International Telecommunication Union

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
International Telecommunication Union

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.

ITU-T Recommendation X.1142

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/.

CONTENTS

ITU-T Recommendation X.1142

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-xmldsig-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.

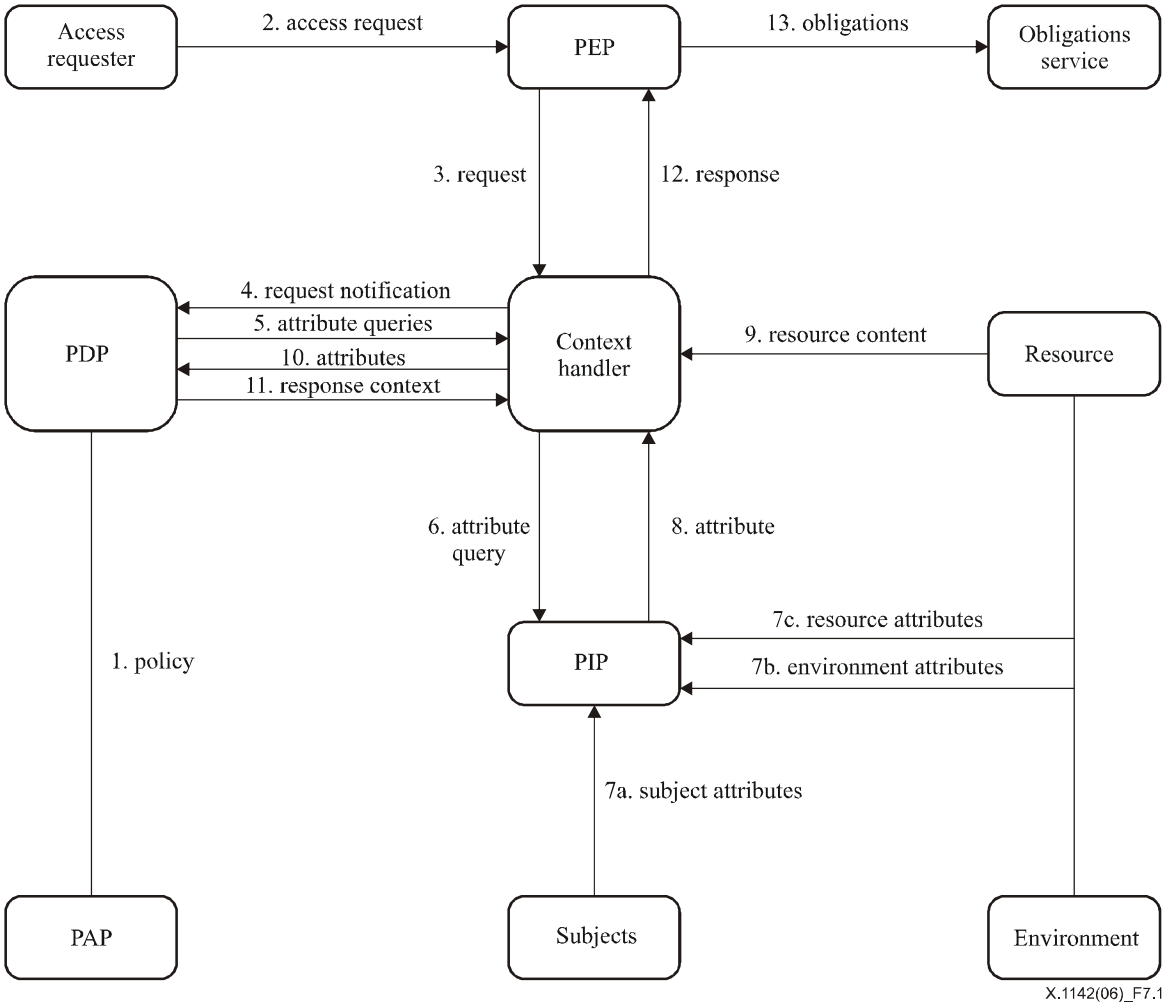

Figure 7-1/X.1142 – Data-flow diagram

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).

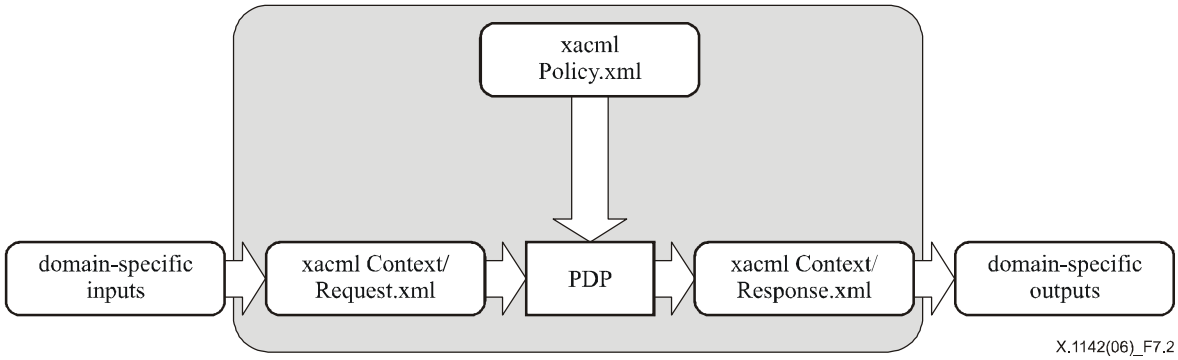

Figure 7-2/X.1142 – XACML context

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.

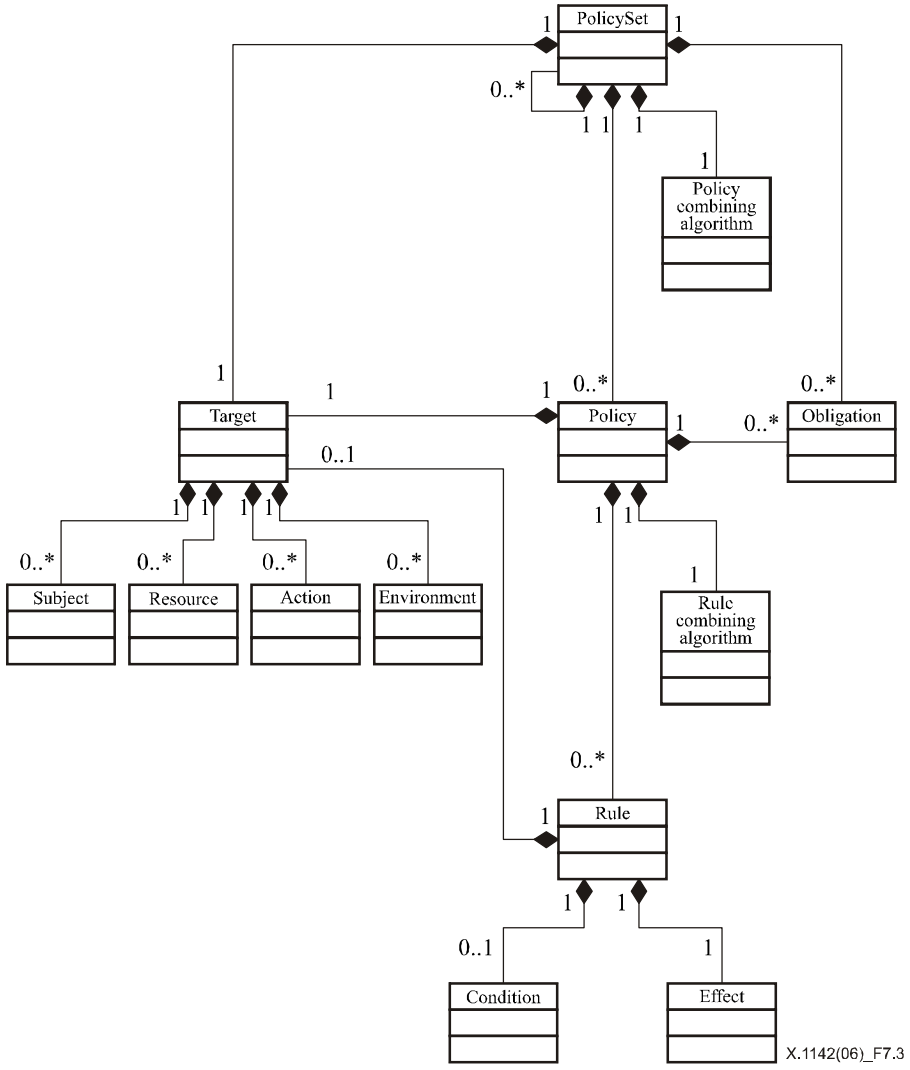

Figure 7-3/X.1142 – Policy language model

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: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.+'.

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 <CominberParameters> 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 <RuleCominberParameters> 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 <PolicySetCominberParameters> 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 <MissingAttributeDetal> 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-regexp-match", described below. This requires that the element be given the data-type "http://www.w3.org/2001/XMLSchema#string". For example, a structured data-type that is actually a **ds:KeyInfo/KeyName** would appear in the Context as:

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

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

2) 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

values that results from evaluating a context in which each <xacml-context:AttributeValue> element appears in a separate <Attribute> element, each carrying identical meta-data.

7.6.2.4 Attribute matching

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

7.6.2.5 Attribute retrieval

The PDP shall request the values of attributes in the request context from the context handler. The PDP shall reference the attributes as if they were in a physical request context document, but the context handler is responsible for obtaining and supplying the requested values by whatever means it deems appropriate. The context handler shall return the values of attributes that match the attribute designator or attribute selector and form them into a bag of values with the specified data-type. If no attributes from the request context match, then the attribute shall be considered missing. If the attribute is missing, then MustBePresent governs whether the attribute designator or attribute selector returns an empty bag or an "Indeterminate" result. If MustBePresent is "False" (default value), then a missing attribute shall result in an empty bag. If MustBePresent is "True", then a missing attribute shall result in "Indeterminate". This "Indeterminate" result shall be handled in accordance with the specification of the encompassing expressions, rules, policies and policy sets. If the result is "Indeterminate", then the AttributeId, DataType and Issuer of the attribute may be listed in the authorization decision. However, a PDP may choose not to return such information for security reasons.

7.6.2.6 Environment attributes

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

7.6.3 Expression evaluation

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

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

- <xacml:AttributeValue>
- <xacml:SubjectAttributeDesignator>
- <xacml:ResourceAttributeDesignator>
- <xacml:ActionAttributeDesignator>
- <xacml:EnvironmentAttributeDesignator>
- <xacml:AttributeSelector>
- <xacml:Apply>
- <xacml:Condition>
- <xacml:Function>
- <xacml:VariableReference>

7.6.4 Arithmetic evaluation

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

- flags: all set to 0;
- trap-enablers: all set to 0 with the exception of the "division-by-zero" trap enabler, which shall be set to 1;
- precision: is set to the designated double precision;
- rounding: is set to round-half-even.

7.6.5 Match evaluation

Attribute matching elements appear in the `<Target>` element of rules, policies and policy sets. They are the following:

- `<SubjectMatch>`
- `<ResourceMatch>`
- `<ActionMatch>`
- `<EnvironmentMatch>`

These elements represent boolean expressions over attributes of the subject, resource, action and environment, respectively. A matching element contains a `MatchId` attribute that specifies the function to be used in performing the match evaluation, an `<xacml:AttributeValue>` and an `<AttributeDesignator>` or `<AttributeSelector>` element that specifies the attribute in the context that is to be matched against the specified value.

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

The XACML standard functions that meet the requirements for use as a `MatchId` attribute value are:

```
urn:oasis:names:tc:xacml:2.0:function:-type-equal
urn:oasis:names:tc:xacml:2.0:function:-type-greater-than
urn:oasis:names:tc:xacml:2.0:function:-type-greater-than-or-equal
urn:oasis:names:tc:xacml:2.0:function:-type-less-than
urn:oasis:names:tc:xacml:2.0:function:-type-less-than-or-equal
urn:oasis:names:tc:xacml:2.0:function:-type-match
```

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

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

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

```
<SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
       John.*
    </AttributeValue>
```

```
    <SubjectAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
</SubjectMatch>
```

Alternatively, the same match semantics can be expressed as an <Apply> element in a condition by using the "urn:oasis:names:tc:xacml:1.0:function:any-of" function, as follows:

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

7.6.6 Target evaluation

The target value shall be "Match" if the subjects, resources, actions and environments specified in the target all match values in the request context. If any one of the subjects, resources, actions and environments specified in the target are "Indeterminate", then the target shall be "Indeterminate". Otherwise, the target shall be "No match". The target match table is shown in Table 7-1.

Table 7-1/X.1142 – Target match table

Subjects value	Resources value	Actions value	Environments value	Target value
"Match"	"Match"	"Match"	"Match"	"Match"
"No match"	"Match" or "No match"	"Match" or "No match"	"Match" or "No match"	"No match"
"Match" or "No match"	"No match"	"Match" or "No match"	"Match" or "No match"	"No match"
"Match" or "No match"	"Match" or "No match"	"No match"	"Match" or "No match"	"No match"
"Match" or "No match"	"Match" or "No match"	"Match" or "No match"	"No match"	"No match"
"Indeterminate"	Don't care	Don't care	Don't care	"Indeterminate"
Don't care	"Indeterminate"	Don't care	Don't care	"Indeterminate"
Don't care	Don't care	"Indeterminate"	Don't care	"Indeterminate"
Don't care	Don't care	Don't care	"Indeterminate"	"Indeterminate"

The subjects, resources, actions and environments shall match values in the request context if at least one of their <Subject>, <Resource>, <Action> or <Environment> elements, respectively, matches a value in the request context. The subjects match table is shown in Table 7-2. The resources, actions and environments match tables are analogous.

Table 7-2/X.1142 – Subjects match table

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

A subject, resource, action or environment shall match a value in the request context if the value of all its <SubjectMatch>, <ResourceMatch>, <ActionMatch> or <EnvironmentMatch> elements, respectively, are "True".

The subject match table is shown in Table 7-3. The resource, action and environment match tables are analogous.

Table 7-3/X.1142 – Subject match table

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

7.6.7 VariableReference evaluation

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

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

7.6.8 Condition evaluation

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

7.6.9 Rule evaluation

A rule has a value that can be calculated by evaluating its contents. Rule evaluation involves separate evaluation of the rule's target and condition. The rule truth table is shown in Table 7-4.

Table 7-4/X.1142 – Rule truth table

Target	Condition	Rule value
"Match"	"True"	Effect
"Match"	"False"	"NotApplicable"
"Match"	"Indeterminate"	"Indeterminate"
"No-match"	Don't care	"NotApplicable"
"Indeterminate"	Don't care	"Indeterminate"

If the target value is "No-match" or "Indeterminate" then the rule value shall be "NotApplicable" or "Indeterminate", respectively, regardless of the value of the condition. For these cases, therefore, the condition need not be evaluated.

If the target value is "Match" and the condition value is "True", then the effect specified in the enclosing <Rule> element shall determine the rule's value.

7.6.10 Policy evaluation

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

The policy's target shall be evaluated to determine the applicability of the policy. If the target evaluates to "Match", then the value of the policy shall be determined by evaluation of the policy's rules, according to the specified rule-combining algorithm. If the target evaluates to "No-match", then the value of the policy shall be "NotApplicable". If the target evaluates to "Indeterminate", then the value of the policy shall be "Indeterminate".

The policy truth table is shown in Table 7-5.

Table 7-5/X.1142 – Policy truth table

Target	Rule values	Policy value
"Match"	At least one rule value is its Effect	Specified by the rule-combining algorithm
"Match"	All rule values are "NotApplicable"	"NotApplicable"
"Match"	At least one rule value is "Indeterminate"	Specified by the rule-combining algorithm
"No-match"	Don't care	"NotApplicable"
"Indeterminate"	Don't care	"Indeterminate"

A rules value of "At least one rule value is its Effect" means either that the `<Rule>` element is absent, or one or more of the rules contained in the policy is applicable to the decision request (i.e., it returns the value of its "Effect"). A rules value of "All rule values are 'NotApplicable'" shall be used if no rule contained in the policy is applicable to the request and if no rule contained in the policy returns a value of "Indeterminate". If no rule contained in the policy is applicable to the request, but one or more rule returns a value of "Indeterminate", then the rules shall evaluate to "At least one rule value is 'Indeterminate'".

If the target value is "No-match" or "Indeterminate" then the policy value shall be "NotApplicable" or "Indeterminate", respectively, regardless of the value of the rules. For these cases, therefore, the rules need not be evaluated.

If the target value is "Match" and the rule value is "At least one rule value is it's Effect" or "At least one rule value is 'Indeterminate'", then the rule-combining algorithm specified in the policy shall determine the policy value.

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

7.6.11 Policy set evaluation

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

The policy set's target shall be evaluated to determine the applicability of the policy set. If the target evaluates to "Match" then the value of the policy set shall be determined by evaluation of the policy set's policies and policy sets, according to the specified policy-combining algorithm. If the target evaluates to "No-match", then the value of the policy set shall be "NotApplicable". If the target evaluates to "Indeterminate", then the value of the policy set shall be "Indeterminate".

The policy set truth table is shown in Table 7-6.

Table 7-6/X.1142 – Policy set truth table

Target	Policy values	Policy set value
"Match"	At least one policy value is its Decision	Specified by the policy-combining algorithm
"Match"	All policy values are "NotApplicable"	"NotApplicable"
"Match"	At least one policy value is "Indeterminate"	Specified by the policy-combining algorithm
"No-match"	Don't care	"NotApplicable"
"Indeterminate"	Don't care	"Indeterminate"

A policies value of "At least one policy value is its Decision" shall be used if there are no contained or referenced policies or policy sets, or if one or more of the policies or policy sets contained in or referenced by the policy set is applicable to the decision request (i.e., returns a value determined by its combining algorithm). A policies value of "All policy values are 'NotApplicable'" shall be used if no policy or policy set contained in or referenced by the policy set is applicable to the request and if no policy or policy set contained in or referenced by the policy set returns a value of "Indeterminate". If no policy or policy set contained in or referenced by the policy set is applicable to the request but one or more policy or policy set returns a value of "Indeterminate", then the policies shall evaluate to "At least one policy value is 'Indeterminate'".

If the target value is "No-match" or "Indeterminate" then the policy set value shall be "NotApplicable" or "Indeterminate", respectively, regardless of the value of the policies. For these cases, therefore, the policies need not be evaluated.

If the target value is "Match" and the policies value is "At least one policy value is its Decision" or "At least one policy value is 'Indeterminate'", then the policy-combining algorithm specified in the policy set shall determine the policy set value.

Note that none of the policy-combining algorithms defined by XACML 2.0 takes parameters. However, non-standard combining algorithms may take parameters. In such a case, the values of these parameters associated with the policies must be taken into account when evaluating the policy set. The parameters and their types should be defined in the specification of the combining algorithm. If the implementation supports combiner parameters and if combiner parameters are present in a policy, then the parameter values must be supplied to the combining algorithm implementation.

7.6.12 Hierarchical resources

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

7.6.13 Authorization decision

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

The PDP must evaluate the policy set as specified in 7.4 and in 7.6. The PDP must return a response context, with one `<Decision>` element of value "Permit", "Deny", "Indeterminate" or "NotApplicable".

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

7.6.14 Obligations

A policy or policy set may contain one or more obligations. When such a policy or policy set is evaluated, an obligation shall be passed up to the next level of evaluation (the enclosing or referencing policy, policy set or authorization decision) only if the effect of the policy or policy set being evaluated matches the value of the `FulfillOn` attribute of the obligation.

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

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

7.6.15 Exception handling

XACML specifies behaviour for the PDP in the following situations.

7.6.15.1 Unsupported functionality

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

7.6.15.2 Syntax and type errors

If a policy that contains invalid syntax is evaluated by the XACML PDP at the time a decision request is received, then the result of that policy shall be "Indeterminate" with a `StatusCode` value of:

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

If a policy that contains invalid static data-types is evaluated by the XACML PDP at the time a decision request is received, then the result of that policy shall be "Indeterminate" with a `StatusCode` value of:

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

7.6.15.3 Missing attributes

The absence of matching attributes in the request context for any of the attribute designators or selectors that are found in the policy shall result in a `<Decision>` element containing the "Indeterminate" value. If, in this case, and a status code is supplied, then the value:

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

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

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

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

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

it must not list the names and data-types of any attribute of the subject, resource, action or environment for which values were supplied in the original request. Note, this requirement forces the PDP to eventually return an authorization decision of "Permit", "Deny" or "Indeterminate" with some other status code, in response to successively-refined requests.

7.7 XACML extensibility points

This clause describes the points within the XACML model and schema where extensions can be added. This clause is informative.

7.7.1 Extensible XML attribute types

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

- `AttributeId`;
- `DataType`;
- `FunctionId`;
- `MatchId`;
- `ObligationId`;
- `PolicyCombiningAlgId`;
- `RuleCombiningAlgId`;
- `StatusCode`;
- `SubjectCategory`.

7.7.2 Structured attributes

`<xacml:AttributeValue>` and `<xacml-context:AttributeValue>` elements may contain an instance of a structured XML data-type. Listed here are some techniques that require XACML extensions.

1) For a given structured data-type, a community of XACML users may define new attribute identifiers for each leaf sub-element of the structured data-type that has a type conformant with one of the XACML-defined primitive data-types. Using these new attribute identifiers, the PEPs or context handlers used by that community of users can flatten instances of the structured data-type into a sequence of individual `<Attribute>` elements. Each such `<Attribute>` element can be compared using the XACML-defined functions. Using this method, the structured data-type itself never appears in an `<xacml-context:AttributeValue>` element.

2) A community of XACML users may define a new function that can be used to compare a value of the structured data-type against some other value. This method may only be used by PDPs that support the new function.

7.8 Conformance

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

This clause lists those portions of this Recommendation that must be included in an implementation of a PDP that claims to conform with XACML v2.0.

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

7.8.1 Schema elements

The implementation must support those schema elements that are marked "M".

Element name	M/O
xacml-context:Action	M
xacml-context:Attribute	M
xacml-context:AttributeValue	M
xacml-context:Decision	M
xacml-context:Environment	M
xacml-context:MissingAttributeDetail	M
xacml-context:Obligations	O
xacml-context:Request	M
xacml-context:Resource	M
xacml-context:ResourceContent	O
xacml-context:Response	M
xacml-context:Result	M
xacml-context:Status	M
xacml-context:StatusCode	M
xacml-context:StatusDetail	O
xacml-context:StatusMessage	O
xacml-context:Subject	M
xacml:Action	M
xacml:ActionAttributeDesignator	M
xacml:ActionMatch	M
xacml:Actions	M
xacml:Apply	M
xacml:AttributeAssignment	O
xacml:AttributeSelector	O
xacml:AttributeValue	M
xacml:CombinerParameters	O
xacml:CombinerParameter	O
xacml:Condition	M
xacml:Description	M
xacml:Environment	M
xacml:EnvironmentMatch	M
xacml:EnvironmentAttributeDesignator	M
xacml:Environments	M
xacml:Expression	M
xacml:Function	M
xacml:Obligation	O
xacml:Obligations	O
xacml:Policy	M
xacml:PolicyCombinerParameters	O

Element name	M/O
xacml:PolicyDefaults	O
xacml:PolicyIdReference	M
xacml:PolicySet	M
xacml:PolicySetDefaults	O
xacml:PolicySetIdReference	M
xacml:Resource	M
xacml:ResourceAttributeDesignator	M
xacml:ResourceMatch	M
xacml:Resources	M
xacml:Rule	M
xacml:RuleCombinerParameters	O
xacml:Subject	M
xacml:SubjectMatch	M
xacml:Subjects	M
xacml:Target	M
xacml:VariableDefinition	M
xacml:VariableReference	M
xacml:XPathVersion	O

7.8.2 Identifier prefixes

The following identifier prefixes are reserved by XACML.

Identifier
urn:oasis:names:tc:xacml:2.0
urn:oasis:names:tc:xacml:2.0:conformance-test
urn:oasis:names:tc:xacml:2.0:context
urn:oasis:names:tc:xacml:2.0:example
urn:oasis:names:tc:xacml:1.0:function
urn:oasis:names:tc:xacml:2.0:function
urn:oasis:names:tc:xacml:2.0:policy
urn:oasis:names:tc:xacml:1.0:subject
urn:oasis:names:tc:xacml:1.0:resource
urn:oasis:names:tc:xacml:1.0:action
urn:oasis:names:tc:xacml:1.0:environment
urn:oasis:names:tc:xacml:1.0:status

7.8.3 Algorithms

The implementation must include the rule- and policy-combining algorithms associated with the following identifiers that are marked "M".

Algorithm	M/O
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides	M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides	M
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides	M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides	M
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable	M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable	M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable	M
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides	M
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides	M
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides	M
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides	M

7.8.4 Status codes

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

Identifier	M/O
urn:oasis:names:tc:xacml:1.0:status:missing-attribute	M
urn:oasis:names:tc:xacml:1.0:status:ok	M
urn:oasis:names:tc:xacml:1.0:status:processing-error	M
urn:oasis:names:tc:xacml:1.0:status:syntax-error	M

7.8.5 Attributes

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

Identifier	M/O
urn:oasis:names:tc:xacml:1.0:environment:current-time	M
urn:oasis:names:tc:xacml:1.0:environment:current-date	M
urn:oasis:names:tc:xacml:1.0:environment:current-dateTime	M

7.8.6 Identifiers

The implementation must use the attributes associated with the following identifiers in the way XACML has defined. This requirement pertains primarily to implementations of a PAP or PEP that uses XACML, since the semantics of the attributes is transparent to the PDP.

Identifier	M/O
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name	O
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address	O
urn:oasis:names:tc:xacml:1.0:subject:authentication-method	O
urn:oasis:names:tc:xacml:1.0:subject:authentication-time	O
urn:oasis:names:tc:xacml:1.0:subject:key-info	O
urn:oasis:names:tc:xacml:1.0:subject:request-time	O
urn:oasis:names:tc:xacml:1.0:subject:session-start-time	O
urn:oasis:names:tc:xacml:1.0:subject:subject-id	O
urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier	O
urn:oasis:names:tc:xacml:1.0:subject-category:access-subject	M
urn:oasis:names:tc:xacml:1.0:subject-category:codebase	O
urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject	O
urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject	O
urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine	O
urn:oasis:names:tc:xacml:1.0:resource:resource-location	O
urn:oasis:names:tc:xacml:1.0:resource:resource-id	M
urn:oasis:names:tc:xacml:1.0:resource:simple-file-name	O
urn:oasis:names:tc:xacml:1.0:action:action-id	O
urn:oasis:names:tc:xacml:1.0:action:implied-action	O

7.8.7 Data-types

The implementation must support the data-types associated with the following identifiers marked "M".

