.6.5.
- <xacml:AttributeValue> [Required]
Embedded attribute value.
- <ActionAttributeDesignator> [Required Choice]
may be used to identify one or more attribute values in the <Action> element of the request context.
- <AttributeSelector> [Required Choice]
may be used to identify one or more attribute values in the request context. The XPath expression should resolve to an attribute in the <Action> element of the context.

7.4.15 Element <Environments>

The <Environments> element shall contain a disjunctive sequence of <Environment> elements.

```

<xs:element name="Environments" type="xacml:EnvironmentsType"/>
<xs:complexType name="EnvironmentsType">
  <xs:sequence>
    <xs:element ref="xacml:Environment" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

```

The <Environments> element is of **EnvironmentsType** complex type.

The <Environments> element contains the following elements:

- <Environment> [One to Many, Required]
See 7.4.16.

7.4.16 Element <Environment>

The <Environment> element shall contain a conjunctive sequence of <EnvironmentMatch> elements.

```
<xs:element name="Environment" type="xacml:EnvironmentType"/>
<xs:complexType name="EnvironmentType">
  <xs:sequence>
    <xs:element ref="xacml:EnvironmentMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <Environment> element is of **EnvironmentType** complex type.

The <Environment> element contains the following elements:

- <EnvironmentMatch> [One to Many]
A conjunctive sequence of individual matches of the environment attributes in the request context and the embedded attribute values.

7.4.17 Element <EnvironmentMatch>

The <EnvironmentMatch> element shall identify an environment by matching attribute values in the <xacml-context:Environment> element of the request context with the embedded attribute value.

```
<xs:element name="EnvironmentMatch" type="xacml:EnvironmentMatchType"/>
<xs:complexType name="EnvironmentMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:EnvironmentAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

The <EnvironmentMatch> element is of **EnvironmentMatchType** complex type.

The <EnvironmentMatch> 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 as documented in 7.6.5.
- <xacml:AttributeValue> [Required]
Embedded attribute value.
- <EnvironmentAttributeDesignator> [Required Choice]
may be used to identify one or more attribute values in the <Environment> element of the request context.
- <AttributeSelector> [Required Choice]
may be used to identify one or more attribute values in the request context. The XPath expression should resolve to an attribute in the <Environment> element of the request context.

7.4.18 Element <PolicySetIdReference>

The <PolicySetIdReference> element shall be used to reference a <PolicySet> element by id. If <PolicySetIdReference> is a 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:extension>
  </xs:simpleContent>
</xs:complexType>
```

```

        <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.

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.

7.4.19 Element <PolicyIdReference>

The <xacml:PolicyIdReference> element shall be used to reference a <Policy> element by id. If <PolicyIdReference> is a 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.

7.4.20 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.

7.4.21 Simple type VersionMatchType

Elements of this type shall contain a restricted regular expression matching a version number. The expression shall match versions of a referenced policy or policy set that are 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.+'.3'.

7.4.22 Element <Policy>

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

A <Policy> element may be evaluated, in which case the evaluation procedure defined in 7.6.10 shall be used.

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

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 the PDP in making its authorization decision.

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 <Obligations> element contains a set of obligations that must be fulfilled by the PEP in conjunction with the authorization decision.

```
<xs:element name="Policy" type="xacml:PolicyType"/>
<xs:complexType name="PolicyType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
    <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
    <xs:element ref="xacml:CombinerParameters" 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:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
  <xs:attribute name="Version" type="xacml:VersionType" default="1.0"/>
  <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
</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 the PAP to ensure that no two policies visible to the 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 a URL, then it may be resolvable.
- Version [Default 1.0]
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.
- <Description> [Optional]
A free-form description 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.
- <RuleCombinerParameters> [Optional]
A sequence of parameters to be used by the rule-combining algorithm.

- **<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 match the decision request must be considered. Rules whose **<Target>** elements do not match the decision request shall be ignored.
- **<Obligations> [Optional]**
A conjunctive sequence of obligations that must be fulfilled by the PEP in conjunction with the authorization decision.

7.4.23 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" minOccurs="0"/>
    </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.

7.4.24 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 XACML 2.0 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.

Support for the **<CombinerParameters>** element is optional.

7.4.25 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 attribute:

- ParameterName [Required]
The identifier of the parameter.
- AttributeValue [Required]
The value of the parameter.

Support for the <CombinerParameter> element is optional.

7.4.26 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 XACML 2.0 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 elements:

- 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.

7.4.27 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 XACML 2.0 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 elements:

- 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.

7.4.28 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 XACML 2.0 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 elements:

- 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.

7.4.29 Element <Rule>

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

A <Rule> element may be evaluated, in which case the evaluation procedure defined in 7.6.9 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: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 7.6.6 for details.
- <Condition> [Optional]
A predicate that must be satisfied for the rule to be assigned its Effect value.

7.4.30 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 <Obligation> element.