Data-type	M/O
http://www.w3.org/2001/XMLSchema#string	M
http://www.w3.org/2001/XMLSchema#boolean	M
http://www.w3.org/2001/XMLSchema#integer	M
http://www.w3.org/2001/XMLSchema#double	M

Data-type	M/O
http://www.w3.org/2001/XMLSchema#time	M
http://www.w3.org/2001/XMLSchema#date	M
http://www.w3.org/2001/XMLSchema#dateTime	M
urn:oasis:names:tc:xacml:2.0:data-types:dayTimeDuration	M
urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration	M
http://www.w3.org/2001/XMLSchema#anyURI	M
http://www.w3.org/2001/XMLSchema#hexBinary	M
http://www.w3.org/2001/XMLSchema#base64Binary	M
urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name	M
urn:oasis:names:tc:xacml:1.0:data-type:x500Name	M

7.8.8 Functions

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

Function	M/O
urn:oasis:names:tc:xacml:1.0:function:string-equal	M
urn:oasis:names:tc:xacml:1.0:function:boolean-equal	M
urn:oasis:names:tc:xacml:1.0:function:integer-equal	M
urn:oasis:names:tc:xacml:1.0:function:double-equal	M
urn:oasis:names:tc:xacml:1.0:function:date-equal	M
urn:oasis:names:tc:xacml:1.0:function:time-equal	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-equal	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-equal	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-equal	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal	M
urn:oasis:names:tc:xacml:1.0:function:integer-add	M
urn:oasis:names:tc:xacml:1.0:function:double-add	M
urn:oasis:names:tc:xacml:1.0:function:integer-subtract	M
urn:oasis:names:tc:xacml:1.0:function:double-subtract	M
urn:oasis:names:tc:xacml:1.0:function:integer-multiply	M
urn:oasis:names:tc:xacml:1.0:function:double-multiply	M
urn:oasis:names:tc:xacml:1.0:function:integer-divide	M
urn:oasis:names:tc:xacml:1.0:function:double-divide	M
urn:oasis:names:tc:xacml:1.0:function:integer-mod	M
urn:oasis:names:tc:xacml:1.0:function:integer-abs	M
urn:oasis:names:tc:xacml:1.0:function:double-abs	M
urn:oasis:names:tc:xacml:1.0:function:round	M
urn:oasis:names:tc:xacml:1.0:function:floor	M
urn:oasis:names:tc:xacml:1.0:function:string-normalize-space	M
urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case	M
urn:oasis:names:tc:xacml:1.0:function:double-to-integer	M
urn:oasis:names:tc:xacml:1.0:function:integer-to-double	M
urn:oasis:names:tc:xacml:1.0:function:or	M
urn:oasis:names:tc:xacml:1.0:function:and	M
urn:oasis:names:tc:xacml:1.0:function:n-of	M
urn:oasis:names:tc:xacml:1.0:function:not	M
urn:oasis:names:tc:xacml:1.0:function:integer-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal	M

Function	M/O
urn:oasis:names:tc:xacml:1.0:function:integer-less-than	M
urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:double-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:double-less-than	M
urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:string-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:string-less-than	M
urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:time-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:time-less-than	M
urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal	M
urn:oasis:names:tc:xacml:2.0:function:time-in-range	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:date-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:date-less-than	M
urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:string-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:string-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:string-is-in	M
urn:oasis:names:tc:xacml:1.0:function:string-bag	M
urn:oasis:names:tc:xacml:1.0:function:boolean-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:boolean-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:boolean-is-in	M
urn:oasis:names:tc:xacml:1.0:function:boolean-bag	M
urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:integer-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:integer-is-in	M
urn:oasis:names:tc:xacml:1.0:function:integer-bag	M
urn:oasis:names:tc:xacml:1.0:function:double-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:double-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:double-is-in	M
urn:oasis:names:tc:xacml:1.0:function:double-bag	M
urn:oasis:names:tc:xacml:1.0:function:time-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:time-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:time-is-in	M
urn:oasis:names:tc:xacml:1.0:function:time-bag	M
urn:oasis:names:tc:xacml:1.0:function:date-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:date-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:date-is-in	M

Function	M/O
urn:oasis:names:tc:xacml:1.0:function:date-bag	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-is-in	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-bag	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-is-in	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-bag	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-is-in	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-bag	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-is-in	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-bag	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-is-in	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-bag	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-is-in	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-bag	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-is-in	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-bag	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-is-in	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-bag	M
urn:oasis:names:tc:xacml:2.0:function:string-concatenate	M
urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate	M
urn:oasis:names:tc:xacml:1.0:function:any-of	M
urn:oasis:names:tc:xacml:1.0:function:all-of	M
urn:oasis:names:tc:xacml:1.0:function:any-of-any	M
urn:oasis:names:tc:xacml:1.0:function:all-of-any	M
urn:oasis:names:tc:xacml:1.0:function:any-of-all	M
urn:oasis:names:tc:xacml:1.0:function:all-of-all	M
urn:oasis:names:tc:xacml:1.0:function:map	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-match	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match	M
urn:oasis:names:tc:xacml:1.0:function:string-regexp-match	M
urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match	M
urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match	M
urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match	M
urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match	M
urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match	M
urn:oasis:names:tc:xacml:1.0:function:xpath-node-count	O
urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal	O

Function	M/O
urn:oasis:names:tc:xacml:1.0:function:xpath-node-match	O
urn:oasis:names:tc:xacml:1.0:function:string-intersection	M
urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:string-union	M
urn:oasis:names:tc:xacml:1.0:function:string-subset	M
urn:oasis:names:tc:xacml:1.0:function:string-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:boolean-intersection	M
urn:oasis:names:tc:xacml:1.0:function:boolean-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:boolean-union	M
urn:oasis:names:tc:xacml:1.0:function:boolean-subset	M
urn:oasis:names:tc:xacml:1.0:function:boolean-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:integer-intersection	M
urn:oasis:names:tc:xacml:1.0:function:integer-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:integer-union	M
urn:oasis:names:tc:xacml:1.0:function:integer-subset	M
urn:oasis:names:tc:xacml:1.0:function:integer-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:double-intersection	M
urn:oasis:names:tc:xacml:1.0:function:double-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:double-union	M
urn:oasis:names:tc:xacml:1.0:function:double-subset	M
urn:oasis:names:tc:xacml:1.0:function:double-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:time-intersection	M
urn:oasis:names:tc:xacml:1.0:function:time-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:time-union	M
urn:oasis:names:tc:xacml:1.0:function:time-subset	M
urn:oasis:names:tc:xacml:1.0:function:time-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:date-intersection	M
urn:oasis:names:tc:xacml:1.0:function:date-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:date-union	M
urn:oasis:names:tc:xacml:1.0:function:date-subset	M
urn:oasis:names:tc:xacml:1.0:function:date-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-intersection	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-union	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-subset	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-intersection	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-union	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-subset	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-intersection	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-union	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-subset	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-intersection	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-union	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-subset	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-intersection	M

Function	M/O
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-union	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-subset	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-intersection	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-union	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-subset	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-intersection	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-union	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-subset	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-intersection	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-union	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-subset	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-set-equals	M

8 Core and hierarchical role based access control (RBAC) profile

This clause defines a profile for the use of the XACML to meet the requirements for "core" and "hierarchical" role based access control (RBAC).

8.1 RBAC background

This clause is informative.

This clause defines a profile for use with XACML to meet the requirements for "core" and "hierarchical" role based access control (RBAC).

NOTE – For information on RBAC see [RBAC].

8.1.1 Scope

Role based access control allows policies to be specified in terms of subject roles rather than strictly in terms of individual subject identities. This is important for scalability and manageability of access control systems.

The policies specified in this profile can answer three types of questions:

1) If a subject has roles R_1 , R_2, ... R_n enabled, can subject X access a given resource using a given action?
2) Is subject X allowed to have role R_i enabled?
3) If a subject has roles R_1 , R_2, ... R_n enabled, does that mean the subject will have permissions associated with a given role R'? That is, is role R' either equal to or *junior* to any of roles R_1 , R_2, ... R_n?

The policies specified in this profile do not answer the question "What set of roles does subject X have?" That question must be handled by a role enablement authority, and not directly by an XACML PDP. Such an entity may make use of XACML policies, but will need additional information.

The policies specified in this profile assume all the roles for a given subject have already been enabled at the time an authorization decision is requested. They do not deal with an environment in which roles must be enabled dynamically based on the resource or actions a subject is attempting to perform. For this reason, the policies specified in this profile also do not deal with static or dynamic "separation of duty". A future profile may address the requirements of this type of environment.

8.1.2 Role

In this Recommendation, roles are expressed as XACML subject attributes. There are two exceptions: in a Role Assignment `<PolicySet>` or `<Policy>` and in a `HasPrivilegesOfRole` `<Policy>`, the role appears as a resource attribute.

Role attributes may be expressed in either of two ways, depending on the requirements of the application environment. In some environments there may be a small number of "role attributes", where the name of each such attribute is some name indicating "role", and where the value of each such attribute indicates the name of the role held. For example, in this first type of environment, there may be one "role attribute" having the `AttributeId` "`&role;`" (this profile recommends use of this identifier). The possible roles are values for this one attribute, and might be "`&roles;officer`", "`&roles;manager`", and "`&roles;employee`". This way of expressing roles works best with the XACML way of expressing policies. This method of identifying roles is also most conducive to interoperability.

Alternatively, in other application environments, there may be a number of different attribute identifiers, each indicating a different role. For example, in this second type of environment, there might be three attribute identifiers: "`urn:someapp:attributes:officer-role`", "`urn:someapp:attributes:manager-role`", and "`urn:someapp:attributes:employee-role`". In this case, the value of the attribute may be empty or it may contain various parameters associated with the role. XACML policies can handle roles expressed in this way, but not as naturally as in the first way.

XACML supports multiple subjects per access request, indicating various entities that may be involved in making the request. For example, there is usually a human user who initiates the request, at least indirectly. There are usually one or more applications or code bases that generate the actual low-level access request on behalf of the user. There is some computing device on which the application or code base is executing, and this device may have an identity such an IP address. XACML identifies each such `Subject` with a `SubjectCategory` xml attribute that indicates the type of subject being described. For example, the human user has a `SubjectCategory` of `&subject-category;access-subject` (this is the default category); the application that generates the access request has a `SubjectCategory` of `&subject-category;codebase` and so on. In this profile, a role attribute may be associated with any of the categories of subjects involved in making an access request.

8.1.3 Policies

In this Recommendation, four types of policies are specified.

1) **Role** `<PolicySet>` or **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. The `<Target>` element of a Role `<PolicySet>` limits the applicability of the `<PolicySet>` to subjects holding the associated role attribute and value. Each Role `<PolicySet>` references a single corresponding Permission `<PolicySet>` but does not contain or reference any other `<Policy>` or `<PolicySet>` elements.

2) **Permission** `<PolicySet>` or **PPS**: A `<PolicySet>` that contains the actual permissions associated with a given role. It contains `<Policy>` elements and `<Rules>` that describe the resources and actions that subjects are permitted to access, along with any further conditions on that access, such as time of day. A given Permission `<PolicySet>` may also contain references to Permission `<PolicySet>`s associated with other roles that are junior to the given role, thereby allowing the given Permission `<PolicySet>` to inherit all permissions associated with the role of the referenced Permission `<PolicySet>`. The `<Target>` element of a Permission `<PolicySet>`, if present, must not limit the subjects to which the `<PolicySet>` is applicable.

3) **Role Assignment** `<Policy>` or `<PolicySet>`: A `<Policy>` or `<PolicySet>` that defines which roles can be enabled or assigned to which subjects. It may also specify restrictions on combinations of roles or total number of roles assigned to or enabled for a given subject. This type of policy is used by a role enablement authority. Use of a Role Assignment `<Policy>` or `<PolicySet>` is optional.

4) **HasPrivilegesOfRole** `<Policy>`: A `<Policy>` in a Permission `<PolicySet>` that supports requests asking whether a subject has the privileges associated with a given role. If this type of request is to be supported, then a `HasPrivilegesOfRole` `<Policy>` must be included in each Permission `<PolicySet>`. Support for this type of `<Policy>`, and thus for requests asking whether a subject has the privileges associated with a given role, is optional.

Permission `<PolicySet>` instances must be stored in the policy repository in such a way that they can never be used as the initial policy for an XACML PDP; Permission `<PolicySet>` instances must be reachable only through the corresponding Role `<PolicySet>`. This is because, in order to support hierarchical roles, a Permission `<PolicySet>` must be applicable to every subject. The Permission `<PolicySet>` depends on its corresponding Role `<PolicySet>` to ensure that only subjects holding the corresponding role attribute will gain access to the permissions in the given Permission `<PolicySet>`.

Use of separate Role `<PolicySet>` and Permission `<PolicySet>` instances allows support for hierarchical RBAC, where a more *senior* role can acquire the permissions of a more *junior* role. A Permission `<PolicySet>` that does not reference other Permission `<PolicySet>` elements could actually be an XACML `<Policy>` rather than a

<PolicySet>. Requiring it to be a <PolicySet>, however, allows its associated role to become part of a role hierarchy at a later time without requiring any change to other policies.

8.1.4 Multi-role permissions

In this profile, it is possible to express policies where a user must hold several roles simultaneously in order to gain access to certain permissions. For example, changing the care instructions for a hospital patient may require that the Subject performing the action have both the physician role and the staff role.

These policies may be expressed using a Role <PolicySet> where the <Target> element requires the Subject to have all necessary role attributes. This is done by using a single <Subject> element containing multiple <SubjectMatch> elements. The associated Permission <PolicySet> should specify the permissions associated with Subjects who simultaneously have all the specified roles enabled.

The Permission <PolicySet> associated with a multi-role policy may reference the Permission <PolicySet> instances associated with other roles, and thus may inherit permissions from other roles. The permissions associated with a given multi-role <PolicySet> may also be inherited by another role if the other role includes a reference to the Permission <PolicySet> associated with the multi-role policy in its own Permission <PolicySet>.

8.2 RBAC example

This clause is informative.

This clause presents a complete example of the types of policies associated with role based access control.

Assume an organization uses two roles, manager and employee. In this example, they are expressed as two separate values for a single XACML Attribute with AttributeId "&role;". The &role; Attribute values corresponding to the two roles are "&roles;employee" and "&roles;manager". An employee has permission to create a purchase order. A manager has permission to sign a purchase order, plus any permissions associated with the employee role. The manager role therefore is senior to the employee role, and the employee role is junior to the manager role.

According to this profile, there will be two Permission <PolicySet> instances: one for the manager role and one for the employee role. The manager Permission <PolicySet> will give any Subject the specific permission to sign a purchase order and will reference the employee Permission <PolicySet> in order to inherit its permissions. The employee Permission <PolicySet> will give any Subject the permission to create a purchase order.

According to this profile, there will also be two Role <PolicySet> instances: one for the manager role and one for the employee role. The manager Role <PolicySet> will contain a <Target> requiring that the Subject hold a &role; attribute with a value of "&roles;manager". It will reference the manager Permission <PolicySet>. The employee Role <PolicySet> will contain a <Target> requiring that the Subject hold a &role; attribute with a value of "&roles;employee". It will reference the employee Permission <PolicySet>.

8.2.1 Permission <PolicySet> for the manager role

The following Permission <PolicySet> contains the permissions associated with the manager role. The PDP's policy retrieval must be set up such that access to this <PolicySet> is gained only by reference from the manager Role <PolicySet> (see Table 8-1).

Table 8-1/X.1142 – Permission <PolicySet> for managers

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
PolicySetId="PPS:manager:role"
PolicyCombiningAlgId="&policy-combine;permit-overrides">

<!-- Permissions specifically for the manager role -->
<Policy PolicyId="Permissions:specifically:for:the:manager:role"
    RuleCombiningAlgId="&rule-combine;permit-overrides">
    <!-- Permission to sign a purchase order -->
    <Rule RuleId="Permission:to:sign:a:purchase:order" Effect="Permit">
                <Target>
                      <Resources>
                            <Resource>
                                  <ResourceMatch MatchId="&function;string-equal">
                                     <AttributeValue
                                         DataType="&xml;string">purchase
order</AttributeValue>
                                  <ResourceAttributeDesignator
AttributeId="&resource;resource-id"
                                         DataType="&xml;string"/>
                            </ResourceMatch>
                            </Resource>
                      </Resources>
                      <Actions>
                            <Action>
                                  <ActionMatch MatchId="&function;string-equal">
                                    <AttributeValue
                                       DataType="&xml;string">sign</AttributeValue>
                                       ActionAttributeDesignator
                                       AttributeId="&action;action-id"
                                       DataType="&xml;string"/>
                                  </ActionMatch>
                            </Action>
                      </Actions>
                </Target>
          </Rule>
</Policy>
<!-- Include permissions associated with employee role -->
<PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
</PolicySet>
```

8.2.2 Permission <PolicySet> for employee role

The following Permission `<PolicySet>` contains the permissions associated with the employee role (see Table 8-2). The PDP's policy retrieval must be set up such that access to this `<PolicySet>` is gained only by reference from the employee Role `<PolicySet>` or by reference from the more senior manager Role `<PolicySet>` via the manager Permission `<PolicySet>`.

Table 8-2/X.1142 – Permission <PolicySet> for employees

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
   PolicySetId="PPS:employee:role"
   PolicyCombiningAlgId="&policy-combine;permit-overrides">
   <!-- Permissions specifically for the employee role -->
   <Policy PolicyId="Permissions:specifically:for:the:employee:role"
          RuleCombiningAlgId="&rule-combine;permit-overrides">
   <!-- Permission to create a purchase order -->
          <Rule RuleId="Permission:to:create:a:purchase:order" Effect="Permit">
                 <Target>
                        <Resources>
                               <Resource>
                                      <ResourceMatch MatchId="&function;string-equal">
                                        <AttributeValue
                                        DataType="&xml;string">purchase
order</AttributeValue>
                                        <ResourceAttributeDesignator
AttributeId="&resource;resource-id"
                                        DataType="&xml;string"/>
                                      </ResourceMatch>
                               </Resource>
                        </Resources>
                        <Actions>
                               <Action>
                                      <ActionMatch MatchId="&function;string-equal">
                                        <AttributeValue
DataType="&xml;string">create</AttributeValue>
                                        <ActionAttributeDesignator
AttributeId="&action;action-id"
                                        DataType="&xml;string"/>
                                      </ActionMatch>
                               </Action>
                        </Actions>
                 </Target>
          </Rule>
   </Policy>
</PolicySet>
```

8.2.3 Role <PolicySet> for the manager role

The following Role `<PolicySet>` is applicable, according to its `<Target>`, only to `Subjects` who hold a `&role;` (see Table 8-3); Attribute with a value of "`&roles;manager`". The `<PolicySetIdReference>` points to the Permission `<PolicySet>` associated with the manager role.

Table 8-3/X.1142 – Role <PolicySet> for managers

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
   PolicySetId="RPS:manager:role"
   PolicyCombiningAlgId="&policy-combine;permit-overrides">
   <Target>
          <Subjects>
                 <Subject>
                        <SubjectMatch MatchId="&function;anyURI-equal">
                               <AttributeValue
DataType="&xml;anyURI">&roles;manager</AttributeValue>
                               <SubjectAttributeDesignator AttributeId="&role;"
                                      DataType="&xml;anyURI"/>
                        </SubjectMatch>
                 </Subject>
          </Subjects>
   </Target>
<!-- Use permissions associated with the manager role -->
   <PolicySetIdReference>PPS:manager:role</PolicySetIdReference>
</PolicySet>
```

8.2.4 Role <PolicySet> for employee role

The following Role <PolicySet> is applicable, according to its <Target>, only to Subjects who hold a &role; (see Table 8-4); Attribute with a value of "&roles;employee". The <PolicySetIdReference> points to the Permission <PolicySet> associated with the employee role.

Table 8-4/X.1142 – Role <PolicySet> for employees

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
   PolicySetId="RPS:employee:role"
   PolicyCombiningAlgId="&policy-combine;permit-overrides">
   <Target>
          <Subjects>
                <Subject>
                      <SubjectMatch MatchId="&function;anyURI-equal">
                            <AttributeValue
DataType="&xml;anyURI">&roles;employee</AttributeValue>
                            <SubjectAttributeDesignator AttributeId="&role;"
DataType="&xml;anyURI"/>
                      </SubjectMatch>
                </Subject>
          </Subjects>
   </Target>
<!-- Use permissions associated with the employee role -->
<PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
</PolicySet>
```

8.2.5 HasPrivilegesOfRole policies and requests

An XACML RBAC system may choose to support queries of the form "Does this subject have the privileges of role X?" If so, each Permission <PolicySet> must contain a HasPrivilegesOfRole <Policy>. For the Permission <PolicySet> for managers, the HasPrivilegesOfRole <Policy> would look as in Table 8-5.

Table 8-5/X.1142 – Permission <PolicySet> for managers

```
<!-- HasPrivilegesOfRole Policy for manager role -->
<Policy PolicyId="Permission:to:have:manager:role:permissions"
   RuleCombiningAlgId="&rule-combine;permit-overrides">
<!-- Permission to have manager role permissions -->
   <Rule RuleId="Permission:to:have:manager:permissions" Effect="Permit">
          <Condition>
                <Apply FunctionId="&function;and">
                      <Apply FunctionId="&function;anyURI-is-in">
                            <AttributeValue
DataType="&xml;anyURI">&roles;manager</AttributeValue>
                            <ResourceAttributeDesignator AttributeId="&role;"
DataType="&xml;anyURI"/>
                      </Apply>
                            <Apply FunctionId="&function;anyURI-is-in">
                            <AttributeValue

   DataType="&xml;anyURI">&actions;hasPrivilegesofRole</AttributeValue>
                            <ActionAttributeDesignator AttributeId="&action;action-
id"
                                  DataType="&xml;anyURI"/>
                      </Apply>
                </Apply>
          </Condition>
   </Rule>
</Policy>
```

For the Permission <PolicySet> for employees, the HasPrivilegesOfRole <Policy> would look as in Table 8-6.

Table 8-6/X.1142 – Permission <PolicySet> for employees

```
<!-- HasPrivilegesOfRole Policy for employee role -->
<Policy PolicyId="Permission:to:have:employee:role:permissions"
   RuleCombiningAlgId="&rule-combine;permit-overrides">
<!-- Permission to have employee role permissions -->
   <Rule RuleId="Permission:to:have:employee:permissions" Effect="Permit">
          <Condition>
                 <Apply FunctionId="&function;and">
                        <Apply FunctionId="&function;anyURI-is-in">
                               <AttributeValue
DataType="&xml;anyURI">&roles;employee</AttributeValue>
                               <ResourceAttributeDesignator AttributeId="&role;"
                                      DataType="&xml;anyURI"/>
                        </Apply>
                        <Apply FunctionId="&function;anyURI-is-in">
                        <AttributeValue
DataType="&xml;anyURI">&actions;hasPrivilegesofRole
                        </AttributeValue>
                        <ActionAttributeDesignator AttributeId="&action;action-id"
                               DataType="&xml;anyURI"/>
                        </Apply>
                 </Apply>
          </Condition>
   </Rule>
</Policy>
```

A Request asking whether subject `Anne` has the privileges associated with `&roles;manager` would look as in Table 8-7.

Table 8-7/X.1142 – Request about a subject

```
<Request>
   <Subject>
          <Attribute AttributeId="&subject;subject-id" DataType="&xml;string">
                 <AttributeValue>Anne</AttributeValue>
          </Attribute>
   </Subject>
   <Resource>
          <Attribute AttributeId="&role;" DataType="&xml;anyURI">
                 <AttributeValue>&roles;manager</AttributeValue>
                 </Attribute>
   </Resource>
   <Action>
          <Attribute AttributeId="&action;action-id"

   DataType="&xml;anyURI">&actions;hasPrivilegesOfRole</AttributeValue>
          </Attribute>
   </Action>
</Request>
```

Either the `<Request>` must contain `Anne's` direct roles (in this case, `&roles;employee`), or else the PDP's context handler must be able to discover them. `HasPrivilegesOfRole` Policies do not do the job of associating roles with subjects.

8.3 Assigning and enabling role attributes

This clause is informative.

The assignment of various role attributes to users and the enabling of those attributes within a session are outside the scope of the XACML PDP. There must be one or more separate entities, referred to a role enablement authorities, implemented to perform these functions. This profile assumes that the presence in the XACML Request context of a role attribute for a given user (`Subject`) is a valid assignment at the time the access decision is requested.

So where do a subject's role attributes come from? What does one of these role enablement authorities look like? The answer is implementation dependent, but some possibilities can be suggested.

In some cases, role attributes might come from an identity management service that maintains information about a user, including the subject's assigned or allowed roles; the identity management service acts as the role enablement authority. This service might store static role attributes in an LDAP directory, and a PDP's context handler might retrieve them

from there. Or this service might respond to requests for a subject's role attributes from a PDP's context handler, where the requests are in the form of SAML attribute queries.

Role enablement authorities may use an XACML role assignment <Policy> or <PolicySet> to determine whether a subject is allowed to have a particular role attribute and value enabled. A Role Assignment <Policy> or <PolicySet> answers the question "Is subject X allowed to have role R_i enabled?" It does not answer the question "Which set of roles is subject X allowed to have enabled?" The role enablement authority must have some way of knowing which role or roles to submit a request for. For example, the role enablement authority might maintain a list of all possible roles, and, when asked for the roles associated with a given subject, make a request against the role assignment policies for each candidate role.

In this profile, role assignment policies are a different set from the Role <PolicySet> and Permission <PolicySet> instances used to determine the access permissions associated with each role. Role assignment policies are to be used only when the XACML Request comes from a role enablement authority. This separation may be managed in various ways, such as by using different PDPs with different policy stores or requiring <Request> elements for role enablement queries to include a <Subject> with a SubjectCategory of "&subject-category;role-enablement-authority".

There is no fixed form for a Role Assignment <Policy>. The following example (Table 8-8) illustrates one possible form. It contains two XACML <Rule> elements. The first <Rule> states that Anne and Seth and Yassir are allowed to have the "&roles;employee" role enabled between the hours of 9am and 5pm. The second <Rule> states that Steve is allowed to have the "&roles;manager" role enabled, with no restrictions on time of day.