```

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

```

7.4.31 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> element, 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.

7.4.32 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 string equality on the value of their respective VariableId attributes. There shall exist one and only one <VariableDefinition> 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 attributes:

- VariableId [Required]
The name used to refer to the value defined in a <VariableDefinition> element.

7.4.33 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>, <ActionAttributeDesignator>, <ResourceAttributeDesignator>, <SubjectAttributeDesignator> and <EnvironmentAttributeDesignator>.

7.4.34 Element <Condition>

The <Condition> element is a boolean function over subject, resource, action and environment 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".

7.4.35 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.

```
<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: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 Annex A.
- **<Expression>** [Optional]
Arguments to the function, which may include other functions.

7.4.36 Element <Function>

The <Function> element shall be used to name a function as an argument to the function defined by the parent <Apply> element. In the case where the parent <Apply> element is a higher-order bag function, the named function is applied to every element of the bag or bags identified in the other arguments of the parent 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 attributes:

- **FunctionId** [Required]
The identifier of the function.

7.4.37 Complex type AttributeDesignatorType

The **AttributeDesignatorType** complex type is the type for elements that identify attributes by name. It contains the information required to match attributes in the request context.

It also contains information to control behaviour in the event that no matching attribute is present in the context.

Elements of this type shall not alter the match semantics of named attributes, but may narrow the search space.

```

<xs:complexType name="AttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <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="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

A named attribute shall match an attribute if the values of their respective `AttributeId`, `DataType` and `Issuer` attributes match. The attribute designator's `AttributeId` must match, by URI equality, the `AttributeId` of the attribute. The attribute designator's `DataType` must match, by URI 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:

- `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` [Optional]
This attribute governs whether the element returns "Indeterminate" or an empty bag in the event the named attribute is absent from the request context.

7.4.38 Element `<SubjectAttributeDesignator>`

The `<SubjectAttributeDesignator>` element retrieves a bag of values for a named categorized subject attribute from the request context. A subject attribute is an attribute contained within a `<Subject>` element of the request context. A categorized subject is a subject that is identified by a particular subject-category attribute. A named categorized subject attribute is a named subject attribute for a particular categorized subject.

The `<SubjectAttributeDesignator>` element shall return a bag containing all the subject attribute values that are matched by the named categorized subject 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".

The **`SubjectAttributeDesignatorType`** extends the match semantics of the **`AttributeDesignatorType`** such that it narrows the attribute search space to the specific categorized subject such that the value of this element's `SubjectCategory` attribute matches, by URI equality, the value of the request context's `<Subject>` element's `SubjectCategory` attribute.

If the request context contains multiple subjects with the same `SubjectCategory` XML attribute, then they shall be treated as if they were one categorized subject.

The `<SubjectAttributeDesignator>` may appear in the `<SubjectMatch>` element and may be passed to the `<Apply>` element as an argument.

```
<xs:element name="SubjectAttributeDesignator"
  type="xacml:SubjectAttributeDesignatorType"
  substitutionGroup="xacml:Expression"/>
<xs:complexType name="SubjectAttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:AttributeDesignatorType">
      <xs:attribute name="SubjectCategory" type="xs:anyURI"
        use="optional" default="urn:oasis:names:tc:xacml:1.0:subject-category:access-
        subject"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The `<SubjectAttributeDesignator>` element is of type **`SubjectAttributeDesignatorType`**. The **`SubjectAttributeDesignatorType`** complex type extends the **`AttributeDesignatorType`** complex type with a `SubjectCategory` attribute.

- SubjectCategory [Optional]

This attribute shall specify the categorized subject from which to match named subject attributes. If SubjectCategory is not present, then its default value of "urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" shall be used.

7.4.39 Element <ResourceAttributeDesignator>

The <ResourceAttributeDesignator> element retrieves a bag of values for a named resource attribute from the request context. A resource attribute is an attribute contained within the <Resource> element of the request context. A named resource attribute is a named attribute that matches a resource attribute. A named resource attribute shall be considered present if there is at least one resource attribute that matches the criteria set out below. A resource attribute value is an attribute value that is contained within a resource attribute.