Table 8-8/X.1142 – Role assignment example

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    PolicyId="Role:Assignment:Policy"
    RuleCombiningAlgId="&rule-combine;permit-overrides">
<!-- Employee role requirements rule -->
    <Rule RuleId="employee:role:requirements" Effect="Permit">
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch MatchId="&function;string-equal">
                        <AttributeValue DataType="&xml;string">Seth</AttributeValue>
                        <SubjectAttributeDesignator AttributeId="&subject;subject-id"
                            DataType="&xml;string"/>
                    </SubjectMatch>
                </Subject>
                <Subject>
                    <SubjectMatch MatchId="&function;string-equal">
                        <AttributeValue DataType="&xml;string">Anne</AttributeValue>
                        <SubjectAttributeDesignator AttributeId="&subject;subject-id"
                            DataType="&xml;string"/>
                    </SubjectMatch>
                </Subject>
            </Subjects>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="&function;anyURI-equal">
                    <AttributeValue
DataType="&xml;anyURI">&roles;employee</AttributeValue>
                        <ResourceAttributeDesignator AttributeId="&role;"
                            DataType="&xml;anyURI"/>
                    </ResourceMatch>
                    </Resource>
                </Resources>
                <Actions>
                <Action>
                    <ActionMatch MatchId="&function;anyURI-equal">
                        <AttributeValue DataType="&xml;anyURI">&actions;
                            enableRole</AttributeValue>
                        <ActionAttributeDesignator AttributeId="&action;action-id"
                            DataType="&xml;anyURI"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
```

Table 8-8/X.1142 – Role assignment example

```
        <Condition>
            <Apply FunctionId="&function;and">
                <Apply FunctionId="&function;time-greater-than-or-equal">
                    <Apply FunctionId="&function;time-one-and-only">
                        <EnvironmentAttributeDesignator
AttributeId="&environment;current-time"
                            DataType="&xml;time"/>
                    </Apply>
                    <AttributeValue DataType="&xml;time">9h</AttributeValue>
                </Apply>
                <Apply FunctionId="&function;time-less-than-or-equal">
                    <Apply FunctionId="&function;time-one-and-only">
                        <EnvironmentAttributeDesignator
                         AttributeId="&environment;current-time"
DataType="&xml;time"/>
                    </Apply>
                    <AttributeValueDataType="&xml;time">17h</AttributeValue>
                </Apply>
        </Apply>
    </Condition>
</Rule>
<!-- Manager role requirements rule -->
    <Rule RuleId="manager:role:requirements" Effect="Permit">
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch MatchId="&function;string-equal">
                        <AttributeValue DataType="&xml;string">Steve</AttributeValue>
                        <SubjectAttributeDesignator AttributeId="&subject;subject-id"
                            DataType="&xml;string"/>
                    </SubjectMatch>
                </Subject>
            </Subjects>
            <Resources>
                <Resource>
                    <ResourceMatch MatchId="&function;anyURI-equal">
                        <AttributeValue
                            DataType="&xml;anyURI">&roles;:manager</AttributeValue>
                        <ResourceAttributeDesignator AttributeId="&role;"
                            DataType="&xml;anyURI"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
        <Actions>
            <Action>
                <ActionMatch MatchId="&function;anyURI-equal">
                    <AttributeValue
                        DataType="&xml;anyURI">&actions;enableRole</AttributeValue>
                        <ActionAttributeDesignator AttributeId="&action;action-id"
                            DataType="&xml;anyURI"/>
                </ActionMatch>
            </Action>
        </Actions>
    </Target>
</Rule>
</Policy>
```

8.4 Implementing the RBAC model

This clause is informative.

The following clauses describe how to use XACML policies to implement various components of the RBAC model (see [RBAC]).

8.4.1 Core RBAC

Core RBAC includes the following five basic data elements:

- Users are implemented using XACML `Subjects`. Any of the XACML `SubjectCategory` values may be used, as appropriate.
- Roles are expressed using one or more XACML subject attributes. The set of roles is very application- and policy domain-specific, and it is very important that different uses of roles not be confused. For these reasons, this profile does not attempt to define any standard set of role values, although this profile does recommend use of a common `AttributeId` value of "`urn:oasis:names:tc:xacml:2.0:subject:role`". It is recommended that each application or policy domain agree on and publish a unique set of `AttributeId` values, `DataType` values, and `<AttributeValue>` values that will be used for the various roles relevant to that domain.
- Objects are expressed using XACML `Resources`.
- Operations are expressed using XACML `Actions`.
- Permissions are expressed using XACML Role `<PolicySet>` and Permission `<PolicySet>` instances as described in previous clauses.

Core RBAC requires support for multiple users per role, multiple roles per user, multiple permissions per role, and multiple roles per permission. Each of these requirements can be satisfied by XACML policies based on this profile as follows. Note, however, that the actual assignment of roles to users is outside the scope of the XACML PDP.

XACML allows multiple subjects to be associated with a given role attribute. XACML Role `<PolicySet>`s defined in terms of possession of a particular role `<Attribute>` and `<AttributeValue>` will apply to any requesting user for which that role `<Attribute>` and `<AttributeValue>` are in the XACML Request context.

XACML allows multiple role attributes or role attribute values to be associated with a given `Subject`. If a `Subject` has multiple roles enabled, then any Role `<PolicySet>` instance applying to any of those roles may be evaluated, and the permissions in the corresponding Permission `<PolicySet>` will be permitted. It is even possible to define policies that require a given `Subject` to have multiple role attributes or values enabled at the same time. In this case, the permissions associated with the multiple-role requirement will apply only to a `Subject` having all the necessary role attributes and values at the time an XACML Request context is presented to the PDP for evaluation.

The Permission `<PolicySet>` associated with a given role may allow access to multiple resources using multiple actions. XACML has a rich set of constructs for composing permissions, so there are multiple ways in which multi-permission roles may be expressed. Any Role A may be associated with a Permission `<PolicySet>` B by including a `<PolicySetIdReference>` to Permission `<PolicySet>` B in the Permission `<PolicySet>` associated with the Role A. In this way, the same set of permissions may be associated with more than one role.

In addition to the basic core RBAC requirements, XACML policies using this profile can also express arbitrary conditions on the application of particular permissions associated with a role. Such conditions might include limiting the permissions to a given time period during the day, or limiting the permissions to role holders who also possess some other attribute, whether it is a role attribute or not.

8.4.2 Hierarchical RBAC

Hierarchical RBAC expands core RBAC with the ability to define inheritance relations between roles. For example, role A may be defined to inherit all permissions associated with role B. In this case, role A is considered to be senior to role B in the role hierarchy. If role B in turn inherits permissions associated with role C, then role A will also inherit those permissions by virtue of being senior to role B.

XACML policies using this profile can implement role inheritance by including a `<PolicySetIdReference>` to the Permission `<PolicySet>` associated with one role inside the Permission `<PolicySet>` associated with another role. The role that includes the `<PolicySetIdReference>` will then inherit the permissions associated with the referenced role.

This profile structures policies in such a way that inheritance properties may be added to a role at any time without requiring changes to `<PolicySet>` instances associated with any other roles. An organization may not initially use role hierarchies, but may later decide to make use of this functionality without having to rewrite existing policies.

8.5 Profile

This clause discusses roles, role attributes, role assignment and access control.

8.5.1 Roles and role attributes

Roles shall be expressed using one or more XACML Attributes. Each application domain using this profile for role based access control shall define or agree upon one or more AttributeId values to be used for role attributes. Each such AttributeId value shall be associated with a set of permitted values and their DataTypes. Each permitted value for such an AttributeId shall have well-defined semantics for the use of the corresponding value in policies.

This profile recommends use of the "urn:oasis:names:tc:xacml:2.0:subject:role" AttributeId value for all role attributes. Instances of this attribute should have a DataType of "http://www.w3.org/2001/XMLSchema#anyURI".

8.5.2 Role assignment or enablement

A Role Enablement Authority, responsible for assigning roles to users and for enabling roles for use within a user's session, may use an XACML Role Assignment <Policy> or <PolicySet> to determine which users are allowed to enable which roles and under which conditions. There is no prescribed form for a Role Assignment <Policy> or <PolicySet>. It is recommended that roles in a Role Assignment <Policy> or <PolicySet> be expressed as Resource Attributes, where the AttributeId is &role; and the <AttributeValue> is the URI for the relevant role value. It is recommended that the action of assigning or enabling a role be expressed as an Action Attribute, where the AttributeId is &action;action-id, the DataType is &xml;anyURI, and the <AttributeValue> is &actions;enableRole.

8.5.3 Access control

Role based access control shall be implemented using two types of <PolicySet>s: Role <PolicySet>, Permission <PolicySet>. The specific functions and requirements of these two types of <PolicySet>s are as follows.

For each role, one Role <PolicySet> shall be defined. Such a <PolicySet> shall contain a <Target> element that makes the <PolicySet> applicable only to subjects having the XACML attribute associated with the given role; the <Target> element shall not restrict the Resource, Action, or Environment. Each Role <PolicySet> shall contain a single <PolicySetIdReference> element that references the unique Permission <PolicySet> associated with the role. The Role <PolicySet> shall not contain any other <Policy>, <PolicySet>, <PolicyIdReference>, or <PolicySetIdReference> elements.

For each role, one Permission <PolicySet> shall be defined. Such a <PolicySet> shall contain <Policy> and <Rule> elements that specify the types of access permitted to Subjects having the given role. The <Target> of the <PolicySet> and its included or referenced <PolicySet>, <Policy>, and <Rule> elements shall not limit the Subjects to which the Permission <PolicySet> is applicable.

If a given role inherits permissions from one or more junior roles, then the Permission <PolicySet> for the given (senior) role shall include a <PolicySetIdReference> element for each junior role. Each such <PolicySetIdReference> shall reference the Permission <PolicySet> associated with the junior role from which the senior role inherits.

A Permission <PolicySet> may include a HasPrivilegesOfRole <Policy>. Such a <Policy> shall have a <Rule> element with an effect of "Permit". This rule shall permit any subject to perform an Action with an attribute having an AttributeId of &action;action-id, a DataType of &xml;anyURI, and an <AttributeValue> having a value of &actions;hasPrivilegesOfRole on a resource having an attribute that is the role to which the Permission <PolicySet> applies (for example, an AttributeId of &role;, a DataType of &xml;anyURI, and an <AttributeValue> whose value is the URI of the specific role value). Note that the role attribute, which is a subject attribute in a Role <PolicySet> <Target>, is treated as a resource attribute in a HasPrivilegesOfRole <Policy>.

The organization of any repository used for policies and the configuration of the PDP shall ensure that the PDP can never use a Permission <PolicySet> as the PDP's initial policy.

8.6 Identifiers

This profile defines the following URN identifiers.

8.6.1 Profile identifier

The following identifier shall be used as the identifier for this profile when an identifier in the form of a URI is required.

```
urn:oasis:names:tc:xacml:2.0:profiles:rbac:core-hierarchical
```

8.6.2 Role attribute

The following identifier may be used as the `AttributeId` for role attributes.

```
urn:oasis:names:tc:xacml:2.0:subject:role
```

8.6.3 SubjectCategory

The following identifier may be used as the `SubjectCategory` for subject attributes identifying that a request is coming from a role enablement authority.

```
urn:oasis:names:tc:xacml:2.0:subject-category:role-enablement-authority
```

8.6.4 Action attribute values

The following identifier may be used as the `<AttributeValue>` of the `&action;action-id` attribute in a `HasPrivilegesOfRole` `<Policy>`.

```
urn:oasis:names:tc:xacml:2.0:actions:hasPrivilegesOfRole
```

The following identifier may be used as the `<AttributeValue>` of the `&action;action-id` attribute in a Role Assignment `<Policy>`.

```
urn:oasis:names:tc:xacml:2.0:actions:enableRole
```

9 Multiple resource profile of XACML

The policy evaluation performed by an XACML policy decision point, or PDP, is defined in terms of a single requested resource, with the authorization decision contained in a single `<Result>` element of the response context. A Policy enforcement point, or PEP, however, may wish to submit a single request context for access to multiple resources, and may wish to obtain a single response context that contains a separate authorization decision (`<Result>` element) for each requested resource. Such a request context might be used to avoid sending multiple decision request messages between a PEP and PDP, for example. Alternatively, a PEP may wish to submit a single request context for all the nodes in a hierarchy, and may wish to obtain a single authorization decision (`<Result>` element) that indicates whether access is permitted to all of the requested nodes. Such a request context might be used when the requester wants access to an entire XML document, to an entire sub-tree of elements in such a document, or to an entire file system directory with all its subdirectories and files, for example.

This Recommendation describes three ways in which a PEP can request authorization decisions for multiple resources in a single request context, and how the result of each such authorization decision is represented in the single response context that is returned to the PEP.

This Recommendation also describes two ways in which a PEP can request a single authorization decision in response to a request for all the nodes in a hierarchy.

Support for each of the mechanisms described in this profile is optional for compliant XACML implementations.

Commonly used resource attributes are abbreviated as follows:

– "`resource-id`" attribute: A resource attribute with an `AttributeId` of:

  `"urn:oasis:names:tc:xacml:1.0:resource:resource-id"`

– "`scope`" attribute: A resource attribute with an `AttributeId` of:

  `"urn:oasis:names:tc:xacml:2.0:resource:scope"`

  See 9.3 for more information about this attribute.

9.1 Requests for multiple resources

This clause is normative, but it is optional.

A single XACML Request context may represent a request for access to multiple resources, with a separate authorization decision desired for each resource. The syntax and semantics of such requests and responses are specified in this clause.

The <Result> elements produced by evaluating a request for access to multiple resources shall be identical to those that would be produced from a series of requests, each requesting access to exactly one of the resources. Each such resource is called an individual resource. The conceptual request context that corresponds to each <Result> element is called an individual resource request. The ResourceId value in the <Result> element shall be the <AttributeValue> of the "resource-id" attribute in the corresponding individual resource request. This mapping of an original request context containing multiple authorization decision requests to Individual resource requests, and the corresponding mapping of multiple authorization decisions to multiple <Result> elements in a single response context may be performed by the context handler in this Recommendation. This profile does not require that the implementation of the evaluation of a request for access to multiple resources conform to the preceding model or that actual Individual resource requests be constructed. The profile requires only that the <Result> elements shall be the same as if the preceding model were used.

Three ways of specifying requests for access to multiple resources are described in the following clauses. Each way of specifying requests describes the individual resource requests that correspond to the <Result> elements in the response context.

A single XACML Request context submitted by a PEP may use more than one of these ways of requesting access to multiple resources in different <Resource> elements.

9.1.1 Nodes identified by "scope"

This clause is normative, but it is optional.

This text describes the use of two values for the "scope" resource attribute to specify a request for access to multiple resources in a hierarchy. This syntax may be used with any hierarchical resource, regardless of whether it is an XML document or not.

9.1.1.1 Profile URI

The following URIs shall be used as URI identifiers for the functionality specified in this part of this profile. The first identifier shall be used when the functionality is supported for XML resources, and the second identifier shall be used when the functionality is supported for resources that are not XML documents:

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:scope:xml
urn:oasis:names:tc:xacml:2.0:profile:multiple:scope:non-xml
```

9.1.1.2 Original request context syntax

The original XACML Request context <Resource> element shall contain a "scope" attribute with a value of either "Children", or "Descendants". If the requested resources are in an XML document, then the <ResourceContent> element shall be present and shall contain the entire XML document of which the requested elements are a part. Also, if the requested resources are in an XML document, then the XPath expression used as the value of the "resource-id" attribute shall evaluate to a nodeset containing exactly one node.

9.1.1.3 Semantics

Such a request context shall be interpreted as a request for access to a set of nodes in a hierarchy relative to the single node specified in the "resource-id" attribute. If the value of the "scope" attribute is "Children", each individual resource is the one node indicated by the "resource-id" attribute (or attributes, where the single resource has multiple normative identifiers) and all of its immediate child nodes. If the value of the "scope" attribute is "Descendants", the individual resource is the one node indicated by the "resource-id" attribute and all of its descendant nodes.

Each individual resource request shall be identical to the original request context with two exceptions: the "scope" attribute shall not be present and the <Resource> element shall represent a single individual resource. This <Resource> element shall contain at least one "resource-id" attribute, and all values for such attributes shall be unique, normative identities of the individual resource. If the "resource-id" attribute in the original request context contained an issuer, the "resource-id" attributes in the individual resource request shall contain the same issuer. If a <ResourceContent> element was present in the original request context, then that same <ResourceContent> element shall be included in each individual resource request.

Neither XACML nor this profile specifies how the context handler obtains the information required to determine which nodes are children or descendants of a given node, except in the case of an XML document, where the information shall be obtained from the <ResourceContent> element.

9.1.2 Nodes identified by XPath

This clause is normative, but it is optional.

This clause describes the use of an XPath expression in the "resource-id" attribute, together with an "XPath-expression" value in the "scope" attribute to specify a request for access to multiple nodes in an XML document. This syntax shall be used only with resources that are XML documents.

9.1.2.1 Profile URI

The following URI shall be used as the URI identifier for the functionality specified in this part of this profile:

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:xpath-expression
```

9.1.2.2 Original request context

The original XACML Request context <Resource> element shall contain a <ResourceContent> element and a "resource-id" attribute with a DataType of "urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression" such that the <AttributeValue> of the "resource-id" attribute is an XPath expression that evaluates to a nodeset that represents multiple nodes in the <ResourceContent> element. The <Resource> element shall contain a "scope" attribute with a value of "XPath-expression".

9.1.2.3 Semantics

Such a request context shall be interpreted as a request for access to the multiple nodes in the nodeset represented by the <AttributeValue> of the "resource-id" attribute. Each such node shall represent an individual resource.

Each individual resource request shall be identical to the original request context with two exceptions: the "scope" attribute shall not be present and the "resource-id" attribute value shall be an XPath expression that evaluates to a single node in the <ResourceContent> element. That node shall be the individual resource. If the "resource-id" attribute in the original request context contained an issuer, the "resource-id" attribute in the individual resource request shall contain the same issuer.

9.1.3 Multiple <Resource> elements

This clause is normative, but it is optional.

This clause describes the use of multiple <Resource> elements in a request context to specify a request for access to multiple resources. This syntax may be used with any resource or resources, regardless of whether they are XML documents or not and regardless of whether they are hierarchical resources or not.

9.1.3.1 Profile URI

The following URI shall be used as the URI identifier for the functionality specified in this part of this profile:

```
    urn:oasis:names:tc:xacml:2.0:profile:multiple:multiple-resource-
elements
```

9.1.3.2 Original request context

The XACML Request context shall contain multiple <Resource> elements.

9.1.3.3 Semantics

Such a request context shall be interpreted as a request for access to all resources specified in the individual <Resource> elements. Each <Resource> element shall represent one individual resource unless that element utilizes the other mechanisms described in this profile.

For each <Resource> element, one individual resource request shall be created. This individual resource request shall be identical to the original request context with one exception: only the one <Resource> element shall be present. If such a <Resource> element contains a "scope" attribute having any value other than "Immediate", then the individual resource request shall be further processed according to the corresponding part of this profile. This processing may involve decomposing the one individual resource request into other individual resource requests before evaluation by the PDP.

9.2 Requests for an entire hierarchy

This clause is normative, but it is optional.

In some cases, a resource is hierarchical, but the authorization decision request is intended to request access to all the nodes within that resource or to an entire sub-hierarchy of nodes within that resource. This might be the case when access to an XML document is being requested for purposes of making a copy of the entire document, or where access to an entire file system directory with all its subdirectories and files is being requested. A single <Result> is desired, indicating whether the requester is permitted to access the entire set of nodes.

The <Result> element produced by evaluating such a request for access shall be identical to that produced by the following process. A series of request contexts is evaluated, each requesting access to exactly one node of the hierarchy. The <Decision> in the single <Result> that is returned to the PEP shall be "Permit" if and only if all <Result> elements resulting from the evaluation of the individual nodes contained a <Decision> of "Permit". Otherwise, the <Decision> in the single <Result> returned to the PEP shall be "Deny". This profile does not require that the implementation of the evaluation of a request for access to such a hierarchical resource conform to the preceding model or that actual request contexts corresponding to the individual nodes in the hierarchy be constructed. This profile requires only that the <Result> element shall be the same as if the preceding model were used.

Two syntaxes for this functionality are specified in the following clauses, one for use with resources that are XML documents, and the other for use with resources that are not XML documents.

9.2.1 XML resources

This clause is normative, but it is optional.

This clause describes the syntax for requesting access to an entire XML document, or to an element within that document with all its recursive sub-elements.

9.2.1.1 Profile URI

The following URI shall be used as the identifier for the functionality specified in this part of this profile:

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:entire-hierarchy:xml
```

9.2.1.2 Original request context

The <Resource> element in the original request context shall contain a "scope" attribute with a value of "EntireHierarchy".

The <Resource> element in the original request context shall contain a single "resource-id" attribute with a DataType of "urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression", such that the <AttributeValue> evaluates to a nodeset that represents exactly one node in the <ResourceContent> element.

The <Resource> element in the original request context may contain other attributes.

9.2.1.3 Semantics

The <Result> of such a request shall be equivalent to that produced by the following process. For each node in the requested hierarchy, the context handler shall create a new request context. Each such request context shall contain a single <Resource> element having a "resource-id" attribute with a DataType of "urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression" and a value that is an XPath expression that evaluates to a nodeset that contains exactly that one node in the <ResourceContent> element. The context handler shall submit each such new request context to the PDP for evaluation and shall keep track of the <Decision> in the corresponding <Result> elements. If, and only if, all the new request contexts evaluate to "Permit", then a single <Result> containing a <Decision> of "Permit" shall be placed into the response context returned to the PEP. If any of the new request contexts evaluates to "Deny", "Indeterminate", or "NotApplicable", then a single <Result> containing a <Decision> of "Deny" shall be placed into the response context returned to the PEP.

9.2.2 Non-XML resources

This clause is normative, but it is optional.

This clause describes the syntax for requesting access to an entire hierarchy of nodes within a hierarchical resource that is not an XML document.

9.2.2.1 Profile URI

The following URI shall be used as the identifier for the functionality specified in this part of this profile:

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:entire-hierarchy:non-xml
```

9.2.2.2 Original request context

The <Resource> element in the original request context shall contain a "scope" attribute with a value of "EntireHierarchy".

The <Resource> element in the original request context shall contain a single "resource-id" attribute that represents a single node in a hierarchical resource.

The <Resource> element in the original request context may contain other attributes.

The representation of nodes in a hierarchical resource specified in the XACML profile for hierarchical resources in this Recommendation may be used to represent the identity of the single node.

9.2.2.3 Semantics

The <Result> of such a request shall be equivalent to that produced by the following process. For each node in the requested hierarchy, the context handler shall create a new request context. Each such request context shall contain a single <Resource> element having a "resource-id" attribute with a value that is the identity of that one node in the hierarchy. The context handler shall submit each such new request context to the PDP for evaluation and shall keep track of the <Decision> in the corresponding <Result> elements. If, and only if, all the new request contexts evaluate to "Permit", then a single <Result> containing a <Decision> of "Permit" shall be placed into the response context returned to the PEP. If any of the new request contexts evaluates to "Deny", "Indeterminate", or "NotApplicable", then a single <Result> containing a <Decision> of "Deny" shall be placed into the response context returned to the PEP.

Neither XACML nor this profile specifies how the context handler obtains the information required to determine which nodes are descendants of the originally specified node, or how to represent the identity of each such node. The representation of nodes in a hierarchical resource specified in the XACML profile for hierarchical resources in this Recommendation may be used to represent the identity of each such node.

9.3 New attribute identifiers

The following identifier is used as the AttributeId of a resource attribute that indicates the scope ("scope" attribute) of a request for access in a single <Resource> element of a request context.

```
urn:oasis:names:tc:xacml:2.0:resource:scope
```

The attribute shall have a DataType of "http://www.w3.org/2001/XMLSchema#string".

The valid values for this attribute are listed below and in 7.5. An implementation may support any subset of these values, including the empty set.

- "Immediate" – The <Resource> element refers to a single non-hierarchical resource or to a single node in a hierarchical resource. This is the default value, if no "scope" attribute is present. The <Resource> element shall be processed according to clause 7.
- "Children" – The <Resource> element refers to multiple resources in a hierarchy. The set of resources consists of a single node described by the "resource-id" resource attribute and of all that node's immediate children in the hierarchy. The <Resource> element shall be processed according to 9.1.1 of this profile.
- "Descendants" – The <Resource> element refers to multiple resources in a hierarchy. The set of resources consists of a single node described by the "resource-id" resource attribute and of all that node's descendants in the hierarchy. The <Resource> element shall be processed according to 9.1.1 of this profile.
- "XPath-expression" – The <Resource> element refers to multiple resources. The set of resources consists of the nodes in a nodeset described by the "resource-id" resource attribute. Each of the nodes shall be contained in the <ResourceContent> element of the <Resource> element. The <Resource> element shall be processed according to 9.1.2 of this profile.

– "EntireHierarchy" – The <Resource> element refers to a single resource. The resource consists of a node described by the "resource-id" resource attribute along with all that node's descendants. All of the nodes shall be nodes in an XML document that is contained in the <ResourceContent> element of the <Resource> element. The <Resource> element shall be processed according to 9.2.

9.4 New profile identifiers

The following URI values shall be used as URI identifiers for the functionality specified in various clauses of this profile:

– "scope" attribute of "children" or "descendants" in <Resource>: XML resources

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:scope:xml
```

– "scope" attribute of "children" or "descendants" in <Resource>: Non-XML resources

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:scope:non-xml
```

– XPath expression in "resource-id" attribute

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:xpath-expression
```

– Multiple <Resource> elements

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:multiple-resource-
elements
```

– Requests for an entire hierarchy: XML resources

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:entire-hierarchy:xml
```

– Requests for an entire hierarchy: Non-XML resources

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:entire-hierarchy:non-
xml
```

10 SAML 2.0 profile of XACML

This clause defines a profile on how to use SAML 2.0 (see ITU-T Rec. X.1141) to protect, transport, and request XACML schema instances and other information needed by an XACML implementation.

SAML is an XML-based framework for exchanging security information. This security information is expressed in the form of assertions about subjects, where a subject is an entity (either human or computer) that has an identity in some security domain. A single assertion might contain several different internal statements about authentication, authorization and attributes. SAML defines a protocol by which clients can request assertions from SAML authorities and get a response from them. This protocol, consisting of XML-based request and response message formats, can be bound to many different underlying communications and transport protocols; SAML currently defines one binding to SOAP over HTTP. In creating their responses, SAML authorities can use various sources of information, such as external policy stores and assertions that were received as input in requests. SAML defines assertion elements, subjects, conditions, advice and statements.

There are six types of queries and statements used in this clause:

1) AttributeQuery: A standard SAML request used for requesting one or more attributes from an attribute authority.
2) AttributeStatement: A standard SAML statement that contains one or more attributes. This statement may be used in a SAML response from an attribute authority, or it may be used in a SAML assertion as a format for storing attributes in an attribute repository.
3) XACMLPolicyQuery: A SAML request extension, defined in this profile. It is used for requesting one or more policies from a policy administration point.
4) XACMLPolicyStatement: A SAML statement extension, defined in this profile. It may be used in a SAML response from a policy administration point, or it may be used in a SAML assertion as a format for storing policies in a policy repository.
5) XACMLAuthzDecisionQuery: A SAML request extension, defined in this profile. It is used by a PEP to request an authorization decision from an XACML PDP.

6) `XACMLAuthzDecisionStatement`: A SAML statement extension, defined in this profile. It may be used in a SAML response from an XACML PDP. It might also be used in a SAML assertion that is used as a credential, but this is not part of the currently defined XACML use model.

The following diagram (Figure 10-1) illustrates the XACML use model and the messages that are used to communicate between the various components. Not all components will be used in every implementation.

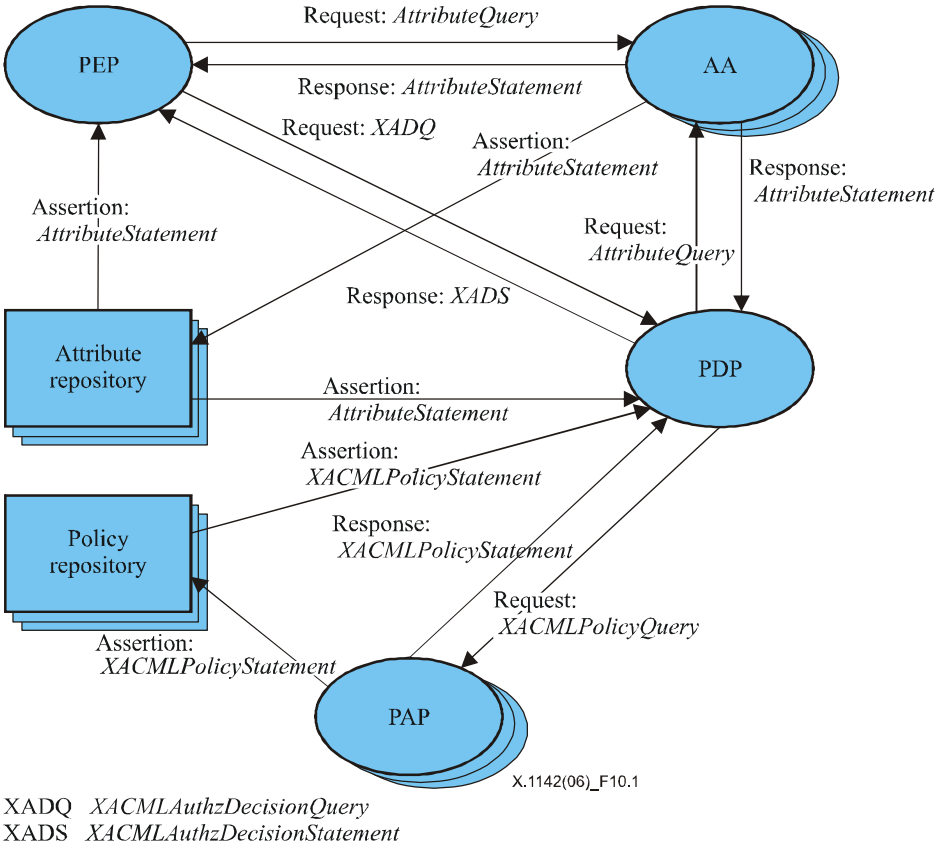

Figure 10-1/X.1142 – XACML use model

This clause describes all these query and statement schema elements, and describes how to use them. It also describes some other aspects of using SAML with XACML. This Recommendation requires no changes or extensions to XACML, but does define extensions to SAML.

In order to improve readability, the examples in this profile assume the use of the following XML internal entity declarations:

```
^lt;!ENTITY saml "urn:oasis:names:tc:SAML:2.0:assertion"
^lt;!ENTITY samlp "urn:oasis:names:tc:SAML:2.0:protocol"
^lt;!ENTITY xacml "urn:oasis:names:tc:xacml:2.0:"
^lt;!ENTITY xacml-context "urn:oasis:names:tc:xacml:2.0:context:schema:os"
^lt;!ENTITY xml "http://www.w3.org/2001/XMLSchema#"
^lt;!ENTITY subject-id "urn:oasis:names:tc:xacml:1.0:subject:subject-id"
^lt;!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource:"
^lt;!ENTITY resource-id "urn:oasis:names:tc:xacml:1.0:resource:resource-id"
^lt;!ENTITY action-id "urn:oasis:names:tc:xacml:1.0:action:action-id"
^lt;!ENTITY environment "urn:oasis:names:tc:xacml:1.0:environment:"
^lt;!ENTITY current-dateTime
    "urn:oasis:names:tc:xacml:1.0:environment:current-dateTime"
```

For example, "`&xml;#string`" is equivalent to http://www.w3.org/2001/XMLSchema#string. The namespace associated with the XACML schema that extends the SAML assertion schema is:

```
xacml-saml="urn:oasis:names:tc:xacml:2.0:saml:assertion:schema:os"
```

The namespace associated with the XACML schema that extends the SAML protocol schema is:

```
xacml-samlp="urn:oasis:names:tc:xacml:2.0:saml:protocol:schema:os"
```

10.1 Mapping SAML and XACML attributes

The SAML assertion schema defines an attribute assertion. The SAML protocol schema defines an `AttributeQuery` used for requesting instances of attribute assertions, and a response that contains the requested instances. Systems using XACML may use instances of these SAML elements transmit and store SAML attributes. Systems using XACML may use the SAML `AttributeQuery` protocol to request instances of SAML attributes. In order to be used in an XACML Request context, the SAML attribute shall be mapped to an XACML attribute.

A SAML attribute assertion is a `<saml:Assertion>` instance that contains one or more `<saml:AttributeStatement>` instances, each of which may contain one or more `<saml:Attribute>` instances.

In order to be used in an XACML Request context, each SAML attribute in the SAML attribute assertion shall comply with *XACML Attribute profile*, namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML`, in ITU-T Rec. X.1141.

An `<xacml-context:Attribute>` shall be constructed from the corresponding `<saml:Attribute>` element in a SAML attribute assertion as follows.

- XACML `AttributeId` XML attribute
  - The fully-qualified value of the `<saml:Attribute>` `Name` XML attribute shall be used.
- XACML `DataType` XML attribute
  - The fully-qualified value of the `<saml:Attribute>` `DataType` XML attribute shall be used. If the `<saml:Attribute>` `DataType` XML attribute is missing, the XACML `DataType` XML attribute shall be http://www.w3.org/2001/XMLSchema#string.
- XACML `Issuer` XML attribute
  - The string value of the `<saml:Issuer>` element from the SAML Attribute Assertion shall be used.
- `<xacml-context:AttributeValue>`
  - The `<saml:AttributeValue>` value shall be used as the value of the `<xacml-context:AttributeValue>` element.

Each `<saml:Attribute>` instance is mapped to a single `<xacml-context:Attribute>` element. Not all `<saml:Attribute>` instances in a SAML attribute assertion need to be mapped; the SAML attribute instances to be mapped may be selected by a mechanism not specified here. The `Issuer` of the `<saml:Assertion>` element is used as the `Issuer` for each `<xacml-context:Attribute>` element that is created.

The `<xacml-context:Attribute>` created from the `<saml:Assertion>` shall be placed into the `<xacml-context:Resource>`, `<xacml-context:Subject>`, `<xacml-context:Action>`, or `<xacml-context:Environment>` element that corresponds to the entity that is the `<saml:Subject>` in the SAML attribute assertion. For example, if the SAML attribute assertion subject contains a `<saml:NameIdentifier>` element, and the value of that `NameIdentifier` matches the value of the `<xacml-context:Attribute>` having an `AttributeId` of `&resource;resource-id`, then `<xacml-context:Attribute>` instances created from `<saml:Attribute>` instances in that SAML attribute assertion shall be placed into the `<xacml-context:Resource>` element. If the `<xacml-context:Attribute>` is placed into an `<xacml-context:Subject>` element, then the XACML `SubjectCategory` XML attribute shall also be consistent with the entity that is the subject of the `<saml:Assertion>`.

The entity performing the mapping shall ensure that the semantics defined by SAML for the elements in the `<saml:Assertion>` has been adhered to. The mapping entity need not perform these semantic checks itself, but it shall ensure that the checks have been done before any `<xacml:Attribute>` created from the `<saml:Assertion>` is used by an XACML PDP. These semantic checks include, but are not limited to, the following.

- Any `NotBefore` and `NotOnOrAfter` XML attributes in the `<saml:Assertion>` shall be valid with respect to the `<xacml:Request>` in which the SAML-derived `<xacml:Attribute>` is used. This means that the `NotBefore` and `NotOnOrAfter` XML attribute values shall be consistent with the `&environment;current-time`, `&environment;current-date`, and `&environment:current-dateTime` `<xacml:Attribute>` values associated with the `<xacml:Request>`.

- The entity doing the mapping shall ensure that the semantics defined by SAML for any <saml:AudienceRestrictionCondition> or <saml:DoNotCacheCondition> elements has been adhered to.
- If a <ds:Signature> element occurs in the <saml:Assertion>, then the entity performing the mapping shall ensure that the signature is valid and that the SAML <Issuer> element is consistent with any <ds:X509IssuerName> value in the signature. The guidelines regarding digital signatures in ITU-T Rec. X.1141 shall be adhered to.

10.2 Authorization decisions

SAML 2.0 defines a rudimentary AuthzDecisionQuery (see ITU-T Rec. X.1141). A SAML AuthzDecisionQuery is unable to convey all the information that an XACML PDP is capable of accepting as part of its request context. Likewise, the SAML AuthzDecisionStatement is unable to convey all the information contained in an XACML response context.

In order to allow a PEP to use the SAML request and response syntax with full support for the XACML Request context and response context syntax, this Recommendation defines two SAML extensions:

- <xacml-samlp:XACMLAuthzDecisionQuery> is a SAML query that extends the SAML protocol schema (see ITU-T Rec. X.1141). It allows a PEP to submit an XACML Request context in a SAML request, along with other information.
- <xacml-saml:XACMLAuthzDecisionStatement> is a SAML statement that extends the SAML assertion schema (see ITU-T Rec. X.1141). It allows an XACML PDP to return an XACML response context in the response to an <XACMLAuthzDecisionStatement>, along with other information. It also allows an XACML response context to be stored or transmitted in the form of a SAML assertion.

10.2.1 Element <XACMLAuthzDecisionQuery>

The <XACMLAuthzDecisionQuery> element may be used by a PEP to request an authorization decision from an XACML PDP. It allows a SAML request to convey an XACML Request context instance.

```
<xs:element name="XACMLAuthzDecisionQuery" type="XACMLAuthzDecisionQueryType"/>
<xs:complexType name="XACMLAuthzDecisionQueryType">
  <xs:complexContent>
   <xs:extension base="samlp:RequestAbstractType">
    <xs:sequence>
        <xs:element ref="xacml-context:Request"/>
    </xs:sequence>
    <xs:attribute name="InputContextOnly" type="boolean" use="optional" default="false"/>
    <xs:attribute name="ReturnContext" type="boolean" use="optional" default="false"/>
   </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The <XACMLAuthzDecisionQuery> element is of **XACMLAuthzDecisionQueryType** complex type. This element is an alternative to the SAML-defined <samlp:AuthzDecisionQuery> that allows a PEP to use the full capabilities of an XACML PDP.

The <XACMLAuthzDecisionQuery> element contains the following XML attributes and elements:

– InputContextOnly [Default "false"]

  This XML attribute governs the sources of information that the PDP is allowed to use in making its authorization decision. If this XML attribute is "true", then the authorization decision shall be made solely on the basis of information contained in the <XACMLAuthzDecisionQuery>; no external attributes may be used. If this XML attribute is "false", then the authorization decision may be made on the basis of external attributes not contained in the <XACMLAuthzDecisionQuery>.

– ReturnContext [Default "false"]

  This XML attribute allows the PEP to request that an <xacml-context:Request> element be included in the <XACMLAuthzDecisionStatement> resulting from the request. It also governs the contents of that <xacml-context:Request> element.

If this XML attribute is "true", then the PDP shall include the <xacml-context:Request> element in the <XACMLAuthzDecisionStatement> element in the <XACMLResponse>. This <xacml-context:Request> element shall include all those attributes supplied by the PEP in the <XACMLAuthzDecisionQuery> that were used in making the authorization decision. The PDP may include additional attributes in this <xacml-context:Request> element, such as external attributes obtained by the PDP and used in making the authorization decision, or other attributes known by the PDP that may be useful to the PEP in making subsequent <XACMLAuthzDecisionQuery> requests.

If this XML attribute is "false", then the PDP shall not include the <xacml-context:Request> element in the <XACMLAuthzDecisionStatement> element of the <XACMLResponse>.

– <xacml-context:Request> [Required]

An XACML Request context.

10.2.2 Element <XACMLAuthzDecisionStatement>

The <XACMLAuthzDecisionStatement> may be used by an XACML PDP to return a SAML response containing an XACML response context to a PEP in response to an <XACMLAuthzDecisionQuery>. It may also be used in a SAML assertion as a format for storage of an authorization decision in a repository.

```
<xs:element name="XACMLAuthzDecisionStatement" type="xacml-
saml:XACMLAuthzDecisionStatementType"/>
<xs:complexType name="XACMLAuthzDecisionStatementType">
  <xs:complexContent>
     <xs:extension base="saml:StatementAbstractType">
        <xs:sequence>
           <xs:element ref="xacml-context:Response"/>
              <xs:element ref="xacml-context:Request" MinOccurs="0"/>
        </xs:sequence>
     </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The <XACMLAuthzDecisionStatement> element is of **XACMLAuthzDecisionStatementType** complex type. This element is an alternative to the SAML-defined <samlp:AuthzDecisionStatement> that allows a SAML assertion to contain the full content of the response from an XACML PDP.

The <XACMLAuthzDecisionStatement> element contains the following elements:

– <xacml-context:Response> [Required]

The XACML response context created by the XACML PDP in response to the <XACMLAuthzDecisionQuery>.

– <xacml-context:Request> [Optional]

An <xacml-context:Request> containing XACML attributes returned by the XACML PDP in response to the <XACMLAuthzDecisionQuery>. This element shall be included if the ReturnResponse XML attribute in the <XACMLAuthzDecisionQuery> is "true". This element shall not be included if the ReturnResponse XML attribute in the <XACMLAuthzDecisionQuery> is "false".

10.3 Policies

XACML defines two policy schema elements: <Policy> and <PolicySet>. SAML does not define any protocol or assertion schemas for policies. This clause defines new SAML extensions for <XACMLPolicyQuery> and <XACMLPolicyStatement> elements. Instances of these new elements can be used to request, transmit, and store XACML <Policy> and <PolicySet> instances.

10.3.1 Element <XACMLPolicyQuery>

The <XACMLPolicyQuery> element is used by a PDP to request one or more XACML policy or PolicySet instances from an on-line policy administration point as part of a SAML request.

```
<xs:element name="XACMLPolicyQuery" type="XACMLPolicyQueryType"/>
<xs:complexType name="XACMLPolicyQueryType">
   <complexContent>
         <xs:extension base="samlp:RequestAbstractType">
```

```
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                      <xs:element ref="xacml-context:Request"/>
                      <xs:element ref="xacml:Target"/>
                      <xs:element ref="xacml:PolicySetIdReference"/>
                      <xs:element ref="xacml:PolicyIdReference"/>
                </xs:choice>
          </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The <XACMLPolicyQuery> element is of **XACMLPolicyQueryType** complex type.

The <XACMLPolicyQuery> element contains one or more of the following elements:

– <xacml-context:Request> [Any Number]

  Supplies an XACML Request context. All XACML policy and PolicySet instances applicable to this request shall be returned.

– <xacml:Target> [Any Number]

  Supplies an XACML <Target> element. All XACML policy and PolicySet instances applicable to this <Target> shall be returned.

– <xacml:PolicySetIdReference> [Any Number]

  Identifies an XACML <PolicySet> to be returned.

– <xacml:PolicyIdReference> [Any Number]

  Identifies an XACML <Policy> to be returned.

10.3.2 Element <XACMLPolicyStatement>

The <XACMLPolicyStatement> is used by a policy administration point to return one or more XACML <Policy> or <PolicySet> instances in a SAML response to an <XACMLPolicyQuery> SAML request. The <XACMLPolicyStatement> may also be used in a SAML assertion as a format for storing the <XACMLPolicyStatement> in a repository.

```
<xs:element name="XACMLPolicyStatement" type="xacml-
saml:XACMLPolicyStatementType"/>
<xs:complexType name="XACMLPolicyStatementType">
   <xs:complexContent>
          <xs:extension base="saml:StatementAbstractType">
                <xs:choice minOccurs="0" maxOccurs=unbounded">
                      <xs:element ref="xacml:Policy"/>
                      <xs:element ref="xacmlPolicySet"/>
                </xs:choice>
                </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The <XACMLPolicyStatement> element is of **XACMLPolicyStatementType** complex type.

The <XACMLPolicyStatement> element contains the following elements. If the <XACMLPolicyStatement> is issued in response to an <XACMLPolicyQuery>, and there are no <xacml:Policy> or <xacml:PolicySet> instances that meet the specifications of the associated <XACMLPolicyQuery>, then there shall be no elements in the <XACMLPolicyStatement>.

– <xacml:Policy> [Any Number]

  An <xacml:Policy> instance that meets the specifications of the associated <XACMLPolicyQuery>, if any.

– <xacml:PolicySet> [Any Number]

  An <xacml:PolicySet> instance that meets the specifications of the associated <XACMLPolicyQuery>, if any.

10.4 Element <saml:Assertion>

An <XACMLAuthzDecisionStatement>, <XACMLPolicyStatement>, or SAML standard <saml:AttributeStatement> shall be encapsulated in a <saml:Assertion>, which may be signed.

Most components of a <saml:Assertion> are fully specified in ITU-T Rec. X.1141. The following elements and XML attributes are further specified here for use with the SAML statement types defined and used in this profile.

Except as specified here, this profile imposes no requirements or restrictions on information in the <saml:Assertion> element.

10.4.1 Element <saml:Issuer>

The <saml:Issuer> element is a required element for holding information about "the SAML authority that is making the claim(s) in the assertion".

In order to support third party digital signatures, this profile does not require that the identity provided in the <saml:Issuer> element be consistent with the identity of the signer. It is up to the relying party to have an appropriate trust relationship with the authority that signs the <saml:Assertion>.

When a <saml:AttributeAssertion> is used to construct an XACML attribute, the string value of the <saml:Issuer> element will be used as the value of the XACML Issuer XML attribute, so the SAML value should be specified with this in mind.

10.4.2 Element <ds:Signature>

The <ds:Signature> element is an optional element for holding "An XML signature that authenticates the assertion".

A <ds:Signature> element may be used in an assertion used with an XACML statement. In order to support third party digital signatures, this profile does not require that the identity provided in the <saml:Issuer> element be consistent with the identity of the signer. It is up to the relying party to have an appropriate trust relationship with the authority that signs the <saml:Assertion>.

A relying party should verify any signature included in the assertion and should not use information derived from the assertion unless the signature is verified successfully.

10.4.3 Element <saml:Subject>

The <saml:Subject> element is an optional element used for holding "The subject of the statement(s) in the assertion".

The <saml:Subject> element shall not be included in an assertion that contains an <XACMLAuthzDecision> or <XACMLPolicy>.

In a <saml:AttributeAssertion> that is to be mapped to an XACML attribute, the <saml:Subject> element shall contain the identity of the entity to which the attribute and its value are bound. For an XACML <Subject> attribute, this identity should be consistent with the value of any XACML &subject-id; Attribute that occurs in the same <Subject> element.

For an XACML <Resource> attribute, this identity should be consistent with the value of any XACML &resource-id; attribute that occurs in the same <Resource> element. For an XACML <Action> attribute, this identity should be consistent with the value of any XACML &action-id; attribute that occurs in the same <Action> element. For an XACML <Environment> attribute, this identity should be consistent with the value of any XACML attribute that occurs in the same <Environment> element and provides an environment identity.

10.4.4 Element <saml:Conditions>

The <saml:Conditions> element is an optional element that is used for "conditions that must be taken into account in assessing the validity of and/or using the assertion".

The <saml:Conditions> element should contain NotBefore and NotOnOrAfter XML attributes to specify the limits on the validity of the assertion. If these XML attributes are present, the relying party should ensure that information derived from the assertion is used by a PDP for evaluating policies only when the value of the request context &current-dateTime; resource attribute is contained within the assertion's specified validity period.

10.5 Element <samlp:RequestAbstractType>

An <XACMLAuthzDecisionQuery> or <XACMLPolicyQuery> shall be encapsulated in a <samlp:RequestAbstractType> element, which may be signed.

Most components of a <samlp:RequestAbstractType> are fully specified in ITU-T Rec. X.1141. For use with the SAML query types defined and used in this profile, Element <saml:Issuer> and Element <ds:Signature> shall be used in the same manner as specified in the previous clause. Except as specified here, this profile imposes no requirements or restrictions on information in the <samlp:RequestAbstractType> element.

10.5.1 Element <saml:Issuer>

See 10.4.1, Element <saml:Issuer>.

10.5.2 Element <ds:Signature>

See 10.4.2, Element <ds:Signature>.

10.6 Element <samlp:Response>

An <XACMLAuthzDecisionStatement> or <XACMLPolicyStatement> shall be encapsulated in a <samlp:Response> element, which may be signed.

Most components of a <samlp:Response> are fully specified in ITU-T Rec. X.1141. The following elements and XML attributes are further specified here for use with the SAML statement types defined and used in this profile. Except as specified here, this profile imposes no requirements or restrictions on information in the <samlp:Response> element.

10.6.1 Element <samlp:Issuer>

See 10.4.1, Element <saml:Issuer>.

10.6.2 Element <ds:Signature>

See 10.4.2, Element <ds:Signature>.

10.6.3 Element <samlp:StatusCode>

The <samlp:StatusCode> element is a component of the <samlp:Status> element in the <samlp:Response>.

10.6.3.1 Response to <XACMLAuthzDecisionQuery>

In the response to an <XACMLAuthzDecisionQuery> request, the <samlp:StatusCode> Value XML attribute shall depend on the <xacml:StatusCode> element of the authorization decision <xacml:Status> element as follows:

1) urn:oasis:names:tc:SAML:2.0:status:Success

   This value for the <samlp:StatusCode> Value XML attribute shall be used if and only if the <xacml:StatusCode> value is urn:oasis:names:tc:xacml:1.0:status:ok.

2) urn:oasis:names:tc:SAML:2.0:status:Requester

   This value for the <samlp:StatusCode> Value XML attribute shall be used when the <xacml:StatusCode> value is urn:oasis:names:tc:xacml:1.0:status:missing-attribute or when the <xacml:StatusCode> value is urn:oasis:names:tc:xacml:1.0:status:syntax-error due to a syntax error in the <xacml:Request>.

3) urn:oasis:names:tc:SAML:2.0:status:Responder

   This value for the <samlp:StatusCode> Value XML attribute shall be used when the <xacml:StatusCode> value is urn:oasis:names:tc:xacml:1.0:status:syntax-error due to a syntax error in an <xacml:Policy> or <xacml:PolicySet>. Note that not all syntax errors in policies will be detected in conjunction with the processing of a particular query, so not all policy syntax errors will be reported this way.

4) urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

   This value for the <samlp:StatusCode> Value XML attribute shall be used only when the SAML interface at the PDP does not support the version of the SAML request message used in the query.

10.6.3.2 Response to < XACMLPolicyQuery >

In the response to an `<XACMLPolicyQuery>` request, the `<samlp:StatusCode>` value XML attribute shall be as specified in ITU-T Rec. X.1141.

11 XML digital signature profile

This clause provides a profile for use with W3C Signature:2002 for providing authentication and integrity protection for XACML schema instances.

A digital signature is useful for authentication and integrity protection only if the signed information includes a specification of the identity of the signer and a specification of the period during which the signed data object is to be considered valid. XACML itself does not define the format for such information, as XACML is intended to use other standards for functions other than the actual specification and evaluation of access control policies, requests and responses.

One appropriate format has been defined in SAML. A profile for the use of SAML with XACML schema instances is defined in clause 10. This profile therefore recommends the use of XACML schema instances in SAML Assertions, Requests, and Responses, which may then be digitally signed as specified in ITU-T Rec. X.1141.

11.1 Use of SAML

This profile recommends the use of XACML schema instances embedded in SAML Assertions, Requests and Responses as described in clause 10. Such SAML objects shall be digitally signed as described in 8.4/X.1141, *SAML and XML Signature Syntax and Processing*.

11.2 Canonicalization

In order for a digital signature to be verified by a relying party, the byte stream that was signed must be identical to the byte stream that is verified. To ensure this, the XML document being signed must be canonicalized (see W3C Canonicalization:2002). ITU-T Rec. X.1141 specifies the use of exclusive canonicalization (see W3C Canonicalization:2002).

11.2.1 Namespace elements in XACML data objects

Any XACML data object that is to be signed must specify all namespace elements used in the data object. If this is not done, then the data object will attract namespace definitions from ancestors of the data object that may differ from one envelope to another.

When exclusive canonicalization is used as the canonicalization or transform method, then the namespace of XACML schemas used by elements in an XACML data object must be bound to prefixes and included in the `InclusiveNamespacesPrefixList` parameter to http://www.w3.org/2001/10/xml-exc-c14n# (see W3C Canonicalization:2002).

11.2.2 Additional canonicalization considerations

Additional transformations on the XACML data object must usually be performed in order to ensure that the data object signed will match the data object that is verified. Some of these transformations are listed here, but this profile does not attempt to specify algorithms for performing these.

If an XACML data object includes data elements that may be represented in more than one form (such as (TRUE, FALSE), (1,0), (true,false)), then a transform method must be defined and specified for normalizing those data elements.

This profile recommends applying the following canonicalizations to values of the corresponding datatypes, whether occurring in XML attribute values or in XACML attributes.

1) Where a canonical representation for an XACML-defined datatype is defined in http://www.w3.org/2001/XMLSchema, then the value of the datatype must be put into the canonical form specified in http://www.w3.org/2001/XMLSchema. This includes boolean {"true", "false"}, double, dateTime, time, date, and hexBinary (upper-case).
2) http://www.w3.org/2001/XMLSchema#anyURI – Use the canonical form defined in IETF RFC 2396.
3) http://www.w3.org/2001/XMLSchema#base64Binary – Remove all line breaks and white space. Remove all characters following the first sequence of "=" characters. The Base64 transform (identifier: http://www.w3.org/TR/xmldsig-core/#sec-Base-64) may be useful in performing this canonicalization.

4) `urn:oasis:names:tc:xacml:1.0:data-type:x500Name` – First normalize according to IETF RFC 2253. If any RDN contains multiple `attributeTypeAndValue` pairs, re-order the `AttributeValuePairs` in that RDN in ascending order when compared as octet strings (see 11.6/X.690).

5) `urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name` – Normalize the domain-part of the name to lower case.

6) XPath expression – Apply http://www.w3.org/2002/06/xmldsig-filter2 to put the XPath expression into canonical form.

11.3 Signing schemas

The parsing of any XACML data object depends on having an accurate copy of all schemas on which the XACML data object depends. Note that the inclusion of a schema URI in the XACML schema instance attributes does not guarantee that an accurate copy of the schema will be used: an attacker may substitute a bogus schema that contains the correct identifier. Signatures can help protect against substitution or modification of the schemas on which an XACML data object depends. Use of signatures for this purpose is described in this clause.

In most cases, a data object signer should include a `<Reference>` element for each schema on which the XACML data object depends in the `<SignedInfo>` element that contains the `<Reference>` to or including the XACML data object itself.

In some cases, the data object signer knows that all PDPs that will evaluate a given XACML data object will have accurate copies of certain schemas needed to parse the data object, and does not want to force the PDP to verify the message digest for such schemas. In these cases the data object signer may omit `<Reference>` elements for any schema whose verification is not needed.

12 Hierarchical resource profile of XACML

It is often the case that a resource is organized as a hierarchy. Examples include file systems, XML documents, and organizations. This clause specifies how XACML can provide access control for a resource that is organized as a hierarchy.

Why are resources organized as hierarchies special? First of all, policies over hierarchies frequently apply the same access controls to entire sub-trees of the hierarchy. Being able to express a single policy constraint that will apply to an entire sub-tree of nodes in the hierarchy, rather than having to specify a separate constraint for each node, increases both ease of use and the likelihood that the policy will correctly reflect the desired access controls. Another special characteristic of hierarchical resources is that access to one node may depend on the value of another node. For example, a medical patient might be granted access to the "diagnosis" node in a XML document medical record only if the patient's name matches the value in the "patient name" node. Where this is the case, the requested node cannot be processed in isolation from the rest of the nodes in the hierarchy, and the PDP must have access to the values of other nodes. Finally, the identity of nodes in a hierarchy often depends on the position of the node in the hierarchy; there also may be multiple ways to describe the identity of a single node. In order for policies to apply to nodes as intended, attention must be paid to consistent representations for the identity of the nodes. Otherwise, a requester may bypass access controls by requesting a node using an identity that differs from the one used by the policy.

A resource organized as a hierarchy may be a "tree" (a hierarchy with a single root) or a "forest" (a hierarchy with multiple roots), but the hierarchy may not have cycles. Another term for these two types of hierarchy is "Directed Acyclic Graph" or "DAG". All such resources are called hierarchical resources in this profile. An XML document is always structured as a "tree". Other types of hierarchical resources, such as files in a file system that supports links, may be structured as "forests".

The nodes in a hierarchical resource are treated as individual resources. An authorization decision that permits access to an interior node does not imply that access to its descendant nodes is permitted. An authorization decision that denies access to an interior node does not imply that access to its descendant nodes is denied.

There are three types of facilities specified in this profile for dealing with hierarchical resources:

- Representing the identity of a node.
- Requesting access to a node.
- Stating policies that apply to one or more nodes.

Support for each of these facilities is optional.

This clause addresses two ways of representing a hierarchical resource. In the first way, the hierarchy of which the node is a part is represented as an XML document that is included in the request, and the requested resource is represented as a node in that document. In the second way, the requested resource is not represented as a node in an XML document, and there is no representation of the hierarchy of which it is a part included in the request. Note that the actual target resource in the first case need not be part of an XML document – it is merely represented that way in the Request. Likewise, the target resource in the second case might actually be part of an XML document, but is being represented in some other way in the Request. Thus there is no assumed correlation between the structure of the resource as represented in the Request and the actual structure of the physical resource being accessed.

Facilities for dealing with resources represented as nodes in XML documents can make use of the fact that the XML document itself is included in the decision request. XPath expressions can be used to reference nodes in this document in a standard way, and can provide unique representations for a given node in the document. These facilities are not available for hierarchical resources that are not represented as XML documents. Other means must be provided in the case of such non-XML resources for determining the location of the requested node in the hierarchy. In some cases, this can be done by including the node's position in the hierarchy as part of the node's identity. In other cases, a node may have more than one normative identity, such as when the pathname of a file in a file system can include hard links. In such cases, the XACML PDP's context handler may need to supply the identities of all the node's ancestors. For all these reasons, the facilities for dealing with nodes in XML documents differ from the facilities for dealing with nodes in other hierarchical resources.

In dealing with a hierarchical resource, it may be useful to request authorization decisions for multiple nodes in the resource in a single decision request. This clause may be considered to be layered on top of the Multiple Resource profile (see clause 9), which in turn is layered on top of the behaviour specified in clause 7. The functionality in this clause may, however, be layered directly on the functionality in clause 7.

This clause for hierarchical resources assumes that all requests for access to multiple nodes in a hierarchical resource have been resolved to individual requests for access to a single node.

12.1 Representing the identity of a node

In order for XACML policies to apply consistently to nodes in a hierarchical resource, it is necessary for the nodes in that resource to be represented in a consistent way. If a policy refers to a node using one representation, but a request refers to the node using a different representation, then the policy will not apply, and security may be compromised.

The following clauses describe recommended representations for nodes in hierarchical resources. Alternative representations of nodes in a given resource are permitted so long as all policy administration points and all policy enforcement points that deal with that resource have contracted to use the alternative representation.

12.1.1 Nodes in XML documents

This clause is normative, but it is optional.

The following URI shall be used as the identifier for the functionality specified in this part of this profile:

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-id
```

The identity of a node in a resource that is represented as an XML document instance shall be an XPath expression that evaluates to exactly that one node in the copy of the resource that is contained in the `<ResourceContent>` element of the `<Resource>` element of the `<Request>`.