The <ResourceAttributeDesignator> element shall return a bag containing all the resource attribute values that are matched by the named resource 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".

A named resource attribute shall match a resource attribute as per the match semantics specified in the **AttributeDesignatorType** complex type.

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

```
<xs:element name="ResourceAttributeDesignator"
type="xacml:AttributeDesignatorType" substitutionGroup="xacml:Expression"/>
```

The <ResourceAttributeDesignator> element is of the **AttributeDesignatorType** complex type.

7.4.40 Element <ActionAttributeDesignator>

The <ActionAttributeDesignator> element retrieves a bag of values for a named action attribute from the request context. An action attribute is an attribute contained within the <Action> element of the request context. A named action attribute has specific criteria (described below) with which to match an action attribute. A named action attribute shall be considered present, if there is at least one action attribute that matches the criteria. An action attribute value is an attribute value that is contained within an action attribute.

The <ActionAttributeDesignator> element shall return a bag of all the action attribute values that are matched by the named action 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".

A named action attribute shall match an action attribute as per the match semantics specified in the **AttributeDesignatorType** complex type.

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

```
<xs:element name="ActionAttributeDesignator" type="xacml:AttributeDesignatorType"
substitutionGroup="xacml:Expression"/>
```

The <ActionAttributeDesignator> element is of the **AttributeDesignatorType** complex type.

7.4.41 Element <EnvironmentAttributeDesignator>

The <EnvironmentAttributeDesignator> element retrieves a bag of values for a named environment attribute from the request context. An environment attribute is an attribute contained within the <Environment> element of request context. A named environment attribute has specific criteria (described below) with which to match an environment attribute. A named environment attribute shall be considered present, if there is at least one environment attribute that matches the criteria. An environment attribute value is an attribute value that is contained within an environment attribute.

The <EnvironmentAttributeDesignator> element shall evaluate to a bag of all the environment attribute values that are matched by the named environment 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".

A named environment attribute shall match an environment attribute as per the match semantics specified in the **AttributeDesignatorType** complex type.

The <EnvironmentAttributeDesignator> may be passed to the <Apply> element as an argument.

```
<xs:element name="EnvironmentAttributeDesignator" type="xacml:AttributeDesignatorType"
substitutionGroup="xacml:Expression"/>
```

The <EnvironmentAttributeDesignator> element is of the **AttributeDesignatorType** complex type.

7.4.42 Element <AttributeSelector>

The <AttributeSelector> element identifies attributes by their location in the request context. Support for the <AttributeSelector> element is optional.

The <AttributeSelector> element's RequestContextPath XML attribute shall contain a legal XPath expression whose context node is the <xacml-context:Request> element. The AttributeSelector element shall evaluate to a bag of values whose data-type is specified by the element's DataType attribute. If the DataType specified in the AttributeSelector is a primitive data type (defined in W3C Schema:2001, W3C Datatypes:2001, 3.2), then the lexical value returned by the XPath expression shall be converted to a value of the DataType specified in the <AttributeSelector>. If an error results converting the value returned by the Xpath expression, such as when the value is not a valid instance of the DataType, then the value of the <AttributeSelector> shall be "Indeterminate".

```
xs:string()
xs:boolean()
xs:integer()
xs:double()
xs:dateTime()
xs:date()
xs:time()
xs:hexBinary()
xs:base64Binary()
xs:anyURI()
```

If the DataType specified in the AttributeSelector is not one of the preceding primitive DataTypes, then the AttributeSelector shall return a bag of instances of the specified DataType. If an error occurs when converting the values returned by the XPath expression to the specified DataType, then the result of the AttributeSelector shall be "Indeterminate".

Each node selected by the specified XPath expression must be either a text node, an attribute node, a processing instruction node or a comment node. The string representation of the value of each node must be converted to an attribute value of the specified data-type, and the result of the AttributeSelector is the bag of the attribute values generated from all the selected nodes.

If the node selected by the specified XPath expression is not one of those listed above (i.e., a text node, an attribute node, a processing instruction node or a comment node), then the result of the enclosing policy shall be "Indeterminate" with a StatusCode value of "urn:oasis:names:tc:xacml:1.0:status:syntax-error".