12.1.2 Nodes in resources that are not XML documents

This clause is normative, but it is optional.

The following URI shall be used as the identifier for the functionality specified in this part of this profile:

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-id
```

The identity of a node in a hierarchical resource that is not represented as an XML document instance shall be represented as a URI that conforms to IETF RFC 2396. Such URIs are of the following form.

```
<scheme> ":" <authority> "/" <pathname>
```

File system resources shall use the "file:" scheme. If no standard `<scheme>` for the resource type is specified in IETF RFC 2396 or in a related standard for a registered URI scheme, then the URI shall use the "`file`:" scheme.

The `<pathname>` portion of the URI shall be of the form:

```
<root name> [ "/" <node name> ]*
```

The sequence of <root name> and <node name> values shall correspond to the individual hierarchical component names of ancestors of the represented node along the path from a <root> node to the represented node.

The following canonicalization shall be used.

- The encoding of the URI shall be UTF8.
- Case-insensitive portions of the URI shall be lower case.
- Escaping of characters shall conform to IETF RFC 2396.
- The <authority> portion of the URI shall be specified and shall be the standard authority representation for the given resource type. Where the <authority> could be specified using either a domain name system (DNS) name or a numeric IPv4 or IPv6 address, the DNS name shall be used.
- The components of the <pathname> portion of the URI shall be specified using the canonical form for such path components at the <authority>.
- In accordance with IETF RFC 2396, the separator character between hierarchical components of the <pathname> portion of the URI shall be the character "/". Sequences of the "/" character shall be resolved to a single "/". Node identities shall not terminate with the "/" character.
- The <pathname> shall contain no soft links.
- All <pathname> values shall be absolute.
- If there is more than one fully resolved, absolute path from a <root> at the <authority> to the represented node, then a separate resource attribute with AttributeId "urn:oasis:names:tc:xacml:1.0:resource:resource-id" and DataType http://urn:oasis:names:tc:xacml:1.0:data-type:anyURI shall be present in the Request Context for each such path.

12.2 Requesting access to a node

In order for XACML policies to apply consistently to nodes in a hierarchical resource, it is necessary for each request context that represents a request for access to a node in that resource to use a consistent description of that node access. If a policy refers to certain expected attributes of a node, but the request context does not contain those attributes, or if the attributes are not expressed in the expected way, then the policy may not apply, and security may be compromised.

The following clauses describe recommended request context descriptions of access to nodes in hierarchical resources. Alternative representations of such requests are permitted so long as all policy administration points and all policy enforcement points that deal with that resource have contracted to use the alternative representation.

12.2.1 Nodes in an XML document

This clause is normative, but it is optional.

The following URI shall be used as the identifier for the functionality specified in this part of this profile:

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req
```

The attributes with AttributeIds of:

```
"urn:oasis::names:tc:xacml:2.0:resource:resource-parent"
"urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor"
```

and:

```
"urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor-or-self"
```

are optional to implement. If supported for use in resources represented as XML documents, the following URIs shall be used as identifiers for the functionality they represent:

```
"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-parent"
"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-ancestor"
```

and:

```
"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-ancestor-
or-self"
```

In order to request access to a resource represented as a node in an XML document, the request context <Resource> element shall contain the following elements and XML attributes.

– A <ResourceContent> element that contains the entire XML document instance of which the requested node is a part.

– An <Attribute> element with an AttributeId of "urn:oasis::names:tc:xacml:1.0:resource:resource-id" and a DataType of "urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression". The <AttributeValue> of this <Attribute> shall be an XPath expression whose context node shall be the one and only child of the <ResourceContent> element. This XPath expression shall evaluate to a nodeset containing the single node in the <ResourceContent> element that is the node to which access is requested. This <Attribute> may specify an Issuer.

– An <Attribute> element with an AttributeId of "urn:oasis::names:tc:xacml:2.0:resource:resource-parent" and a DataType of "urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression". The <AttributeValue> of this <Attribute> shall be an XPath expression; the context node for this XPath expression shall be the one and only child of the <ResourceContent> element. This XPath expression shall evaluate to a nodeset containing the single node in the <ResourceContent> element that is the immediate parent of the node represented in the "resource-id" attribute. This <Attribute> may specify an Issuer.

– For each node in the XML document instance that is an ancestor of the node represented by the "resource-id" attribute, an <Attribute> element with an AttributeId of "urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor" and a DataType of "urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression". The <AttributeValue> of this <Attribute> shall be an XPath expression; the context node for this XPath expression shall be the one and only child of the <ResourceContent> element. This XPath expression shall evaluate to a nodeset containing the single node in the <ResourceContent> element that is the respective ancestor of the node represented in the "resource-id" attribute. For each "resource-parent" attribute, there shall be a corresponding "resource-ancestor" attribute. This <Attribute> may specify an Issuer.

– For each node in the XML document instance that is an ancestor of the node represented by the "resource-id" attribute, and for the "resource-id" node itself, an <Attribute> element with an AttributeId of "urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor-or-self" and a DataType of "urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression". The <AttributeValue> of this <Attribute> shall be an XPath expression; the context node for this XPath expression shall be the one and only child of the <ResourceContent> element. This XPath expression shall evaluate to a nodeset containing the single node in the <ResourceContent> element that is the respective ancestor of the node represented in the "resource-id" attribute, or that is the "resource-id" node itself. For each "resource-parent" and "resource-id" attribute, there shall be a corresponding "resource-ancestor-or-self" attribute. This <Attribute> may specify an Issuer.

Additional attributes may be included in the <Resource> element. In particular, the following attribute may be included.

– An <Attribute> element with an AttributeId of "urn:oasis::names:tc:xacml:2.0:resource:document-id" and a DataType of "urn:oasis:names:tc:xacml:1.0:data-type:anyURI". The <AttributeValue> of this <Attribute> shall be a URI that identifies the XML document of which the requested resource is a part, and of which a copy is present in the <ResourceContent> element. This <Attribute> may specify an Issuer.

12.2.2 Nodes in a resource that is not an XML document

This clause is normative, but it is optional.

The following URI shall be used as the identifier for the functionality specified in this part of this clause:

urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req

The attributes with AttributeIds of:

"urn:oasis::names:tc:xacml:2.0:resource:resource-parent"

"urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor"

and:

"urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor-or-self"

are optional to implement. If supported for use in resources that are not represented as XML documents, the following URIs shall be used as identifiers for the functionality they represent:

"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-parent"

"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-ancestor"

and:

"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-ancestor-or-self"

In order to request access to a node in a hierarchical resource that is not represented as an XML document, the request context <Resource> element shall not contain a <ResourceContent> element. The request context <Resource> element shall contain the following elements and XML attributes. Note that a node in a hierarchical resource that is not represented as an XML document may have multiple parents. For example, in a file system that supports hard links, there may be multiple normative paths to a single file. Each such path may contain different sets of parents and ancestors.

- For each normative representation of the requested node, an <Attribute> element with AttributeId of "urn:oasis::names:tc:xacml:1.0:resource:resource-id". The <AttributeValue> of this <Attribute> shall be a unique, normative identity of the node to which access is requested. The DataType of this <Attribute> shall depend on the representation chosen for the identity of nodes in this particular resource. This <Attribute> may specify an Issuer.
- For each immediate parent of the node specified in the "resource-id" attribute or attributes, and for each normative representation of that parent node, an <Attribute> element with AttributeId "urn:oasis::names:tc:xacml:2.0:resource:resource-parent". The <AttributeValue> of this <Attribute> shall be the normative identity of the parent node. The DataType of this <Attribute> shall depend on the representation chosen for the identity of nodes in this particular resource. This <Attribute> may specify an Issuer. If the requested node is part of a forest rather than part of a single tree, or if the parent node has more than one normative representation, there shall be at least one instance of this attribute for each parent along each path to the multiple roots of which the requested node is a descendant, and for each normative representation of each such parent.
- For each ancestor of the node specified in the "resource-id" attribute or attributes, and for each normative representation of that ancestor node, an <Attribute> element with AttributeId "urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor". The <AttributeValue> of this <Attribute> shall be the normative identity of the ancestor node. The DataType of this <Attribute> shall depend on the representation chosen for the identity of nodes in this particular resource. This <Attribute> may specify an Issuer. For each "resource-parent" attribute, there shall be a corresponding "resource-ancestor" attribute. If the requested node is part of a forest rather than part of a single tree, or if the ancestor node has more than one normative representation, there shall be at least one instance of this attribute for each ancestor along each path to the multiple roots of which the requested node is a descendant, and for each normative representation of each such ancestor. The order of the values for this attribute do not necessarily reflect the position of each ancestor node in the hierarchy.
- For each ancestor of the node specified in the "resource-id" attribute or attributes, and for each normative representation of that ancestor node, and for each normative representation of the "resource-id" node itself, an <Attribute> element with AttributeId "urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor-or-self". The <AttributeValue> of this <Attribute> shall be the respective normative identity of the ancestor node or of the "resource-id" node itself. The DataType of this <Attribute> shall depend on the representation chosen for the identity of nodes in this particular resource. This <Attribute> may specify an Issuer. For each "resource-ancestor" and "resource-id" attribute, there shall be a corresponding "resource-ancestor-or-self" attribute. If the requested node is part of a forest rather than part of a single tree, or if the ancestor node has more than one normative representation, there shall be at least one instance of this attribute for each ancestor along each path to the multiple roots of which the requested node is a descendant, and for each normative representation of each such ancestor.

The order of the values for this attribute do not necessarily reflect the position of each ancestor node in the hierarchy.

Additional attributes may be included in the <Resource> element.

12.3 Stating policies that apply to nodes

This clause is informative.

This clause describes various ways to specify a policy predicate that can apply to multiple nodes in a hierarchical resource. This is not intended to be an exhaustive list.

12.3.1 Policies applying to nodes in any hierarchical resource

This clause is informative.

Resource attributes with the following AttributeId values, described in 12.5, may be used to state policies that apply to one or more nodes in any hierarchical resource.

```
urn:oasis:names:tc:xacml:2.0:resource:resource-parent
urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor
urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor-or-self
```

Note that a <ResourceAttributeDesignator> that refers to the "resource-parent", "resource-ancestor", or "resource-ancestor-or-self" attribute will return a bag of values representing all normative identities of all parents, ancestors, or ancestors plus the resource itself, respectively, of the resource to which access is being requested. The representations of the identities of these parents, ancestors, or self will not necessarily indicate the path from the root of the hierarchy to the respective parent, ancestor, or self unless the representation recommended in 12.2.2, Nodes in a resource that is not an XML document, is used.

The standard XACML bag and higher-order bag functions may be used to state policies that apply to one or more nodes in any hierarchical resource. The nodes used as arguments to these functions may be specified using a <ResourceAttributeDesignator> with the "resource-parent", "resource-ancestor", or "resource-ancestor-or-self" AttributeId value.

12.3.2 Policies applying only to nodes in XML documents

This clause is informative.

For hierarchical resources that are represented as XML document instances, the following function may be used to state policy predicates that apply to one or more nodes in that resource.

```
urn:oasis:names:tc:xacml:2.0:function:xpath-node-match
```

The standard XACML <AttributeSelector> element may be used in policies to refer to all or portions of a resource represented as an XML document and contained in the <ResourceContent> element of a request context.

The standard XACML bag and higher-order bag functions may be used to state policies that apply to one or more nodes in a resource represented as an XML document. The nodes used as arguments to these functions may be specified using an <AttributeSelector> that selects a portion of the <ResourceContent> element of the <Resource> element.

12.3.3 Policies applying only to nodes in non-XML resources

This clause is informative.

For hierarchical resources that are not represented as XML document instances, and where the URI representation of nodes as specified in this profile is used, the following functions may be used to state policies that apply to one or more nodes in that resource.

```
urn:oasis:names:tc:xacml:1.0:function:anyURI-equal
urn:oasis:names:tc:xacml:1.0:function:regexp-uri-match
```

12.4 New DataType: xpath-expression

This clause is normative but optional.

The following value for the XML DataType attribute value may be supported for use with hierarchical resources represented as XML documents. Support for this DataType is required in order to support clause 12.1.1.

The DataType represented by the following URI represents an XPath expression. Attribute values having this DataType shall be strings that are to be interpreted as XPath expressions. The result of evaluating such an attribute shall be the nodeset that results from evaluating the XPath expression. If the string is not a valid XPath expression, the result of evaluating the attribute shall be "Indeterminate".

```
urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression
```

12.5 New attribute identifiers

This clause is normative but optional.

12.5.1 document-id

The following identifier indicates the identity of the XML document that represents the hierarchy of which the requested resource is a part, and of which a copy is present in the <ResourceContent> element. Whenever access to a node in a resource represented as an XML document is requested, one or more instances of an attribute with this AttributeId may be provided in the <Resource> element of the request context. The DataType of these attributes shall be "urn:oasis:names:tc:xacml:1.0:data-type:anyURI".

```
urn:oasis:names:tc:xacml:2.0:resource:document-id
```

12.5.2 resource-parent

The following identifier indicates one normative identity of one parent node in the tree or forest of which the requested node is a part. Whenever access to a node in a hierarchical resource is requested, one instance of an attribute with this AttributeId shall be provided in the <Resource> element of the request context for each normative representation of each node that is a parent of the requested node.

```
urn:oasis:names:tc:xacml:2.0:resource:resource-parent
```

12.5.3 resource-ancestor

The following identifier indicates one normative identity of one ancestor node in the tree or forest of which the requested node is a part. Whenever access to a node in a hierarchical resource is requested, one instance of an attribute with this AttributeId shall be provided in the <Resource> element of the request context for each normative representation of each node that is an ancestor of the requested node.

```
urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor
```

12.5.4 resource-ancestor-or-self

The following identifier indicates one normative identity of one ancestor node in the tree or forest of which the requested node is a part, or one normative identity of the requested node itself. Whenever access to a node in a hierarchical resource is requested, one instance of an attribute with this AttributeId shall be provided in the <Resource> element of the request context for each normative representation of each node that is an ancestor of the requested node, and for each normative representation of the requested node itself.

```
urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor-or-self
```

12.6 New profile identifiers

The following URI values shall be used as identifiers for the functionality specified in various clauses of this profile:

Clause 12.1.1: Nodes in XML documents

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-id
```

Clause 12.1.2: Nodes in resources that are not XML documents

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-id
```

Clause 12.2.1: Nodes in an XML document

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req
```

Support for the "resource-parent", "resource-ancestor", and "resource-ancestor-or-self" attributes is optional within this clause, so these have separate identifiers:

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-parent
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-ancestor
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-ancestor-
or-self
```

Clause 12.2.2: Nodes in a resource that is not an XML document

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req
```

Support for the "`resource-parent`", "`resource-ancestor`", and "`resource-ancestor-or-self`" attributes is optional within this clause, so these have separate identifiers:

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-parent
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-
ancestor
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-
ancestor-or-self
```

13 Privacy policy profile

Two obligations on a data custodian are to ensure that the use of personal data is limited to the fulfilment of the purposes for which it is collected, or other purposes that are not incompatible with these, and to prevent the disclosure of personal data except with the consent of the data subject or by the authority of law. This clause provides a profile for standard attributes and a standard `<Rule>` element for enforcing these obligations, related to the purpose for which personally identifiable information is collected and used.

13.1 Standard attributes

This profile defines two attributes.

```
urn:oasis:names:tc:xacml:2.0:resource:purpose
```

This attribute, of type "http://www.w3.org/2001/XMLSchema#string", indicates the purpose for which the data resource was collected. The owner of the resource should be informed and consent to the use of the resource for this purpose. The attribute value may be a regular expression. The custodian's privacy policy should define the semantics of all available values.

```
urn:oasis:names:tc:xacml:2.0:action:purpose
```

This attribute, of type "http://www.w3.org/2001/XMLSchema#string", indicates the purpose for which access to the data resource is requested. Action purposes may be organized hierarchically, in which case the value must represent a node in the hierarchy.

13.2 Standard rules: Matching purpose

This rule must be used with the "`urn:oasis:names:tc:xacml:2.0:rule-combining-algorithm:deny-overrides`" rule-combining algorithm. It stipulates that access shall be denied unless the purpose for which access is requested matches, by regular-expression match, the purpose for which the data resource was collected.

```
<?xml version="1.0" encoding="UTF-8"?>
<Rule xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os" RuleId="
urn:oasis:names:tc:xacml:2.0:matching-purpose"
Effect="Permit">
      <Condition FunctionId="urn:oasis:names:tc:xacml:2.0:function:regexp-
string-match">
            <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:2.0:resource:purpose"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:2.0:action:purpose"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Condition>
</Rule>
```

Annex A

Data-types and functions

A.1 Introduction

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

This Recommendation describes the primitive data-types and bags. The standard functions are named and their operational semantics are described.

NOTE – See [IEEE 754] and [RBAC] for information string representation of numeric values.

A.2 Data-types

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

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

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

An XACML PDP shall be capable of converting string representations into various primitive data-types.

NOTE – For integers and doubles, XACML shall use the conversions described in [IEEE 754].

XACML defines six data-types; these are:

```
"urn:oasis:names:tc:xacml:2.0:data-type:dayTimeDuration"
"urn:oasis:names:tc:xacml:2.0:data-type:yearMonthDuration"
"urn:oasis:names:tc:xacml:1.0:data-type:x500Name"
"urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"
"urn:oasis:names:tc:xacml:2.0:data-type:ipAddress"
"urn:oasis:names:tc:xacml:2.0:data-type:dnsName"
```

These types represent identifiers for subjects or resources and appear in several standard applications, such as TLS/SSL and electronic mail.

A.2.1 Duration in days and times

The "`urn:oasis:names:tc:xacml:2.0:data-type:dayTimeDuration`" primitive type is defined as a restriction of the **xs:duration** type, retaining only the sign, year and month components.

```
<xs:simpleType name='urn:oasis:names:tc:xacml:2.0:data-
type:yearMonthDuration'>
    <xs:restriction base='xs:duration'>
        <xsd:pattern value="[-]?P\p{Nd}+(Y(\p{Nd}+M)?|M)"/>
    </xs:restriction>
</xs:simpleType>
```

The value of a `yearMonthDuration` is in units of months, and is:

```
('value of the year component' * 12) + ('value of the month component')
```

If the sign component is "-", then the resulting value is negative; otherwise, the resulting value is positive.

A.2.2 Duration in years and months

The "`urn:oasis:names:tc:xacml:2.0:data-type:yearMonthDuration`" primitive type is defined as a restriction of the **xs:duration** type, retaining only the sign, day, hour, minute and second components.

```
<xs:simpleType name='urn:oasis:names:tc:xacml:2.0:data-
type:dayTimeDuration'>
    <xs:restriction base='xs:duration'>
          <xsd:pattern value="[-
]?P(\p{Nd}D(T(\p{Nd}+(H(\p{Nd}+(M(\p{Nd}+(\.\p{Nd}*)?S
            |\.\p{Nd}+S)?|(\.\p{Nd}*)?S)|(\.\p{Nd}*)?S)?|M(\p{Nd}+
            (\.\p{Nd}*)?S|\.\p{Nd}+S)?|(\.\p{Nd}*)?S)|\.\p{Nd}+S))?
            |T(\p{Nd}+(H(\p{Nd}+(M(\p{Nd}+(\.\p{Nd}*)?S|\.\p{Nd}+S)?

|(\.\p{Nd}*)?S)|(\.\p{Nd}*)?S)?|M(\p{Nd}+(\.\p{Nd}*)?S|\.\p{Nd}+S)?
            |(\.\p{Nd}*)?S)|\.\p{Nd}+S))"/>
    </xs:restriction>
</xs:simpleType>
```

The value of a `dayTimeDuration` is in units of seconds, and is:

```
('value of the day component' * 24) +
('value of the hour component' * 60) +
('value of the minute component' * 60) +
('value of the second component')
```

If the sign component is "-", then the resulting value is negative; otherwise, the resulting value is positive.

A.2.3 X.500 directory name

The "`urn:oasis:names:tc:xacml:1.0:data-type:x500Name`" primitive type represents an X.520 Distinguished Name. The valid syntax for such a name is described in IETF RFC 2253 .

A.2.4 IETF RFC 822 name

The "`urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name`" primitive type represents an electronic mail address. The valid syntax for such a name is described in IETF RFC 2821, 4.1.2.

A.2.5 IP address

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

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

For an IPv4 address, the address and mask are formatted in accordance with the syntax for a "host" in IETF RFC 2396, 3.2.

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

A.2.6 DNS name

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

```
dnsName = hostname [ ":" portrange ]
```

The hostname is formatted in accordance with IETF RFC 2396, 3.2, except that a wildcard "*" may be used in the left-most component of the hostname to indicate "any subdomain" under the domain specified to its right.

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

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

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

A.3 Functions

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

A.3.1 Equality predicates

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

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

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall return "True" if, and only if, the value of both of its arguments are of equal length and each string is determined to be equal byte-by-byte according to the function "integer-equal". Otherwise, it shall return "False".

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

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

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

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#integer" and shall return an "http://www.w3.org/2001/XMLSchema#boolean".

NOTE 1 – See [IEEE 754] for information on integer evaluation on integers.

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

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#double" and shall return an "http://www.w3.org/2001/XMLSchema#boolean".

NOTE 2 – See [IEEE 754] on how to evaluate doubles.

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

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#date" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall return "True" if, and only if, the values of the two arguments are equal. If either argument lacks an explicit timezone, then a timezone value shall be provided by the implementation.

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

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall return "True" if, and only if, the values of the two arguments are equal. If either argument lacks an explicit timezone, then a timezone value shall be provided by the implementation.

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

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall return "True" if, and only if, the values of the two arguments are equal. If either argument lacks an explicit timezone, then a timezone value shall be provided by the implementation.

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

This function shall take two arguments of data-type "urn:oasis:names:tc:xacml:2.0:data-types:dayTimeDuration" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall return "True" if, and only if, the values of the two arguments are equal. Note that the lexical representation of each argument must be converted to a value expressed in fractional seconds.

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

This function shall take two arguments of data-type "urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall return "True" if, and only if, the values of the two arguments are equal. Note that the lexical representation of each argument must be converted to a value expressed in integer months.

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

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#anyURI" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall return "True" if, and only if, the values of the two arguments are equal on a codepoint-by-codepoint basis.

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

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

1) Normalize the two arguments according to IETF RFC 2253.
2) If any RDN contains multiple attributeTypeAndValue pairs, re-order the AttributeValuePairs in that RDN in ascending order when compared as octet strings (described in 11.6/X.690).
3) Compare RDNs using the rules in IETF RFC 3280, 4.1.2.4.

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

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

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

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

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

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

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

A.3.2 Arithmetic functions

All of the following functions shall take two arguments of the specified data-type, integer or double, and shall return an element of integer or double data-type, respectively. However, the "add" functions may take more than two arguments. In an expression that contains any of these functions, if any argument is "Indeterminate", then the expression shall evaluate to "Indeterminate". In the case of the divide functions, if the divisor is zero, then the function shall evaluate to "Indeterminate".

NOTE – Each function evaluation should proceed as specified by their logical counterparts in [IEEE 754].

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

This function may have two or more arguments.

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

This function may have two or more arguments.

```
urn:oasis:names:tc:xacml:1.0:function:integer-subtract
urn:oasis:names:tc:xacml:1.0:function:double-subtract
urn:oasis:names:tc:xacml:1.0:function:integer-multiply
urn:oasis:names:tc:xacml:1.0:function:double-multiply
urn:oasis:names:tc:xacml:1.0:function:integer-divide
urn:oasis:names:tc:xacml:1.0:function:double-divide
urn:oasis:names:tc:xacml:1.0:function:integer-mod
```

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

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

A.3.3 String conversion functions

The following functions convert between values of the data-type "http://www.w3.org/2001/XMLSchema#string" primitive types.

```
urn:oasis:names:tc:xacml:1.0:function:string-normalize-space
```

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string" and shall normalize the value by stripping off all leading and trailing white space characters.

```
urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case
```

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#string" and shall normalize the value by converting each upper case character to its lower case equivalent.

A.3.4 Numeric data-type conversion functions

The following functions convert between the data-type "http://www.w3.org/2001/XMLSchema#integer" and "http://www.w3.org/2001/XMLSchema#double" primitive types.

```
urn:oasis:names:tc:xacml:1.0:function:double-to-integer
```

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#double" and shall truncate its numeric value to a whole number and return an element of data-type "http://www.w3.org/2001/XMLSchema#integer".

```
urn:oasis:names:tc:xacml:1.0:function:integer-to-double
```

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#integer" and shall promote its value to an element of data-type "http://www.w3.org/2001/XMLSchema#double" with the same numeric value.

A.3.5 Logical functions

This clause contains the specification for logical functions that operate on arguments of data-type "http://www.w3.org/2001/XMLSchema#boolean".

```
urn:oasis:names:tc:xacml:1.0:function:or
```

This function shall return "False" if it has no arguments and shall return "True" if at least one of its arguments evaluates to "True". The order of evaluation shall be from first argument to last. The evaluation shall stop with a result of "True" if any argument evaluates to "True", leaving the rest of the arguments unevaluated.

```
urn:oasis:names:tc:xacml:1.0:function:and
```

This function shall return "True" if it has no arguments and shall return "False" if one of its arguments evaluates to "False". The order of evaluation shall be from first argument to last. The evaluation shall stop with a result of "False" if any argument evaluates to "False", leaving the rest of the arguments unevaluated.

```
urn:oasis:names:tc:xacml:1.0:function:n-of
```

The first argument to this function shall be of data-type http://www.w3.org/2001/XMLSchema#integer. The remaining arguments shall be of data-type http://www.w3.org/2001/XMLSchema#boolean. The first argument specifies the minimum number of the remaining arguments that must evaluate to "True" for the expression to be considered "True". If the first argument is 0, the result shall be "True". If the number of arguments after the first one is less than the value of the first argument, then the expression shall result in "Indeterminate". The order of evaluation shall be: first evaluate the integer value, then evaluate each subsequent argument. The evaluation shall stop and return "True" if the specified number of arguments evaluate to "True". The evaluation of arguments shall stop if it is determined that evaluating the remaining arguments will not satisfy the requirement.

```
urn:oasis:names:tc:xacml:1.0:function:not
```

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#boolean". If the argument evaluates to "True", then the result of the expression shall be "False". If the argument evaluates to "False", then the result of the expression shall be "True".

NOTE – When evaluating and, or, or n-of, it may not be necessary to attempt a full evaluation of each argument in order to determine whether the evaluation of the argument would result in "Indeterminate". Analysis of the argument regarding the availability of its attributes, or other analysis regarding errors, such as "divide-by-zero", may render the argument error free. Such arguments occurring in the expression in a position after the evaluation is stated to stop need not be processed.

A.3.6 Numeric comparison functions

These functions form a minimal set for comparing two numbers, yielding a boolean result.

NOTE – These functions should comply with rules governed in [IEEE 754].

```
urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal
urn:oasis:names:tc:xacml:1.0:function:integer-less-than
urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal
urn:oasis:names:tc:xacml:1.0:function:double-greater-than
urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal
urn:oasis:names:tc:xacml:1.0:function:double-less-than
urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal
```

A.3.7 Date and time arithmetic functions

These functions perform arithmetic operations with date and time.

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration
```

This function shall take two arguments, the first shall be of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and the second shall be of data-type "`urn:oasis:names:tc:xacml:2.0:data-types:dayTimeDuration`". It shall return a result of "http://www.w3.org/2001/XMLSchema#dateTime". This function shall return the value by adding the second argument to the first argument according to W3C DataTypes:2001, Appendix E.

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration
```

This function shall take two arguments, the first shall be a "http://www.w3.org/2001/XMLSchema#dateTime" and the second shall be a "`urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration`". It shall return a result of "http://www.w3.org/2001/XMLSchema#dateTime". This function shall return the value by adding the second argument to the first argument according to W3C DataTypes:2001, Appendix E.

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration
```

This function shall take two arguments, the first shall be a "http://www.w3.org/2001/XMLSchema#dateTime" and the second shall be a "`urn:oasis:names:tc:xacml:2.0:data-types:dayTimeDuration`". It shall return a result of "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration, then this function shall return the value by adding the corresponding negative duration, as per W3C DataTypes:2001, Appendix E. If the second argument is a negative duration, then the result shall be as if the function "`urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration`" had been applied to the corresponding positive duration.

urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration

This function shall take two arguments, the first shall be a "http://www.w3.org/2001/XMLSchema#dateTime" and the second shall be a "urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration". It shall return a result of "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration, then this function shall return the value by adding the corresponding negative duration, as per W3C DataTypes:2001, Appendix E. If the second argument is a negative duration, then the result shall be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration" had been applied to the corresponding positive duration.

urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration

This function shall take two arguments, the first shall be a "http://www.w3.org/2001/XMLSchema#date" and the second shall be a "urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration". It shall return a result of "http://www.w3.org/2001/XMLSchema#date". This function shall return the value by adding the second argument to the first argument according to W3C DataTypes:2001, Appendix E.

urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration

This function shall take two arguments, the first shall be a "http://www.w3.org/2001/XMLSchema#date" and the second shall be a "urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration". It shall return a result of "http://www.w3.org/2001/XMLSchema#date". If the second argument is a positive duration, then this function shall return the value by adding the corresponding negative duration, as per the (W3C DataTypes:2001, Appendix E). If the second argument is a negative duration, then the result shall be as if the function "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" had been applied to the corresponding positive duration.

A.3.8 Non-numeric comparison functions

These functions perform comparison operations on two arguments of non-numerical types.

urn:oasis:names:tc:xacml:1.0:function:string-greater-than

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the arguments are compared byte by byte and, after an initial prefix of corresponding bytes from both arguments that are considered equal by "urn:oasis:names:tc:xacml:1.0:function:integer-equal", the next byte by byte comparison is such that the byte from the first argument is greater than the byte from the second argument by the use of the function "urn:oasis:names:tc:xacml:2.0:function:integer-greater-then". Otherwise, it shall return "False".

urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return a result as if evaluated with the logical function "urn:oasis:names:tc:xacml:1.0:function:or" with two arguments containing the functions "urn:oasis:names:tc:xacml:1.0:function:string-greater-than" and "urn:oasis:names:tc:xacml:1.0:function:string-equal" containing the original arguments.

urn:oasis:names:tc:xacml:1.0:function:string-less-than

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the arguments are compared byte by byte and, after an initial prefix of corresponding bytes from both arguments that are considered equal by "urn:oasis:names:tc:xacml:1.0:function:integer-equal", the next byte by byte comparison is such that the byte from the first argument is less than the byte from the second argument by the use of the function "urn:oasis:names:tc:xacml:1.0:function:integer-less-than". Otherwise, it shall return "False".

urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return a result as if evaluated with the function "urn:oasis:names:tc:xacml:1.0:function:or" with two arguments containing the functions "urn:oasis:names:tc:xacml:1.0:function:string-less-than" and "urn:oasis:names:tc:xacml:1.0:function:string-equal" containing the original arguments.

urn:oasis:names:tc:xacml:1.0:function:time-greater-than

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is greater than the second argument according to the order relation specified for http://www.w3.org/2001/XMLSchema#time (W3C Signature:2002, 3.2.8). Otherwise, it shall return "False".

NOTE 1 – It is not appropriate to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

`urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal`

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is greater than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" (W3C Datatypes:2001, 3.2.8). Otherwise, it shall return "False".

NOTE 2 – It is not appropriate to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

`urn:oasis:names:tc:xacml:1.0:function:time-less-than`

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is less than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" (W3C Datatypes:2001, 3.2.8). Otherwise, it shall return "False".

NOTE 3– It is not appropriate to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

`urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal`

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is less than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time". Otherwise, it shall return "False".

NOTE 4 – It is not appropriate to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

`urn:oasis:names:tc:xacml:1.0:function:time-in-range`

This function shall take three arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if the first argument falls in the range defined inclusively by the second and third arguments. Otherwise, it shall return "False". Regardless of its value, the third argument shall be interpreted as a time that is equal to, or later than by less than twenty-four hours, the second argument. If no time zone is provided for the first argument, it shall use the default time zone at the context handler. If no time zone is provided for the second or third arguments, then they shall use the time zone from the first argument.

`urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than`

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is greater than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#dateTime" by W3C Datatype:2001, 3.2.7. Otherwise, it shall return "False".

NOTE 5 – If a `dateTime` value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in W3C Datatype:2001.

`urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal`

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is greater than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#dateTime" by W3C Datatype:2001, 3.2.7. Otherwise, it shall return "False".

NOTE 6 – If a `dateTime` value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in W3C Datatype:2001.

`urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than`

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is less than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#dateTime" by W3C Datatype:2001, 3.2.7. Otherwise, it shall return "False".

NOTE 7 – If a `dateTime` value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in W3C Datatype:2001.

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal
```

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema# dateTime" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is less than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#dateTime" by W3C Datatypes:2001, 3.2.7. Otherwise, it shall return "False".

NOTE 8 – If a `dateTime` value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in W3C Datatype:2001.

```
urn:oasis:names:tc:xacml:1.0:function:date-greater-than
```

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#date" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is greater than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#date". Otherwise, it shall return "False".

NOTE 9 – If a date value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in W3C Datatype:2001.

```
urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal
```

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#date" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is greater than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#date". Otherwise, it shall return "False".

NOTE 10 – If a date value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in W3C Datatype:2001.

```
urn:oasis:names:tc:xacml:1.0:function:date-less-than
```

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#date" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is less than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#date". Otherwise, it shall return "False".

NOTE 11 – If a date value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in W3C Datatype:2001.

```
urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal
```

This function shall take two arguments of data-type "http://www.w3.org/2001/XMLSchema#date" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is less than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#date". Otherwise, it shall return "False".

NOTE 12 – If a date value does not include a time-zone value, then an implicit time-zone value shall be assigned, as described in W3C Datatype:2001.

A.3.9 String functions

The following functions operate on strings and URIs.

```
urn:oasis:names:tc:xacml:2.0:function:string-concatenate
```

This function shall take two or more arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and shall return a "http://www.w3.org/2001/XMLSchema#string". The result shall be the concatenation, in order, of the arguments.

```
urn:oasis:names:tc:xacml:2.0:function:url-string-concatenate
```

This function shall take one argument of data-type "http://www.w3.org/2001/XMLSchema#anyURI" and one or more arguments of type "http://www.w3.org/2001/XMLSchema#string", and shall return a "http://www.w3.org/2001/XMLSchema#anyURI". The result shall be the URI constructed by appending, in order, the "string" arguments to the "anyURI" argument.

A.3.10 Bag functions

These functions operate on a bag of 'type' values, where type is one of the primitive data-types. Some additional conditions defined for each function below shall cause the expression to evaluate to "Indeterminate".

`urn:oasis:names:tc:xacml:1.0:function:type-one-and-only`

This function shall take a bag of 'type' values as an argument and shall return a value of '-type'. It shall return the only value in the bag. If the bag does not have one and only one value, then the expression shall evaluate to "Indeterminate".

`urn:oasis:names:tc:xacml:1.0:function:type-bag-size`

This function shall take a bag of 'type' values as an argument and shall return an "http://www.w3.org/2001/XMLSchema#integer" indicating the number of values in the bag.

`urn:oasis:names:tc:xacml:1.0:function:type-is-in`

This function shall take an argument of 'type' as the first argument and a bag of type values as the second argument and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall evaluate to "True" if, and only if, the first argument matches by the "`urn:oasis:names:tc:xacml:x.x:function:type-equal`" any value in the bag. Otherwise, it shall return "False".

`urn:oasis:names:tc:xacml:1.0:function:type-bag`

This function shall take any number of arguments of 'type' and return a bag of 'type' values containing the values of the arguments. An application of this function to zero arguments shall produce an empty bag of the specified data-type.

A.3.11 Set functions

These functions operate on bags mimicking sets by eliminating duplicate elements from a bag.

`urn:oasis:names:tc:xacml:1.0:function:type-intersection`

This function shall take two arguments that are both a bag of 'type' values. It shall return a bag of 'type' values such that it contains only elements that are common between the two bags, which is determined by "`urn:oasis:names:tc:xacml:x.x:function:type-equal`". No duplicates, as determined by "`urn:oasis:names:tc:xacml:x.x:function:type-equal`", shall exist in the result.

`urn:oasis:names:tc:xacml:1.0:function:type-at-least-one-member-of`

This function shall take two arguments that are both a bag of 'type' values. It shall return a "http://www.w3.org/2001/XMLSchema#boolean". The function shall evaluate to "True" if, and only if, at least one element of the first argument is contained in the second argument as determined by "`urn:oasis:names:tc:xacml:x.x:function:type-is-in`".

`urn:oasis:names:tc:xacml:1.0:function:type-union`

This function shall take two arguments that are both a bag of 'type' values. The expression shall return a bag of 'type' such that it contains all elements of both bags. No duplicates, as determined by "`urn:oasis:names:tc:xacml:x.x:function:type-equal`", shall exist in the result.

`urn:oasis:names:tc:xacml:1.0:function:type-subset`

This function shall take two arguments that are both a bag of 'type' values. It shall return a "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument is a subset of the second argument. Each argument shall be considered to have had its duplicates removed, as determined by "`urn:oasis:names:tc:xacml:x.x:function:type-equal`", before the subset calculation.

`urn:oasis:names:tc:xacml:1.0:function:type-set-equals`

This function shall take two arguments that are both a bag of 'type' values. It shall return a "http://www.w3.org/2001/XMLSchema#boolean". It shall return the result of applying "`urn:oasis:names:tc:xacml:1.0:function:and`" to the application of "`urn:oasis:names:tc:xacml:x.x:function:type-subset`" to the first and second arguments and the application of "`urn:oasis:names:tc:xacml:x.x:function:type-subset`" to the second and first arguments.

A.3.12 Higher-order bag functions

This clause discusses functions in XACML that perform operations on bags such that these functions may be applied to the bags in general.

A general-purpose functional language could be beneficial for specifying the semantics of these functions (see Appendix III for an informative example of the use of a functional language).

1) `urn:oasis:names:tc:xacml:1.0:function:any-of`

   This function applies a boolean function between a specific primitive value and a bag of values, and shall return "True" if, and only if, the predicate is "True" for at least one element of the bag.

   This function shall take three arguments. The first argument shall be an `<xacml:Function>` element that names a boolean function that takes two arguments of primitive types. The second argument shall be a value of a primitive data-type. The third argument shall be a bag of a primitive data-type. The expression shall be evaluated as if the function named in the `<xacml:Function>` argument were applied to the second argument and each element of the third argument (the bag) and the results are combined with "`urn:oasis:names:tc:xacml:1.0:function:or`".

2) `urn:oasis:names:tc:xacml:1.0:function:all-of`

   This function applies a boolean function between a specific primitive value and a bag of values, and returns "True" if, and only if, the predicate is "True" for every element of the bag.

   This function shall take three arguments. The first argument shall be an `<xacml:Function>` element that names a boolean function that takes two arguments of primitive types. The second argument shall be a value of a primitive data-type. The third argument shall be a bag of a primitive data-type. The expression shall be evaluated as if the function named in the `<xacml:Function>` argument were applied to the second argument and each element of the third argument (the bag) and the results were combined using "`urn:oasis:names:tc:xacml:1.0:function:and`".

3) `urn:oasis:names:tc:xacml:1.0:function:any-of-any`

   This function applies a boolean function between each element of a bag of values and each element of another bag of values, and returns "True" if, and only if, the predicate is "True" for at least one comparison.

   This function shall take three arguments. The first argument shall be an `<xacml:Function>` element that names a boolean function that takes two arguments of primitive types. The second argument shall be a bag of a primitive data-type. The third argument shall be a bag of a primitive data-type. The expression shall be evaluated as if the function named in the `<xacml:Function>` argument were applied between every element of the second argument and every element of the third argument and the results were combined using "`urn:oasis:names:tc:xacml:1.0:function:or`". The semantics is that the result of the expression shall be "True" if, and only if, the applied predicate is "True" for at least one comparison of elements from the two bags.

4) `urn:oasis:names:tc:xacml:1.0:function:all-of-any`

   This function applies a boolean function between the elements of two bags. The expression shall be "True" if, and only if, the supplied predicate is 'True' between each element of the first bag and any element of the second bag.

   This function shall take three arguments. The first argument shall be an `<xacml:Function>` element that names a boolean function that takes two arguments of primitive types. The second argument shall be a bag of a primitive data-type. The third argument shall be a bag of a primitive data-type. The expression shall be evaluated as if the "`urn:oasis:names:tc:xacml:1.0:function:any-of`" function had been applied to each value of the first bag and the whole of the second bag using the supplied `xacml:Function`, and the results were then combined using "`urn:oasis:names:tc:xacml:1.0:function:and`".

5) `urn:oasis:names:tc:xacml:1.0:function:any-of-all`

   This function applies a boolean function between the elements of two bags. The expression shall be "True" if, and only if, the supplied predicate is "True" between each element of the second bag and any element of the first bag.

   This function shall take three arguments. The first argument shall be an `<xacml:Function>` element that names a boolean function that takes two arguments of primitive types. The second argument shall be a bag of a primitive data-type. The third argument shall be a bag of a primitive data-type. The expression shall be evaluated as if the "`urn:oasis:names:tc:xacml:1.0:function:any-of`" function had been applied to each value of

the second bag and the whole of the first bag using the supplied xacml:Function, and the results were then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

6) urn:oasis:names:tc:xacml:1.0:function:all-of-all

This function applies a boolean function between the elements of two bags. The expression shall be "True" if, and only if, the supplied predicate is "True" between each and every element of the first bag collectively against all the elements of the second bag.

This function shall take three arguments. The first argument shall be an <xacml:Function> element that names a boolean function that takes two arguments of primitive types. The second argument shall be a bag of a primitive data-type. The third argument shall be a bag of a primitive data-type. The expression is evaluated as if the function named in the <xacml:Function> element were applied between every element of the second argument and every element of the third argument and the results were combined using "urn:oasis:names:tc:xacml:1.0:function:and". The semantics is that the result of the expression is "True" if, and only if, the applied predicate is "True" for all elements of the first bag compared to all the elements of the second bag.

7) urn:oasis:names:tc:xacml:1.0:function:map

This function converts a bag of values to another bag of values.

This function shall take two arguments. The first function shall be an <xacml:Function> element naming a function that takes a single argument of a primitive data-type and returns a value of a primitive data-type. The second argument shall be a bag of a primitive data-type. The expression shall be evaluated as if the function named in the <xacml:Function> element were applied to each element in the bag resulting in a bag of the converted value. The result shall be a bag of the primitive data-type that is returned by the function named in the <xacml:Function> element.

A.3.13 Regular-expression-based functions

These functions operate on various types using regular expressions and evaluate to "http://www.w3.org/2001/XMLSchema#boolean".

```
urn:oasis:names:tc:xacml:1.0:function:string-regexp-match
```

This function decides a regular expression match. It shall take two arguments of "http://www.w3.org/2001/XMLSchema#string" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be a general string. The function shall return "True" if, and only if, the second argument matches the value in the regular expression pattern in the first argument. The regular expression syntax shall be that defined in W3C Datatypes:2001, Appendix F, augmented with the following additional pattern expressions and rules:

- X?? matches X no more than once
- X*? matches X any number of times, including zero
- X+? matches X at least once
- X{n}? matches X exactly n times
- X(n,}? matches X at least n times
- X{n,m}? matches X at least n times, but no more than m times

Where one of these additional pattern expressions is used, the regular expression shall match the shortest possible substring of the first argument that is consistent with the pattern.

Except for these additional expressions, the regular expression shall match the longest possible substring of the first argument that is consistent with the pattern.

Except when explicitly anchored to the beginning or end of the string using the pattern characters "^" and "$", respectively, the pattern is considered to match if it matches any substring of the second argument.

All conforming implementations shall support the specified regular expression patterns. A conforming implementation may support additional regular expression patterns.

```
urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match
```

This function decides a regular expression match. It shall take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "http://www.w3.org/2001/XMLSchema#anyURI". It shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be a URI. The function shall

convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match

This function decides a regular expression match. It shall take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". It shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be an IPv4 or IPv6 address. The function shall convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match

This function decides a regular expression match. It shall take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". It shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be a DNS name. The function shall convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match

This function decides a regular expression match. It shall take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". It shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be an RFC 822 name. The function shall convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match

This function decides a regular expression match. It shall take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". It shall return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument shall be a regular expression and the second argument shall be an X.500 directory name. The function shall convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

A.3.14 Special match functions

These functions operate on various types and evaluate to "http://www.w3.org/2001/XMLSchema#boolean" based on the specified standard matching algorithm.

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

This function shall take two arguments of "urn:oasis:names:tc:xacml:2.0:data-type:x500Name" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if, and only if, the first argument matches some terminal sequence of RDNs from the second argument when compared using x500Name-equal.

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

This function shall take two arguments; the first is of data-type "http://www.w3.org/2001/XMLSchema#string" and the second is of data-type "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". This function shall evaluate to "True" if the first argument matches the second argument according to W3C Datatypes:2001, 3.2.1.

An IETF RFC 822 name consists of a local-part followed by "@" followed by a domain-part. The local-part is case-sensitive, while the domain-part (which is usually a DNS name) is not case-sensitive.

The second argument contains a complete rfc822Name. The first argument is a complete or partial rfc822Name used to select appropriate values in the second argument as follows.

In order to match a particular address in the second argument, the first argument must specify the complete mail address to be matched. In order to match any address at a particular domain in the second argument, the first argument must specify only a domain name (usually a DNS name). In order to match any address in a particular domain in the second argument, the first argument must specify the desired domain-part with a leading ".".

A.3.15 XPath-based functions

This clause specifies functions that take XPath expressions for arguments. An XPath expression evaluates to a node-set, which is a set of XML nodes that match the expression. A node or node-set is not in the formal data-type system of XACML. All comparison or other operations on node-sets are performed in isolation of the particular function specified. That is, the XPath expressions in these functions are restricted to the XACML Request context. The `<xacml-context:Request>` element is the context node for every XPath expression. The following functions are defined:

```
urn:oasis:names:tc:xacml:1.0:function:xpath-node-count
```

This function shall take an "http://www.w3.org/2001/XMLSchema#string" as an argument, which shall be interpreted as an XPath expression, and evaluates to an "http://www.w3.org/2001/XMLSchema#integer". The value returned from the function shall be the count of the nodes within the node-set that match the given XPath expression.

```
urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal
```

This function shall take two "http://www.w3.org/2001/XMLSchema#string" arguments, which shall be interpreted as XPath expressions, and shall return an "http://www.w3.org/2001/XMLSchema#boolean". The function shall return "True" if any of the XML nodes in the node-set matched by the first argument equals any of the XML nodes in the node-set matched by the second argument. Two nodes are considered equal if they have the same identity.

```
urn:oasis:names:tc:xacml:1.0:function:xpath-node-match
```

This function shall take two "http://www.w3.org/2001/XMLSchema#string" arguments, which shall be interpreted as XPath expressions and shall return an "http://www.w3.org/2001/XMLSchema#boolean". This function shall evaluate to "True" if one of the following two conditions is satisfied:

1) Any of the XML nodes in the node-set matched by the first argument is equal, to any of the XML nodes in the node-set matched by the second argument;
2) any attribute and element node below any of the XML nodes in the node-set matched by the first argument is equal, to any of the XML nodes in the node-set matched by the second argument.

Two nodes are considered equal if they have the same identity.

NOTE – The first condition is equivalent to "xpath-node-equal", and guarantees that "xpath-node-equal" is a special case of "xpath-node-match".

A.3.16 Extension functions and primitive types

Functions and primitive types are specified by string identifiers allowing for the introduction of functions in addition to those specified by XACML. This approach allows one to extend the XACML module with special functions and special primitive data-types.

In order to preserve the integrity of the XACML evaluation strategy, the result of an extension function shall depend only on the values of its arguments. Global and hidden parameters shall not affect the evaluation of an expression. Functions shall not have side effects, as evaluation order cannot be guaranteed in a standard way.

Annex B

XACML identifiers

This annex defines standard identifiers for commonly used entities.

B.1 XACML namespaces

There are currently two defined XACML namespaces.

Policies are defined using this identifier.

```
urn:oasis:names:tc:xacml:2.0:policy:schema:os
```

Request and response contexts are defined using this identifier.

```
urn:oasis:names:tc:xacml:2.0:context:schema:os
```

B.2 Access subject categories

This identifier indicates the system entity that initiated the access request. That is, the initial entity in a request chain. If subject category is not specified, this is the default value.

```
urn:oasis:names:tc:xacml:1.0:subject-category:access-subject
```

This identifier indicates the system entity that will receive the results of the request (used when it is distinct from the access-subject).

```
urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject
```

This identifier indicates a system entity through which the access request was passed. There may be more than one. No means is provided to specify the order in which they passed the message.

```
urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject
```

This identifier indicates a system entity associated with a local or remote codebase that generated the request. Corresponding subject attributes might include the URL from which it was loaded and/or the identity of the code-signer. There may be more than one. No means is provided to specify the order in which they processed the request.

```
urn:oasis:names:tc:xacml:1.0:subject-category:codebase
```

This identifier indicates a system entity associated with the computer that initiated the ***access*** request. An example would be an IPSEC identity.

```
urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine
```

B.3 Data-types

The following identifiers indicate data-types that are defined in A.2.

```
urn:oasis:names:tc:xacml:2.0:data-types:dayTimeDuration
urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration
urn:oasis:names:tc:xacml:1.0:data-type:x500Name.
urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name
urn:oasis:names:tc:xacml:2.0:data-type:ipAddress
urn:oasis:names:tc:xacml:2.0:data-type:dnsName
```

The following data-type identifiers are defined by W3C DataTypes:2001.

```
http://www.w3.org/2001/XMLSchema#string
http://www.w3.org/2001/XMLSchema#boolean
http://www.w3.org/2001/XMLSchema#integer
http://www.w3.org/2001/XMLSchema#double
http://www.w3.org/2001/XMLSchema#time
http://www.w3.org/2001/XMLSchema#date
http://www.w3.org/2001/XMLSchema#dateTime
```

```
http://www.w3.org/2001/XMLSchema#anyURI
http://www.w3.org/2001/XMLSchema#hexBinary
http://www.w3.org/2001/XMLSchema#base64Binary
```

B.4 Subject attributes

These identifiers indicate attributes of a subject. When used, they shall appear within a `<Subject>` element of the request context. They shall be accessed by means of a `<SubjectAttributeDesignator>` element, or an `<AttributeSelector>` element that points into a `<Subject>` element of the request context.

At most one of each of these attributes is associated with each subject. Each attribute associated with authentication included within a single `<Subject>` element relates to the same authentication event.