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

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

The <AttributeSelector> element has the following attributes:

– RequestContextPath [Required]

An XPath expression whose context node is the <xacml-context:Request> element. There shall be no restriction on the XPath syntax.

- **DataType** [Required]
The bag returned by the <AttributeSelector> element shall contain values of this data-type.
- **MustBePresent** [Optional]
This attribute governs whether the element returns "Indeterminate" or an empty bag in the event the XPath expression selects no node.

7.4.43 Element <AttributeValue>

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

The <xacml:AttributeValue> element is of **AttributeValueType** complex type.

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

- **DataType** [Required]
The data-type of the attribute value.

7.4.44 Element <Obligations>

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

Support for the <Obligations> element is optional.

```
<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** complexType.

The <Obligations> element contains the following element:

- <Obligation> [One to Many]
A sequence of obligations.

7.4.45 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. The FulfillOn attribute shall indicate the effect for which this obligation must be fulfilled by the PEP.

```
<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:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
</xs:complexType>
```

The <Obligation> element is of **ObligationType** complexType. See 7.6.14 for a description of how the set of obligations to be returned by the 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 the PEP.
- FulfillOn [Required]
The effect for which this obligation must be fulfilled by the PEP.
- <AttributeAssignment> [Optional]
Obligation arguments assignment. The values of the obligation arguments shall be interpreted by the PEP.

7.4.46 Element <AttributeAssignment>

The <AttributeAssignment> element is used for including arguments in obligations. 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 the PEP, but it is not further specified by XACML.

```
<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: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.

7.5 Context syntax

Schema fragments in the following clauses are non-normative.

7.5.1 Element <Request>

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

The <Request> element contains <Subject>, <Resource>, <Action> and <Environment> elements. There may be multiple <Subject> elements and, under some conditions, multiple <Resource> elements. Each child element contains a sequence of <xacml-context:Attribute> elements associated with the subject, resource, action and environment respectively. These <Attribute> elements may form a part of policy evaluation.

```
<xs:element name="Request" type="xacml-context:RequestType"/>
<xs:complexType name="RequestType">
  <xs:sequence>
    <xs:element ref="xacml-context:Subject" maxOccurs="unbounded"/>
    <xs:element ref="xacml-context:Resource" maxOccurs="unbounded"/>
    <xs:element ref="xacml-context:Action"/>
    <xs:element ref="xacml-context:Environment"/>
  </xs:sequence>
</xs:complexType>
```

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

The <Request> element contains the following elements:

- <Subject> [One to Many]
Specifies information about a subject of the request context by listing a sequence of <Attribute> elements associated with the subject. One or more <Subject> elements are allowed. A subject is an entity associated with the access request. For example, one subject might represent the human user that initiated the application from which the request was issued; another subject might represent the application's executable code responsible for creating the request; another subject might represent the machine on which the application was executing; and another subject might represent the entity that is to be the recipient of the resource. Attributes of each of these entities must be enclosed in separate <Subject> elements.
- <Resource> [One to Many]
Specifies information about the resource or resources for which access is being requested by listing a sequence of <Attribute> elements associated with the resource. It may include a <ResourceContent> element.
- <Action> [Required]
Specifies the requested action to be performed on the resource by listing a set of <Attribute> elements associated with the action.
- <Environment> [Required]
Contains a set of <Attribute> elements for the environment.

7.5.2 Element <Subject>

The <Subject> element specifies a subject by listing a sequence of <Attribute> elements associated with the subject.

```
<xs:element name="Subject" type="xacml-context:SubjectType"/>
<xs:complexType name="SubjectType">
  <xs:sequence>
    <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="SubjectCategory" type="xs:anyURI"
default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
</xs:complexType>
```

The <Subject> element is of **SubjectType** complex type.

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

- SubjectCategory [Optional]
This attribute indicates the role that the parent <Subject> played in the formation of the access request. If this attribute is not present in a given <Subject> element, then the default value of "urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" shall be used, indicating that the parent <Subject> element represents the entity ultimately responsible for initiating the access request.
If more than one <Subject> element contains a "urn:oasis:names:tc:xacml:2.0:subject-category" attribute with the same value, then the PDP shall treat the contents of those elements as if they were contained in the same <Subject> element.
- <Attribute> [Any Number]
A sequence of attributes that apply to the subject.

Typically, a <Subject> element will contain an <Attribute> with an AttributeId of "urn:oasis:names:tc:xacml:1.0:subject:subject-id", containing the identity of the subject.

A <Subject> element may contain additional <Attribute> elements.

7.5.3 Element <Resource>

The <Resource> element specifies information about the resource to which access is requested, by listing a sequence of <Attribute> elements associated with the resource. It may include the resource content.

```
<xs:element name="Resource" type="xacml-context:ResourceType"/>
<xs:complexType name="ResourceType">
  <xs:sequence>
    <xs:element ref="xacml-context:ResourceContent" minOccurs="0"/>
    <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <Resource> element is of **ResourceType** complex type.

The <Resource> element contains the following elements:

- <ResourceContent> [Optional]
The resource content.
- <Attribute> [Any Number]
A sequence of resource attributes.

The <Resource> element may contain one or more <Attribute> elements with an AttributeId of "urn:oasis:names:tc:xacml:2.0:resource:resource-id". Each such <Attribute> shall be an absolute and fully-resolved representation of the identity of the single resource to which access is being requested. If there is more than one such absolute and fully-resolved representation, and if any <Attribute> with this AttributeId is specified, then an <Attribute> for each such distinct representation of the resource identity shall be specified. All such <Attribute> elements shall refer to the same single resource instance. A profile for a particular resource may specify a single normative representation for instances of the resource; in this case, any <Attribute> with this AttributeId shall use only this one representation.

A <Resource> element may contain additional <Attribute> elements.

7.5.4 Element <ResourceContent>

The <ResourceContent> element is a notional placeholder for the content of the resource. If an XACML policy references the contents of the resource by means of an <AttributeSelector> element, then the <ResourceContent> element must be included in the RequestContextPath string.

```
<xs:complexType name="ResourceContentType" mixed="true">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

The <ResourceContent> element is of **ResourceContentType** complex type.

The <ResourceContent> element allows arbitrary elements and attributes.

7.5.5 Element <Action>

The <Action> element specifies the requested action on the resource, by listing a set of <Attribute> elements associated with the action.

```
<xs:element name="Action" type="xacml-context:ActionType"/>
<xs:complexType name="ActionType">
  <xs:sequence>
    <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <Action> element is of **ActionType** complex type.

The <Action> element contains the following elements:

- <Attribute> [Any Number]
List of attributes of the action to be performed on the resource.

7.5.6 Element <Environment>

The <Environment> element contains a set of attributes of the environment.

```
<xs:element name="Environment" type="xacml-context:EnvironmentType"/>
<xs:complexType name="EnvironmentType">
  <xs:sequence>
    <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <Environment> element is of **EnvironmentType** complex type.

The <Environment> element contains the following elements:

- <Attribute> [Any Number]
A list of environment attributes. Environment attributes are attributes that are not associated with either the resource, the action or any of the subjects of the access request.

7.5.7 Element <Attribute>

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

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

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

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

- AttributeId [Required]
The Attribute identifier. A number of identifiers are reserved by XACML to denote commonly used attributes.
- DataType [Required]
The data-type of the contents of the <xacml-context:AttributeValue> element. This shall be either a primitive type defined by this Recommendation or a type (primitive or structured) defined in a namespace declared in the <xacml-context> element.
- Issuer [Optional]
The Attribute issuer. For example, this attribute value may be an x500Name that binds to a public key, or it may be some other identifier exchanged out-of-band by issuing and relying parties.
- <xacml-context:AttributeValue> [One to Many]
One or more attribute values. Each attribute value may have contents that are empty, occur once or occur multiple times.

7.5.8 Element <AttributeValue>

The <xacml-context:AttributeValue> element contains the value of an attribute.

```
<xs:element name="AttributeValue" type="xacml-context:AttributeValueType"/>
<xs:complexType name="AttributeValueType" mixed="true">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

The <xacml-context:AttributeValue> element is of **AttributeValueType** complex type.

The data-type of the <xacml-context:AttributeValue> shall be specified by using the **DataType** attribute of the parent <Attribute> element.

7.5.9 Element <Response>

The <Response> element is a top-level element in the XACML context schema. The <Response> element is an abstraction layer used by the policy language. Any proprietary system using XACML must transform an XACML context <Response> element into the form of its authorization decision.

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

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

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

The <Response> element contains the following elements:

- <Result> [One to Many]
An authorization decision result.

7.5.10 Element <Result>

The <Result> element represents an authorization decision result for the resource specified by the **ResourceId** attribute. It may include a set of obligations that must be fulfilled by the PEP. If the PEP does not understand or cannot fulfil an obligation, then it must act as if the PDP had denied access to the requested resource.