This identifier indicates the name of the subject. The default format is "http://www.w3.org/2001/XMLSchema#string". To indicate other formats, use the `DataType` attributes listed in B.3.

```
urn:oasis:names:tc:xacml:1.0:subject:subject-id
```

This identifier indicates the subject category. "access-subject" is the default value.

```
urn:oasis:names:tc:xacml:1.0:subject-category
```

This identifier indicates the security domain of the subject. It identifies the administrator and policy that manages the name-space in which the subject id is administered.

```
urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier
```

This identifier indicates a public key used to confirm the subject's identity.

```
urn:oasis:names:tc:xacml:1.0:subject:key-info
```

This identifier indicates the time at which the subject was authenticated.

```
urn:oasis:names:tc:xacml:1.0:subject:authentication-time
```

This identifier indicates the method used to authenticate the subject.

```
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:authentication-method
```

This identifier indicates the time at which the subject initiated the access request, according to the PEP.

```
urn:oasis:names:tc:xacml:1.0:subject:request-time
```

This identifier indicates the time at which the subject's current session began, according to the PEP.

```
urn:oasis:names:tc:xacml:1.0:subject:session-start-time
```

The following identifiers indicate the location where authentication credentials were activated. They are intended to support the corresponding entities from the SAML authentication statement.

This identifier indicates that the location is expressed as an IP address.

```
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address
```

The corresponding attribute shall be of data-type "http://www.w3.org/2001/XMLSchema#string".

This identifier indicates that the location is expressed as a DNS name.

```
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name
```

The corresponding attribute shall be of data-type "http://www.w3.org/2001/XMLSchema#string".

Where a suitable attribute is already defined in LDAP, the XACML identifier shall be formed by adding the attribute name to the URI of the IETF LDAP RFC (see IETF RFC 2256). For example, the attribute name for the `userPassword` defined in the IETF RFC 2256 shall be:

```
http://www.ietf.org/rfc/rfc2256.txt#userPassword
```

B.5 Resource attributes

These identifiers indicate attributes of the resource. The corresponding attributes may appear in the <Resource> element of the request context and be accessed by means of a <ResourceAttributeDesignator> element, or by an <AttributeSelector> element that points into the <Resource> element of the request context.

This attribute identifies the resource to which access is requested. If an <xacml-context:ResourceContent> element is provided, then the resource to which access is requested shall be all or a portion of the resource supplied in the <xacml-context:ResourceContent> element.

```
urn:oasis:names:tc:xacml:1.0:resource:resource-id
```

This attribute identifies the namespace of the top element of the contents of the <xacml-context:ResourceContent> element. In the case where the resource content is supplied in the request context and the resource namespace is defined in the resource, the PDP shall confirm that the namespace defined by this attribute is the same as that defined in the resource. The type of the corresponding attribute shall be "http://www.w3.org/2001/XMLSchema#anyURI".

```
urn:oasis:names:tc:xacml:2.0:resource:target-namespace
```

B.6 Action attributes

These identifiers indicate attributes of the action being requested. When used, they shall appear within the <Action> element of the request context. They shall be accessed by means of an <ActionAttributeDesignator> element, or an <AttributeSelector> element that points into the <Action> element of the request context.

This attribute identifies the action for which access is requested.

```
urn:oasis:names:tc:xacml:1.0:action:action-id
```

Where the action is implicit, the value of the action-id attribute shall be:

```
urn:oasis:names:tc:xacml:1.0:action:implied-action
```

This attribute identifies the namespace in which the action-id attribute is defined.

```
urn:oasis:names:tc:xacml:1.0:action:action-namespace
```

B.7 Environment attributes

These identifiers indicate attributes of the environment within which the decision request is to be evaluated. When used in the decision request, they shall appear in the <Environment> element of the request context. They shall be accessed by means of an <EnvironmentAttributeDesignator> element, or an <AttributeSelector> element that points into the <Environment> element of the request context.

This identifier indicates the current time at the context handler. In practice it is the time at which the request context was created. For this reason, if these identifiers appear in multiple places within a <Policy> or <PolicySet>, then the same value shall be assigned to each occurrence in the evaluation procedure, regardless of how much time elapses between the processing of the occurrences.

```
urn:oasis:names:tc:xacml:1.0:environment:current-time
```

The corresponding attribute shall be of data-type "http://www.w3.org/2001/XMLSchema#time".

```
urn:oasis:names:tc:xacml:1.0:environment:current-date
```

The corresponding attribute shall be of data-type "http://www.w3.org/2001/XMLSchema#date".

```
urn:oasis:names:tc:xacml:1.0:environment:current-dateTime
```

The corresponding attribute shall be of data-type "http://www.w3.org/2001/XMLSchema#dateTime".

B.8 Status codes

The following status code values are defined.

This identifier indicates success.

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

This identifier indicates that all the attributes necessary to make a policy decision were not available.

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

This identifier indicates that some attribute value contained a syntax error, such as a letter in a numeric field.

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

This identifier indicates that an error occurred during policy evaluation. An example would be division by zero.

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

B.9 Combining algorithms

The deny-overrides rule-combining algorithm has the following value for the `ruleCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides
```

The deny-overrides policy-combining algorithm has the following value for the `policyCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides
```

The permit-overrides rule-combining algorithm has the following value for the `ruleCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides
```

The permit-overrides policy-combining algorithm has the following value for the `policyCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides
```

The first-applicable rule-combining algorithm has the following value for the `ruleCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable
```

The first-applicable policy-combining algorithm has the following value for the `policyCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable
```

The only-one-applicable-policy policy-combining algorithm has the following value for the `policyCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable
```

The ordered-deny-overrides rule-combining algorithm has the following value for the `ruleCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides
```

The ordered-deny-overrides policy-combining algorithm has the following value for the `policyCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides
```

The ordered-permit-overrides rule-combining algorithm has the following value for the `ruleCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides
```

The ordered-permit-overrides policy-combining algorithm has the following value for the `policyCombiningAlgId` attribute:

```
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-
overrides
```

Annex C

Combining algorithms

This annex contains a description of the rule- and policy-combining algorithms specified by XACML.

C.1 Deny-overrides

C.1.1 This clause defines the "Deny-overrides" rule-combining algorithm of a policy.

In the entire set of rules in the policy, if any rule evaluates to "Deny", then the result of the rule combination shall be "Deny". If any rule evaluates to "Permit" and all other rules evaluate to "NotApplicable", then the result of the rule combination shall be "Permit". In other words, "Deny" takes precedence, regardless of the result of evaluating any of the other rules in the combination. If all rules are found to be "NotApplicable" to the decision request, then the rule combination shall evaluate to "NotApplicable".

If an error occurs while evaluating the target or condition of a rule that contains an effect value of "Deny" then the evaluation shall continue to evaluate subsequent rules, looking for a result of "Deny". If no other rule evaluates to "Deny", then the combination shall evaluate to "Indeterminate", with the appropriate error status.

If at least one rule evaluates to "Permit", all other rules that do not have evaluation errors evaluate to "Permit" or "NotApplicable" and all rules that do have evaluation errors contain effects of "Permit", then the result of the combination shall be "Permit".

The following pseudo-code represents the evaluation strategy of this rule-combining algorithm.

```
Decision denyOverridesRuleCombiningAlgorithm(Rule rule[])
{
   Boolean atLeastOneError  = false;
   Boolean potentialDeny    = false;
   Boolean atLeastOnePermit = false;
   for( i=0 ; i < lengthOf(rules) ; i++ )
   {
          Decision decision = evaluate(rule[i]);
          if (decision == Deny)
          {
                return Deny;
          }
          if (decision == Permit)
          {
                atLeastOnePermit = true;
                continue;
          }
          if (decision == NotApplicable)
          {
                continue;
          }
          if (decision == Indeterminate)
          {
                atLeastOneError = true;

                if (effect(rule[i]) == Deny)
                {
                      potentialDeny = true;
                }
                continue;
          }
   }
   if (potentialDeny)
   {
          return Indeterminate;
   }
   if (atLeastOnePermit)
   {
          return Permit;
   }
   if (atLeastOneError)
```

```
    {
          return Indeterminate;
    }
    return NotApplicable;
}
```

C.1.2 This clause defines the "Deny-overrides" policy-combining algorithm of a policy set.

In the entire set of policies in the policy set, if any policy evaluates to "Deny", then the result of the policy combination shall be "Deny". In other words, "Deny" takes precedence, regardless of the result of evaluating any of the other policies in the policy set. If all policies are found to be "NotApplicable" to the decision request, then the policy set shall evaluate to "NotApplicable".

If an error occurs while evaluating the target of a policy, or a reference to a policy is considered invalid or the policy evaluation results in "Indeterminate", then the policy set shall evaluate to "Deny".

The following pseudo-code represents the evaluation strategy of this policy-combining algorithm.

```
Decision denyOverridesPolicyCombiningAlgorithm(Policy policy[])
{
    Boolean atLeastOnePermit = false;
    for( i=0 ; i < lengthOf(policy) ; i++ )
    {
          Decision decision = evaluate(policy[i]);
          if (decision == Deny)
          {
                return Deny;
          }
          if (decision == Permit)
          {
                atLeastOnePermit = true;
                continue;
          }
          if (decision == NotApplicable)
          {
                continue;
          }
          if (decision == Indeterminate)
          {
                return Deny;
          }
    }
    if (atLeastOnePermit)
    {
          return Permit;
    }
    return NotApplicable;
}
```

Obligations of the individual policies shall be combined as described in 7.6.14.

C.2 Ordered-deny-overrides

The following paragraph defines the "Ordered-deny-overrides" rule-combining algorithm of a policy.

The behaviour of this algorithm is identical to that of the Deny-overrides rule-combining algorithm with one exception. The order in which the collection of rules is evaluated shall match the order as listed in the policy.

The following paragraph defines the "Ordered-deny-overrides" policy-combining algorithm of a policy set.

The behaviour of this algorithm is identical to that of the Deny-overrides policy-combining algorithm with one exception. The order in which the collection of policies is evaluated shall match the order as listed in the policy set.

C.3 Permit-overrides

C.3.1 This clause defines the "Permit-overrides" rule-combining algorithm of a policy.

In the entire set of rules in the policy, if any rule evaluates to "Permit", then the result of the rule combination shall be "Permit". If any rule evaluates to "Deny" and all other rules evaluate to "NotApplicable", then the policy shall evaluate to "Deny". In other words, "Permit" takes precedence, regardless of the result of evaluating any of the other rules in the

policy. If all rules are found to be "NotApplicable" to the decision request, then the policy shall evaluate to "NotApplicable".

If an error occurs while evaluating the target or condition of a rule that contains an effect of "Permit" then the evaluation shall continue looking for a result of "Permit". If no other rule evaluates to "Permit", then the policy shall evaluate to "Indeterminate", with the appropriate error status.

If at least one rule evaluates to "Deny", all other rules that do not have evaluation errors evaluate to "Deny" or "NotApplicable" and all rules that do have evaluation errors contain an effect value of "Deny", then the policy shall evaluate to "Deny".

The following pseudo-code represents the evaluation strategy of this rule-combining algorithm.

```
Decision permitOverridesRuleCombiningAlgorithm(Rule rule[])
{
   Boolean atLeastOneError  = false;
   Boolean potentialPermit  = false;
   Boolean atLeastOneDeny   = false;
   for( i=0 ; i < lengthOf(rule) ; i++ )
   {
          Decision decision = evaluate(rule[i]);
          if (decision == Deny)
          {
                atLeastOneDeny = true;
                continue;
          }
          if (decision == Permit)
          {
                return Permit;
          }
          if (decision == NotApplicable)
          {
                continue;
          }
          if (decision == Indeterminate)
          {
                atLeastOneError = true;

                if (effect(rule[i]) == Permit)
                {
                      potentialPermit = true;
                }
                continue;
          }
   }
   if (potentialPermit)
   {
          return Indeterminate;
   }
   if (atLeastOneDeny)
   {
          return Deny;
   }
   if (atLeastOneError)
   {
          return Indeterminate;
   }
   return NotApplicable;
}
```

C.3.2 This clause defines the "Permit-overrides" policy-combining algorithm of a policy set.

In the entire set of policies in the policy set, if any policy evaluates to "Permit", then the result of the policy combination shall be "Permit". In other words, "Permit" takes precedence, regardless of the result of evaluating any of the other policies in the policy set. If all policies are found to be "NotApplicable" to the decision request, then the policy set shall evaluate to "NotApplicable".

If an error occurs while evaluating the target of a policy, a reference to a policy is considered invalid or the policy evaluation results in "Indeterminate", then the policy set shall evaluate to "Indeterminate", with the appropriate error status, provided no other policies evaluate to "Permit" or "Deny".

The following pseudo-code represents the evaluation strategy of this policy-combining algorithm.

```
Decision permitOverridesPolicyCombiningAlgorithm(Policy policy[])
{
   Boolean atLeastOneError = false;
   Boolean atLeastOneDeny  = false;
   for( i=0 ; i < lengthOf(policy) ; i++ )
   {
          Decision decision = evaluate(policy[i]);
          if (decision == Deny)
          {
                atLeastOneDeny = true;
                continue;
          }
          if (decision == Permit)
          {
                return Permit;
          }
          if (decision == NotApplicable)
          {
                continue;
          }
          if (decision == Indeterminate)
          {
                atLeastOneError = true;
                continue;
          }
   }
   if (atLeastOneDeny)
   {
          return Deny;
   }
   if (atLeastOneError)
   {
          return Indeterminate;
   }
   return NotApplicable;
}
```

Obligations of the individual policies shall be combined as described in 7.6.14.

C.4 Ordered-permit-overrides

The following paragraph defines the "Ordered-permit-overrides" rule-combining algorithm of a policy.

The behaviour of this algorithm is identical to that of the Permit-overrides rule-combining algorithm with one exception. The order in which the collection of rules is evaluated shall match the order as listed in the policy.

The following paragraph defines the "Ordered-permit-overrides" policy-combining algorithm of a policy set.

The behaviour of this algorithm is identical to that of the Permit-overrides policy-combining algorithm with one exception. The order in which the collection of policies is evaluated shall match the order as listed in the policy set.

C.5 First-applicable

C.5.1 This clause defines the "First-Applicable " rule-combining algorithm of a policy.

Each rule shall be evaluated in the order in which it is listed in the policy. For a particular rule, if the target matches and the condition evaluates to "True", then the evaluation of the policy shall halt and the corresponding effect of the rule shall be the result of the evaluation of the policy (i.e., "Permit" or "Deny"). For a particular rule selected in the evaluation, if the target evaluates to "False" or the condition evaluates to "False", then the next rule in the order shall be evaluated. If no further rule in the order exists, then the policy shall evaluate to "NotApplicable".

If an error occurs while evaluating the target or condition of a rule, then the evaluation shall halt, and the policy shall evaluate to "Indeterminate", with the appropriate error status.

The following pseudo-code represents the evaluation strategy of this rule-combining algorithm.

```
Decision firstApplicableEffectRuleCombiningAlgorithm(Rule rule[])
{
   for( i = 0 ; i < lengthOf(rule) ; i++ )
   {
          Decision decision = evaluate(rule[i]);
          if (decision == Deny)
          {
                return Deny;
          }
          if (decision == Permit)
          {
                return Permit;
          }
          if (decision == NotApplicable)
          {
                continue;
          }
          if (decision == Indeterminate)
          {
                return Indeterminate;
          }
   }
   return NotApplicable;
}
```

C.5.2 This clause defines the "First-applicable" policy-combining algorithm of a policy set.

Each policy is evaluated in the order that it appears in the policy set. For a particular policy, if the target evaluates to "True" and the policy evaluates to a determinate value of "Permit" or "Deny", then the evaluation shall halt and the policy set shall evaluate to the effect value of that policy. For a particular policy, if the target evaluate to "False", or the policy evaluates to "NotApplicable", then the next policy in the order shall be evaluated. If no further policy exists in the order, then the policy set shall evaluate to "NotApplicable".

If an error were to occur when evaluating the target, or when evaluating a specific policy, the reference to the policy is considered invalid, or the policy itself evaluates to "Indeterminate", then the evaluation of the policy-combining algorithm shall halt, and the policy set shall evaluate to "Indeterminate" with an appropriate error status.

The following pseudo-code represents the evaluation strategy of this policy-combination algorithm.

```
Decision firstApplicableEffectPolicyCombiningAlgorithm(Policy policy[])
{
    for( i = 0 ; i < lengthOf(policy) ; i++ )
    {
        Decision decision = evaluate(policy[i]);
        if(decision == Deny)
        {
            return Deny;
        }
        if(decision == Permit)
        {
            return Permit;
        }
        if (decision == NotApplicable)
        {
            continue;
        }
        if (decision == Indeterminate)
        {
            return Indeterminate;
        }
    }
    return NotApplicable;
}
```

Obligations of the individual policies shall be combined as described in 7.6.14.

C.6 Only-one-applicable

This clause defines the "Only-one-applicable" policy-combining algorithm of a policy set.

In the entire set of policies in the policy set, if no policy is considered applicable by virtue of its target, then the result of the policy combination algorithm shall be "NotApplicable". If more than one policy is considered applicable by virtue of its target, then the result of the policy combination algorithm shall be "Indeterminate".

If only one policy is considered applicable by evaluation of its target, then the result of the policy-combining algorithm shall be the result of evaluating the policy.

If an error occurs while evaluating the target of a policy, or a reference to a policy is considered invalid or the policy evaluation results in "Indeterminate, then the policy set shall evaluate to "Indeterminate", with the appropriate error status.

The following pseudo-code represents the evaluation strategy of this policy combining algorithm.

```
Decision onlyOneApplicablePolicyPolicyCombiningAlogrithm(Policy policy[])
{
  Boolean          atLeastOne     = false;
  Policy           selectedPolicy = null;
  ApplicableResult appResult;

  for ( i = 0; i < lengthOf(policy) ; i++ )
  {
     appResult = isApplicable(policy[I]);

     if ( appResult == Indeterminate )
     {
         return Indeterminate;
     }
     if( appResult == Applicable )
     {
         if ( atLeastOne )
         {
             return Indeterminate;
         }
         else
         {
             atLeastOne     = true;
             selectedPolicy = policy[i];
         }
     }
     if ( appResult == NotApplicable )
     {
         continue;
     }
  }
  if ( atLeastOne )
  {
      return evaluate(selectedPolicy);
  }
  else
  {
      return NotApplicable;
  }
}
```

Annex D

XACML schema

D.1 XACML context schema

This clause provides XACML context schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
   schemaLocation="http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-
schema-os.xsd"/>
   <!--   -->
   <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>
   <!-- -->
   <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>
   <!-- -->
   <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>
   <!-- -->
   <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>
   <!-- -->
   <xs:element name="ResourceContent" type="xacml-context:ResourceContentType"/>
   <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>
   <!-- -->
   <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>
   <!-- -->
   <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>
   <!-- -->
   <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>
   <!-- -->
   <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>
   <!-- -->
   <xs:element name="Result" type="xacml-context:ResultType"/>
   <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>
   <!-- -->
   <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>
   <!-- -->
   <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>
   <!-- -->
   <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>
   <!-- -->
   <xs:element name="StatusMessage" type="xs:string"/>
   <!-- -->
   <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>
   <!-- -->
   <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>
   <!-- -->
</xs:schema>
```

D.2 Policy schema

This clause provides XACML policy schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <!-- -->
   <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>
   <!-- -->
   <xs:element name="CombinerParameters" type="xacml:CombinerParametersType"/>
   <xs:complexType name="CombinerParametersType">
          <xs:sequence>
                <xs:element ref="xacml:CombinerParameter" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
   </xs:complexType>
   <!-- -->
   <xs:element name="CombinerParameter" type="xacml:CombinerParameterType"/>
   <xs:complexType name="CombinerParameterType">
          <xs:sequence>
                <xs:element ref="xacml:AttributeValue"/>
          </xs:sequence>
          <xs:attribute name="ParameterName" type="xs:string" use="required"/>
   </xs:complexType>
   <!-- -->
   <xs:element name="RuleCombinerParameters"
type="xacml:RuleCombinerParametersType"/>
```

```
   <xs:complexType name="RuleCombinerParametersType">
          <xs:complexContent>
                 <xs:extension base="xacml:CombinerParametersType">
                        <xs:attribute name="RuleIdRef" type="xs:string"
use="required"/>
                 </xs:extension>
          </xs:complexContent>
   </xs:complexType>
   <!-- -->
   <xs:element name="PolicyCombinerParameters"
type="xacml:PolicyCombinerParametersType"/>
   <xs:complexType name="PolicyCombinerParametersType">
          <xs:complexContent>
                 <xs:extension base="xacml:CombinerParametersType">
                        <xs:attribute name="PolicyIdRef" type="xs:anyURI"
use="required"/>
                 </xs:extension>
          </xs:complexContent>
   </xs:complexType>
   <!-- -->
   <xs:element name="PolicySetCombinerParameters"
type="xacml:PolicySetCombinerParametersType"/>
   <xs:complexType name="PolicySetCombinerParametersType">
          <xs:complexContent>
                 <xs:extension base="xacml:CombinerParametersType">
                        <xs:attribute name="PolicySetIdRef" type="xs:anyURI"
use="required"/>
                 </xs:extension>
          </xs:complexContent>
   </xs:complexType>
   <!-- -->
   <xs:element name="PolicySetIdReference" type="xacml:IdReferenceType"/>
   <xs:element name="PolicyIdReference" type="xacml:IdReferenceType"/>
   <!-- -->
   <xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
   <xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
   <xs:complexType name="DefaultsType">
          <xs:sequence>
                 <xs:choice>
                        <xs:element ref="xacml:XPathVersion"/>
                 </xs:choice>
          </xs:sequence>
   </xs:complexType>
   <!-- -->
   <xs:element name="XPathVersion" type="xs:anyURI"/>
   <!-- -->
   <xs:complexType name="IdReferenceType">
          <xs:simpleContent>
                 <xs:extension base="xs:anyURI">
                        <xs:attribute name="Version" type="xacml:VersionMatchType"
use="optional"/>
                        <xs:attribute name="EarliestVersion"
type="xacml:VersionMatchType" use="optional"/>
                        <xs:attribute name="LatestVersion"
type="xacml:VersionMatchType" use="optional"/>
                 </xs:extension>
          </xs:simpleContent>
   </xs:complexType>
   <!-- -->
   <xs:simpleType name="VersionType">
          <xs:restriction base="xs:string">
                 <xs:pattern value="(\d+\.)*\d+"/>
          </xs:restriction>
   </xs:simpleType>
   <!-- -->
   <xs:simpleType name="VersionMatchType">
          <xs:restriction base="xs:string">
                 <xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)"/>
          </xs:restriction>
   </xs:simpleType>
   <!-- -->
```

```
<xs:element name="Policy" type="xacml:PolicyType"/>
<xs:complexType name="PolicyType">
      <xs:sequence>
            <xs:element ref="xacml:Description" minOccurs="0"/>
            <xs:element ref="xacml: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>
<!-- -->
<xs:element name="Description" type="xs:string"/>
<!-- -->
<xs:element name="Rule" type="xacml:RuleType"/>
<xs:complexType name="RuleType">
      <xs:sequence>
            <xs:element ref="xacml:Description" minOccurs="0"/>
            <xs:element ref="xacml:Target" minOccurs="0"/>
            <xs:element ref="xacml:Condition" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="RuleId" type="xs:string" use="required"/>
      <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
</xs:complexType>
<!-- -->
<xs:simpleType name="EffectType">
      <xs:restriction base="xs:string">
            <xs:enumeration value="Permit"/>
            <xs:enumeration value="Deny"/>
      </xs:restriction>
</xs:simpleType>
<!-- -->
<xs:element name="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>
<!-- -->
<xs:element name="Subjects" type="xacml:SubjectsType"/>
<xs:complexType name="SubjectsType">
      <xs:sequence>
            <xs:element ref="xacml:Subject" maxOccurs="unbounded"/>
      </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Subject" type="xacml:SubjectType"/>
<xs:complexType name="SubjectType">
      <xs:sequence>
            <xs:element ref="xacml:SubjectMatch" maxOccurs="unbounded"/>
      </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Resources" type="xacml:ResourcesType"/>
<xs:complexType name="ResourcesType">
      <xs:sequence>
            <xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
      </xs:sequence>
</xs:complexType>
<!-- -->
```

```
<xs:element name="Resource" type="xacml:ResourceType"/>
<xs:complexType name="ResourceType">
       <xs:sequence>
              <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
       </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Actions" type="xacml:ActionsType"/>
<xs:complexType name="ActionsType">
       <xs:sequence>
              <xs:element ref="xacml:Action" maxOccurs="unbounded"/>
       </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Action" type="xacml:ActionType"/>
<xs:complexType name="ActionType">
       <xs:sequence>
              <xs:element ref="xacml:ActionMatch" maxOccurs="unbounded"/>
       </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Environments" type="xacml:EnvironmentsType"/>
<xs:complexType name="EnvironmentsType">
       <xs:sequence>
              <xs:element ref="xacml:Environment" maxOccurs="unbounded"/>
       </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Environment" type="xacml:EnvironmentType"/>
<xs:complexType name="EnvironmentType">
       <xs:sequence>
              <xs:element ref="xacml:EnvironmentMatch" maxOccurs="unbounded"/>
       </xs:sequence>
</xs:complexType>
<!-- -->
<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>
<!-- -->
<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>
<!-- -->
<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>
<!-- -->
```

```
   <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>
   <!-- -->
   <xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>
   <xs:complexType name="VariableDefinitionType">
          <xs:sequence>
                 <xs:element ref="xacml:Expression"/>
          </xs:sequence>
          <xs:attribute name="VariableId" type="xs:string" use="required"/>
   </xs:complexType>
   <!-- -->
   <xs:element name="Expression" type="xacml:ExpressionType" abstract="true"/>
   <xs:complexType name="ExpressionType" abstract="true"/>
   <!-- -->
   <xs:element name="VariableReference"
   type="xacml:VariableReferenceType" substitutionGroup="xacml:Expression"/>
   <xs:complexType name="VariableReferenceType">
          <xs:complexContent>
                 <xs:extension base="xacml:ExpressionType">
                        <xs:attribute name="VariableId" type="xs:string"
use="required"/>
                 </xs:extension>
          </xs:complexContent>
   </xs:complexType>
   <!-- -->
   <xs:element name="AttributeSelector"
   type="xacml:AttributeSelectorType" substitutionGroup="xacml:Expression"/>
   <xs:complexType name="AttributeSelectorType">
          <xs:complexContent>
                 <xs:extension base="xacml:ExpressionType">
                        <xs:attribute name="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>
   <!-- -->
   <xs:element name="ResourceAttributeDesignator"
   type="xacml:AttributeDesignatorType" substitutionGroup="xacml:Expression"/>
   <xs:element name="ActionAttributeDesignator"
   type="xacml:AttributeDesignatorType" substitutionGroup="xacml:Expression"/>
   <xs:element name="EnvironmentAttributeDesignator"
   type="xacml:AttributeDesignatorType" substitutionGroup="xacml:Expression"/>
   <!-- -->
   <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>
   <!-- -->
   <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>
   <!-- -->
   <xs:element name="AttributeValue" type="xacml:AttributeValueType"
          substitutionGroup="xacml:Expression"/>
   <xs:complexType name="AttributeValueType" mixed="true">
          <xs:complexContent mixed="true">
                 <xs:extension base="xacml:ExpressionType">
                        <xs:sequence>
                               <xs:any namespace="##any" processContents="lax"
minOccurs="0"
                                      maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attribute name="DataType" type="xs:anyURI"
use="required"/>
                        <xs:anyAttribute namespace="##any" processContents="lax"/>
                 </xs:extension>
          </xs:complexContent>
   </xs:complexType>
   <!-- -->
   <xs:element name="Function" type="xacml:FunctionType"
substitutionGroup="xacml:Expression"/>
   <xs:complexType name="FunctionType">
          <xs:complexContent>
                 <xs:extension base="xacml:ExpressionType">
                        <xs:attribute name="FunctionId" type="xs:anyURI"
use="required"/>
                 </xs:extension>
          </xs:complexContent>
   </xs:complexType>
   <!-- -->
   <xs:element name="Condition" type="xacml:ConditionType"/>
   <xs:complexType name="ConditionType">
          <xs:sequence>
                 <xs:element ref="xacml:Expression"/>
          </xs:sequence>
   </xs:complexType>
   <!-- -->
   <xs:element name="Apply" type="xacml:ApplyType"
substitutionGroup="xacml:Expression"/>
   <xs:complexType name="ApplyType">
          <xs:complexContent>
                 <xs:extension base="xacml:ExpressionType">
                        <xs:sequence>
                               <xs:element ref="xacml:Expression" minOccurs="0"
maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attribute name="FunctionId" type="xs:anyURI"
use="required"/>
                 </xs:extension>
          </xs:complexContent>
   </xs:complexType>
   <!-- -->
   <xs:element name="Obligations" type="xacml:ObligationsType"/>
   <xs:complexType name="ObligationsType">
          <xs:sequence>
                 <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
          </xs:sequence>
   </xs:complexType>
   <!-- -->
   <xs:element name="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>
   <!-- -->
   <xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
   <xs:complexType name="AttributeAssignmentType" mixed="true">
          <xs:complexContent mixed="true">
                <xs:extension base="xacml:AttributeValueType">
                      <xs:attribute name="AttributeId" type="xs:anyURI"
use="required"/>
                </xs:extension>
          </xs:complexContent>
   </xs:complexType>
   <!-- -->
</xs:schema>
```