```
<xs:complexType name="ResultType">
  <xs:sequence>
    <xs:element ref="xacml-context:Decision"/>
    <xs:element ref="xacml-context:Status" minOccurs="0"/>
    <xs:element ref="xacml:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="ResourceId" type="xs:string" use="optional"/>
</xs:complexType>
```

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

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

- **ResourceId** [Optional]
The identifier of the requested resource. If this attribute is omitted, then the resource identity is that specified by the "urn:oasis:names:tc:xacml:1.0:resource:resource-id" resource attribute in the corresponding <Request> element.
- <Decision> [Required]
The authorization decision: "Permit", "Deny", "Indeterminate" or "NotApplicable".

- **<Status> [Optional]**
Indicates whether errors occurred during evaluation of the decision request, and optionally, information about those errors. If the **<Response>** element contains **<Result>** elements whose **<Status>** elements are all identical, and the **<Response>** element is contained in a protocol wrapper that can convey status information, then the common status information may be placed in the protocol wrapper and this **<Status>** element may be omitted from all **<Result>** elements.
- **<Obligations> [Optional]**
A list of obligations that must be fulfilled by the PEP. If the PEP does not understand or cannot fulfil an obligation, then it must act as if the PDP had denied access to the requested resource.

7.5.11 Element **<Decision>**

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

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

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

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

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

7.5.12 Element **<Status>**

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

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

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

The **<Status>** element contains the following elements:

- **<StatusCode> [Required]**
Status code.
- **<StatusMessage> [Optional]**
A status message describing the status code.
- **<StatusDetail> [Optional]**
Additional status information.

7.5.13 Element **<StatusCode>**

The **<StatusCode>** element contains a major status code value and an optional sequence of minor status codes.

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

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

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

- value [Required]
See B.8 for a list of values.
- <StatusCode> [Any Number]
Minor status code. This status code qualifies its parent status code.

7.5.14 Element <StatusMessage>

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

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

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

7.5.15 Element <StatusDetail>

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

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

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

The <StatusDetail> element allows arbitrary XML content.

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

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

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

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

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

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

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

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

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

7.5.16 Element <MissingAttributeDetail>

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

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

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

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

- AttributeValue [Optional]
The required value of the missing attribute.
- <AttributeId> [Required]
The identifier of the missing attribute.
- <DataType> [Required]
The data-type of the missing attribute.
- Issuer [Optional]
This attribute, if supplied, shall specify the required Issuer of the missing attribute.

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

7.6 XACML functional requirements

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

7.6.1 Policy enforcement point

This clause describes the requirements for the PEP.

An application functions in the role of the PEP if it guards access to a set of resources and asks the PDP for an authorization decision. The PEP must abide by the PDP authorization decisions.

7.6.1.1 Base PEP

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

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

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

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

7.6.1.2 Deny-biased PEP

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

All other decisions shall result in the denial of access.

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

7.6.1.3 Permit-biased PEP

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

All other decisions shall result in the permission of access.

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

7.6.2 Attribute evaluation

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

7.6.2.1 Structured attributes

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

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

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

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

- 2) An `<AttributeSelector>` element may be used to select the contents of a leaf sub-element of the structured data-type by means of an XPath expression. That value may then be compared using one of the supported XACML functions appropriate for its primitive data-type. This method requires support by the PDP for the optional XPath expressions feature.
- 3) An `<AttributeSelector>` element may be used to select any node in the structured data-type by means of an XPath expression. This node may then be compared using one of the XPath-based functions described in A.3. This method requires support by the PDP for the optional XPath expressions and XPath functions features.

7.6.2.2 Attribute bags

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

The `<AttributeSelector>` element uses an XPath expression to specify the selection of data from an XML resource. The result of an XPath expression is termed a node-set, which contains all the leaf nodes from the XML resource that match the predicate in the XPath expression. Based on the various indexing functions provided in the W3C XPath:1999, it shall be implied that a resultant node-set is the collection of the matching nodes. This Recommendation also defines the `<AttributeDesignator>` element to have the same matching methodology for attributes in the XACML Request context.

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

7.6.2.3 Multivalued attributes

If a single `<Attribute>` element in a request context contains multiple `<xacml-context:AttributeValue>` child elements, then the bag of values resulting from evaluation of the `<Attribute>` element must be identical to the bag of