D.3 XACML SAML protocol schema

This clause provides XACML SAML protocol schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="urn:oasis:xacml:2.0:saml:protocol:schema:os"
    xmlns:xs="http://www.23.org/2001/XMLSchema"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
    xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
  <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
      schemaLocation="http://www.oasis-
open.org/committees/tc_home.php?wg_abbrev=security"/>
  <xs:import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
      schemaLocation="http://www.oasis-
open.org/committees/tc_home.php?wg_abbrev=security"/>
  <xs:import namespace="urn:oasis:names:tc:xacml:2.0:context:schema:os"
      schemaLocation="http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-
context-schema-os.xsd"/>
  <xs:import namespace="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
      schemaLocation="http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-
policy-schema-os.xsd"/>
  <xs:annotation>
    <xs:documentation>
        Document identifier: access_control-xacml-2.0-saml-protocol-schema-os.xsd
        Location: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-
protocol-schema-os.xsd
    </xs:documentation>
  </xs:annotation>
  <!--    -->
  <xs:element name="XACMLAuthzDecisionQuery"
           type="XACMLAuthzDecisionQueryType"/>
  <xs:complexType name="XACMLAuthzDecisionQueryType">
    <xs:complexContent>
      <xs:extension base="samlp:RequestAbstractType">
        <xs:sequence>
          <xs:element ref="xacml-context:Request"/>
        </xs:sequence>
        <xs:attribute name="InputContextOnly"
                      type="boolean"
                      use="optional"
                      default="false"/>
        <xs:attribute name="ReturnContext"
                      type="boolean"
```

```
                    use="optional"
                    default="false"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--    -->
  <xs:element name="XACMLPolicyQuery"
           type="XACMLPolicyQueryType"/>
  <xs:complexType name="XACMLPolicyQueryType">
    <xs:complexContent>
      <xs:extension base="samlp:RequestAbstractType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">>
          <xs:element ref="xacml-context:Request"/>
          <xs:element ref="xacml:Target"/>
          <xs:element ref="xacml:PolicySetIdReference"/>
          <xs:element ref="xacml:PolicyIdReference"/>
        </xs:choice>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</schema>
```

D.4 XACML SAML assertion schema

This clause provides XACML SAML assertion schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="urn:oasis:xacml:2.0:saml:assertion:schema:os"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:xs="http://www.23.org/2001/XMLSchema"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
    xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
  <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
      schemaLocation="http://www.oasis-
open.org/committees/tc_home.php?wg_abbrev=security"/>
  <xs:import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
      schemaLocation="http://www.oasis-
open.org/committees/tc_home.php?wg_abbrev=security"/>
  <xs:import namespace="urn:oasis:names:tc:xacml:2.0:context:schema:os"
      schemaLocation="http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-
context-schema-os.xsd"/>
  <xs:import namespace="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
      schemaLocation="http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-
policy-schema-os.xsd"/>
  <xs:annotation>
    <xs:documentation>
        Document identifier: access_control-xacml-2.0-saml-assertion-schema-cd-02.xsd
        Location: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-
assertion-schema-cd-os.xsd
    </xs:documentation>
  </xs:annotation>
  <!--    -->
  <xs:element name="XACMLAuthzDecisionStatement"
           type="XACMLAuthzDecisionStatementType"/>
  <xs:complexType name="XACMLAuthzDecisionStatementType">
    <xs:complexContent>
      <xs:extension base="samlp:StatementAbstractType">
        <xs:sequence>
          <xs:element ref="xacml-context:Response"/>
          <xs:element ref="xacml-context:Request"  MinOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
```

```
  </xs:complexType>
  <!--    -->
  <xs:element name="XACMLPolicyStatement"
           type="XACMLPolicyStatementType"/>
  <xs:complexType name="XACMLPolicyStatementType">
    <xs:complexContent>
      <xs:extension base="samlp:StatementAbstractType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">>
          <xs:element ref="xacml:Policy"/>
          <xs:element ref="xacml:PolicySet"/>
        </xs:choice>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</schema>
```

Appendix I

Security considerations

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

I.1 Threat model

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

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

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

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

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

I.1.1 Unauthorized disclosure

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

Unauthorized disclosure is addressed by confidentiality safeguards.

I.1.2 Message replay

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

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

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

I.1.3 Message insertion

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

The solution to a message insertion attack is to use mutual authentication and message sequence integrity safeguards between the actors. It should be noted that just using SSL mutual authentication is not sufficient. This only proves that the other party is the one identified by the subject of the X.509 certificate. In order to be effective, it is necessary to confirm that the certificate subject is authorized to send the message.

I.1.4 Message deletion

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

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

I.1.5 Message modification

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

I.1.6 NotApplicable results

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

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

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

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

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

I.1.7 Negative rules

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

A common use for negative rules is to deny access to an individual or subgroup when their membership in a larger group would otherwise permit them access. For example, we might want to write a rule that allows all Vice Presidents to see the unpublished financial data, except for Joe, who is only a Ceremonial Vice President and can be indiscreet in his communications. If we have complete control over the administration of subject attributes, a superior approach would be to define "Vice President" and "Ceremonial Vice President" as distinct groups and then define rules accordingly. However, in some environments this approach may not be feasible. (It is worth noting in passing that, generally speaking, referring to individuals in rules does not scale well. Generally, shared attributes are preferred.)

If not used with care, negative rules can lead to policy violation in two common cases. They are: when attributes are suppressed and when the base group changes. An example of suppressed attributes would be if we have a policy that access should be permitted, unless the subject is a credit risk. If it is possible that the attribute of being a credit risk may

be unknown to the PDP for some reason, then unauthorized access may result. In some environments, the subject may be able to suppress the publication of attributes by the application of privacy controls, or the server or repository that contains the information may be unavailable for accidental or intentional reasons.

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

I.2 Safeguards

I.2.1 Authentication

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

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

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

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

I.2.2 Policy administration

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

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

I.2.3 Confidentiality

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

I.2.3.1 Communication confidentiality

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

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

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

I.2.3.2 Statement level confidentiality

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

W3C Encryption:2002 can be used to encrypt all or parts of an XML document. W3C Encryption:2002 is recommended for use with XACML.

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

I.2.4 Policy integrity

The XACML policy, used by the PDP to evaluate the request context, is the heart of the system. Therefore, maintaining its integrity is essential. There are two aspects to maintaining the integrity of the policy. One is to ensure that `<Policy>` elements have not been altered since they were originally created by the PAP. The other is to ensure that `<Policy>` elements have not been inserted or deleted from the set of policies.

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

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

I.2.5 Policy identifiers

Since policies can be referenced by their identifiers, it is the responsibility of the PAP to ensure that these are unique. Confusion between identifiers could lead to misidentification of the applicable policy. This Recommendation is silent on whether a PAP must generate a new identifier when a policy is modified or may use the same identifier in the modified policy. This is a matter of administrative practice. However, care must be taken in either case. If the identifier is reused, there is a danger that other policies or policy sets that reference it may be adversely affected. Conversely, if a new identifier is used, these other policies may continue to use the prior policy, unless it is deleted. In either case, the results may not be what the policy administrator intends.

I.2.6 Trust model

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

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

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

I.2.7 Privacy

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

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

Appendix II

XACML examples

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

II.1 Example one

This clause contains the first example.

II.1.1 Example policy

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

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

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

```
[a02]  <?xml version="1.0" encoding="UTF-8"?>
[a03]  <Policy
[a04]  xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
[a05]  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a06]  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-
os.xsd"
[a07]  PolicyId="urn:oasis:names:tc:example:SimplePolicy1"
[a08]  RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-
overrides">
[a09]   <Description>
[a10]    Medi Corp access control policy
[a11]   </Description>
[a12]   <Target/>
[a13]   <Rule
[a14]   RuleId= "urn:oasis:names:tc:xacml:2.0:example:SimpleRule1"
[a15]   Effect="Permit">
[a16]    <Description>
[a17]     Any subject with an e-mail name in the med.example.com domain
[a18]     can perform any action on any resource.
[a19]    </Description>
[a20]    <Target>
[a21]     <Subjects>
[a22]      <Subject>
[a23]       <SubjectMatch
[a24]       MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-
match">
[a25]        <AttributeValue
[a26]        DataType="http://www.w3.org/2001/XMLSchema#string">
[a27]         med.example.com
[a28]        </AttributeValue>
[a29]        <SubjectAttributeDesignator
[a30]        AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[a31]        DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
[a32]       </SubjectMatch>
[a33]      </Subject>
[a34]     </Subjects>
[a35]    </Target>
[a36]   </Rule>
[a37]  </Policy>
```

[a02] is a standard XML document tag indicating which version of XML is being used and what the character encoding is.

[a03] introduces the XACML Policy itself.

[a04] – [a05] are XML namespace declarations.

[a04] gives a URN for the XACML policies schema.

[a07] assigns a name to this policy instance. The name of a policy has to be unique for a given PDP so that there is no ambiguity if one policy is referenced from another policy. The `version` attribute is omitted, so it takes its default value of "1.0".

[a08] specifies the algorithm that will be used to resolve the results of the various rules that may be in the policy. The deny-overrides rule-combining algorithm specified here says that, if any rule evaluates to "Deny", then the policy must return "Deny". If all rules evaluate to "Permit", then the policy must return "Permit". The rule-combining algorithm, which is fully described in Annex C, also says what to do if an error were to occur when evaluating any rule, and what to do with rules that do not apply to a particular decision request.

[a09] – [a11] provide a text description of the policy. This description is optional.

[a12] describes the decision requests to which this policy applies. If the subject, resource, action and environment in a decision request do not match the values specified in the policy target, then the remainder of the policy does not need to be evaluated. This target section is useful for creating an index to a set of policies. In this simple example, the target section says the policy is applicable to any decision request.

[a13] introduces the one and only rule in this simple policy.

[a14] specifies the identifier for this rule. Just as for a policy, each rule must have a unique identifier (at least unique for any PDP that will be using the policy).

[a15] says what effect this rule has if the rule evaluates to "True". Rules can have an effect of either "Permit" or "Deny". In this case, if the rule is satisfied, it will evaluate to "Permit", meaning that, as far as this one rule is concerned, the requested access should be permitted. If a rule evaluates to "False", then it returns a result of "NotApplicable". If an error occurs when evaluating the rule, then the rule returns a result of "Indeterminate". As mentioned above, the rule-combining algorithm for the policy specifies how various rule values are combined into a single policy value.

[a16] – [a19] provide a text description of this rule. This description is optional.

[a20] introduces the target of the rule. As described above for the target of a policy, the target of a rule describes the decision requests to which this rule applies. If the subject, resource, action and environment in a decision request do not match the values specified in the rule target, then the remainder of the rule does not need to be evaluated, and a value of "NotApplicable" is returned to the rule evaluation.

The rule target is similar to the target of the policy itself, but with one important difference. [a23] – [a32] spells out a specific value that the subject in the decision request must match. The `<SubjectMatch>` element specifies a matching function in the `MatchId` attribute, a literal value of "med.example.com" and a pointer to a specific subject attribute in the request context by means of the `<SubjectAttributeDesignator>` element. The matching function will be used to compare the literal value with the value of the subject attribute . Only if the match returns "True" will this rule apply to a particular decision request. If the match returns "False", then this rule will return a value of "NotApplicable".

[a36] closes the rule. In this rule, all the work is done in the `<Target>` element. In more complex rules, the `<Target>` may have been followed by a `<Condition>` element (which could also be a set of conditions to be ANDed or ORed together).

[a37] closes the policy. As mentioned above, this policy has only one rule, but more complex policies may have any number of rules.

II.1.2 Example request context

Let us examine a hypothetical decision request that might be submitted to a PDP that executes the policy above. In English, the access request that generates the decision request may be stated as follows:

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

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

```
[a38]  <?xml version="1.0" encoding="UTF-8"?>
[a39]  <Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a40]  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
http://docs.oasis-
open.org/xacml/access_control-xacml-2.0-context-schema-os.xsd">
[a41]   <Subject>
[a42]    <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
[a43]     <AttributeValue>
[a44]      bs@simpsons.com
[a45]     </AttributeValue>
[a46]    </Attribute>
[a47]   </Subject>
[a48]   <Resource>
[a49]    <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#anyURI">
[a50]     <AttributeValue>
[a51]      file://example/med/record/patient/BartSimpson
[a52]     </AttributeValue>
[a53]    </Attribute>
[a54]   </Resource>
[a55]   <Action>
[a56]    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id"
DataType="http://www.w3.org/2001/XMLSchema#string">
[a57]     <AttributeValue>
[a58]      read
[a59]     </AttributeValue>
[a60]    </Attribute>
[a61]   </Action>
[a62]   <Environment/>
[a63]  </Request>
```

[a38] – [a40] contain the header information for the request context, and are used the same way as the header for the policy explained above.

The `<Subject>` element contains one or more attributes of the entity making the access request. There can be multiple subjects, and each subject can have multiple attributes. In this case, in [a41] – [a47], there is only one subject, and the subject has only one attribute: the subject's identity, expressed as an e-mail name, is "bs@simpsons.com". In this example, the `subject-category` attribute is omitted. Therefore, it adopts its default value of "access-subject".

The `<Resource>` element contains one or more attributes of the resource to which the subject (or subjects) has requested access. There can be only one `<Resource>` per decision request. Lines [a48] – [a54] contain the one attribute of the resource to which Bart Simpson has requested access: the resource identified by its file URI, which is "**file**://`medico/record/patient/BartSimpson`".

The `<Action>` element contains one or more attributes of the action that the subject (or subjects) wishes to take on the resource. There can be only one action per decision request. [a55] – [a61] describe the identity of the action Bart Simpson wishes to take, which is "read".

The `<Environment>` element, [a62], is empty.

[a63] closes the request context. A more complex request context may have contained some attributes not associated with the subject, the resource or the action. These would have been placed in an optional `<Environment>` element following the `<Action>` element.

The PDP processing this request context locates the policy in its policy repository. It compares the subject, resource, action and environment in the request context with the subjects, resources, actions and environments in the policy target. Since the policy target is empty, the policy matches this context.

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

II.1.3 Example response context

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

```
[a64]  <?xml version="1.0" encoding="UTF-8"?>
[a65]  <Response xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
http://docs.oasis-open.org/xacml/xacml-core-2.0-context-schema-os.xsd">
[a66]   <Result>
[a67]    <Decision>NotApplicable</Decision>
[a68]   </Result>
[a69]  </Response>
```

[a64] – [a65] contain the same sort of header information for the response as was described above for a policy.

The `<Result>` element in lines [a66] – [a68] contains the result of evaluating the decision request against the policy. In this case, the result is "NotApplicable". A policy can return "Permit", "Deny", "NotApplicable" or "Indeterminate". Therefore, the PEP is required to deny access.

[a69] closes the response context.

II.2 Example two

This clause contains an example XML document, an example request context and example XACML rules. The XML document is a medical record. Four separate rules are defined. These illustrate a rule-combining algorithm, conditions and obligations.

II.2.1 Example medical record instance

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

```
[a70]  <?xml version="1.0" encoding="UTF-8"?>
[a71]   <record xmlns="urn:example:med:schemas:record"
[a72]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
[a73]    <patient>
[a74]     <patientName>
[a75]      <first>Bartholomew</first>
[a76]      <last>Simpson</last>
[a77]     </patientName>
[a78]     <patientContact>
[a79]      <street>27 Shelbyville Road</street>
[a80]      <city>Springfield</city>
[a81]      <state>MA</state>
[a82]      <zip>12345</zip>
[a83]      <phone>555.123.4567</phone>
[a84]      <fax/>
[a85]      <email/>
[a86]     </patientContact>
[a87]     <patientDoB>1992-03-21</patientDoB>
[a88]     <patientGender>male</patientGender>
[a89]     <patient-number>555555</patient-number>
[a90]    </patient>
[a91]    <parentGuardian>
[a92]     <parentGuardianId>HS001</parentGuardianId>
[a93]     <parentGuardianName>
[a94]      <first>Homer</first>
[a95]      <last>Simpson</last>
[a96]     </parentGuardianName>
[a97]     <parentGuardianContact>
[a98]      <street>27 Shelbyville Road</street>
[a99]      <city>Springfield</city>
[a100]     <state>MA</state>
[a101]     <zip>12345</zip>
[a102]     <phone>555.123.4567</phone>
[a103]     <fax/>
[a104]     <email>homers@aol.com</email>
```

```
[a105]    </parentGuardianContact>
[a106]   </parentGuardian>
[a107]   <primaryCarePhysician>
[a108]    <physicianName>
[a109]     <first>Julius</first>
[a110]     <last>Hibbert</last>
[a111]    </physicianName>
[a112]    <physicianContact>
[a113]     <street>1 First St</street>
[a114]     <city>Springfield</city>
[a115]     <state>MA</state>
[a116]     <zip>12345</zip>
[a117]     <phone>555.123.9012</phone>
[a118]     <fax>555.123.9013</fax>
[a119]     <email/>
[a120]    </physicianContact>
[a121]     <registrationID>ABC123</registrationID>
[a122]    </primaryCarePhysician>
[a123]    <insurer>
[a124]     <name>Blue Cross</name>
[a125]     <street>1234 Main St</street>
[a126]     <city>Springfield</city>
[a127]     <state>MA</state>
[a128]     <zip>12345</zip>
[a129]     <phone>555.123.5678</phone>
[a130]     <fax>555.123.5679</fax>
[a131]     <email/>
[a132]    </insurer>
[a133]   <medical>
[a134]    <treatment>
[a135]     <drug>
[a136]      <name>methylphenidate hydrochloride</name>
[a137]      <dailyDosage>30mgs</dailyDosage>
[a138]      <startDate>1999-01-12</startDate>
[a139]     </drug>
[a140]     <comment>
[a141]      patient exhibits side-effects of skin coloration and carpal
degeneration
[a142]     </comment>
[a143]    </treatment>
[a144]    <result>
[a145]     <test>blood pressure</test>
[a146]     <value>120/80</value>
[a147]     <date>2001-06-09</date>
[a148]     <performedBy>Nurse Betty</performedBy>
[a149]    </result>
[a150]   </medical>
[a151]  </record>
```

II.2.2 Example request context

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

```
[a152] <?xml version="1.0" encoding="UTF-8"?>
[a153] <Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" urn:oasis:names:tc:xacml:2.0:context:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-
os.xsd">
[a154]  <Subject>
[a155]   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject-
category"
DataType="http://www.w3.org/2001/XMLSchema#anyURI">
[a156]    <AttributeValue>urn:oasis:names:tc:xacml:1.0:subject-
category:access-subject</AttributeValue>
[a157]   </Attribute>
[a158]   <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
 DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="med.example.com">
```

```
[a159]    <AttributeValue>CN=Julius Hibbert</AttributeValue>
[a160]   </Attribute>
[a161]   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:name-
format"
DataType="http://www.w3.org/2001/XMLSchema#anyURI"
Issuer="med.example.com">
[a162]    <AttributeValue>
[a163]     urn:oasis:names:tc:xacml:1.0:datatype:x500name
[a164]    </AttributeValue>
[a165]   </Attribute>
[a166]   <Attribute
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="med.example.com">
[a167]    <AttributeValue>physician</AttributeValue>
[a168]   </Attribute>
[a169]   <Attribute
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:physician-id"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="med.example.com">
[a170]    <AttributeValue>jh1234</AttributeValue>
[a171]   </Attribute>
[a172]  </Subject>
[a173]  <Resource>
[a174]   <ResourceContent>
[a175]    <md:record xmlns:md="urn:example:med:schemas:record"
xsi:schemaLocation="urn:example:med:schemas:record
http:www.med.example.com/schemas/record.xsd">
[a176]     <md:patient>
[a177]      <md:patientDoB>1992-03-21</md:patientDoB>
[a178]      <md:patient-number>555555</md:patient-number>
[a179]     </md:patient>
[a180]    </md:record>
[a181]   </ResourceContent>
[a182]   <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
[a183]    <AttributeValue>
[a184]     //med.example.com/records/bart-simpson.xml#
[a185] xmlns(md=:Resource/ResourceContent/xpointer
[a186] (/md:record/md:patient/md:patientDoB)
[a187]    </AttributeValue>
[a188]   </Attribute>
[a189] </Resource>
[a190]  <Action>
[a191]   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id"
DataType="http://www.w3.org/2001/XMLSchema#string">
[a192]    <AttributeValue>read</AttributeValue>
[a193]   </Attribute>
[a194]  </Action>
[a195]  <Environment/>
[a196] </Request>
```

[a152] – [a153] Standard namespace declarations.

[a154] – [a172] Subject attributes are placed in the `<Subject>` element of the `<Request>` element. Each attribute consists of the attribute meta-data and the attribute value. There is only one subject involved in this request.

[a155] – [a157] Each `<Subject>` element has a `SubjectCategory` attribute. The value of this attribute describes the role that the related subject plays in making the decision request. The value of "`access-subject`" denotes the identity for which the request was issued.

[a158] – [a160] Subject `subject-id` attribute.

[a161] – [a165] The format of the subject-id.

[a166] – [a168] Subject `role` attribute.

[a169] – [a171] Subject `physician-id` attribute.

[a173] – [a189] Resource attributes are placed in the <Resource> element of the <Request> element. Each attribute consists of attribute meta-data and an attribute value.

[a174] – [a181] Resource content. The XML resource instance, access to all or part of which may be requested, is placed here.

[a182] – [a188] The identifier of the Resource instance for which access is requested, which is an XPath expression into the <ResourceContent> element that selects the data to be accessed.

[a190] – [a194] Action attributes are placed in the <Action> element of the <Request> element.

[a192] Action identifier.

[a195] The empty <Environment> element.

II.2.3 Example plain-language rules

The following plain-language rules are to be enforced:

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

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

II.2.4 Example XACML rule instances

II.2.4.1 Rule 1

Rule 1 illustrates a simple rule with a single <Condition> element. It also illustrates the use of the <VariableDefinition> element to define a function that may be used throughout the policy.

The following XACML <Rule> instance expresses rule 1:

```
[a197] <?xml version="1.0" encoding="UTF-8"?>
[a198] <Policy
[a199] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
[a200] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-
os.xsd"
[a201] xmlns:md="http://www.med.example.com/schemas/record.xsd"
[a202] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:1"
[a203] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides">
[a204]  <PolicyDefaults>
[a205]         <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
19991116</XPathVersion>
[a206]  </PolicyDefaults>
[a207]  <Target/>
[a208]  <VariableDefinition VariableId="17590034">
[a209]   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
[a210]    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
one-and-only">
[a211]     <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:patient-number"
[a212]     DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a213]    </Apply>
[a214]    <Apply
[a215]    FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-
only">
[a216]     <AttributeSelector
[a217]     RequestContextPath="//xacml-context:Resource/xacml-
context:ResourceContent/md:record/md:patient/md:patient-number/text()"
```

```
[a218]     DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a219]    </Apply>
[a220]   </Apply>
[a221]  </VariableDefinition>
[a222]  <Rule
[a223]  RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1"
[a224]  Effect="Permit">
[a225]   <Description>
[a226]     A person may read any medical record in the
[a227]     http://www.med.example.com/schemas/record.xsd namespace
[a228]     for which he or she is the designated patient
[a229]   </Description>
[a230]   <Target>
[a231]    <Resources>
[a232]     <Resource>
[a233]      <ResourceMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a234]       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a235]         urn:example:med:schemas:record
[a236]       </AttributeValue>
[a237]       <ResourceAttributeDesignator AttributeId=
[a238]       "urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
[a239]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a240]      </ResourceMatch>
[a241]      <ResourceMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
[a242]       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a243]         /md:record
[a244]       </AttributeValue>
[a245]       <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
[a246]        DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a247]      </ResourceMatch>
[a248]     </Resource>
[a249]    </Resources>
[a250]    <Actions>
[a251]     <Action>
[a252]      <ActionMatch
[a253]       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a254]       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a255]          read
[a256]       </AttributeValue>
[a257]       <ActionAttributeDesignator
[a258]        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[a259]        DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a260]      </ActionMatch>
[a261]     </Action>
[a262]    </Actions>
[a263]   </Target>
[a264]   <Condition>
[a265]    <VariableReference VariableId="17590034"/>
[a266]   </Condition>
[a267]  </Rule>
[a268] </Policy>
```

[a199] – [a201] XML namespace declarations.

[a205] XPath expressions in the policy are to be interpreted according to W3C XPath:1999.

[a208] – [a221] A `<VariableDefinition>` element. It defines a function that evaluates the truth of the statement: the `patient-number` subject attribute is equal to the `patient-number` in the resource.

[a209] The `FunctionId` attribute names the function to be used for comparison. In this case, comparison is done with the "`urn:oasis:names:tc:xacml:1.0:function:string-equal`" function; this function takes two arguments of type "http://www.w3.org/2001/XMLSchema#string".

[a210] The first argument of the variable definition is a function specified by the `FunctionId` attribute. Since `urn:oasis:names:tc:xacml:1.0:function:string-equal` takes arguments of type

"http://www.w3.org/2001/XMLSchema#string" and `SubjectAttributeDesignator` selects a bag of type "http://www.w3.org/2001/XMLSchema#string", "`urn:oasis:names:tc:xacml:1.0:function:string-one-and-only`" is used. This function guarantees that its argument evaluates to a bag containing exactly one value.

[a211] The `SubjectAttributeDesignator` selects a bag of values for the `patient-number` subject attribute in the request context.

[a215] The second argument of the variable definition is a function specified by the `FunctionId` attribute. Since "`urn:oasis:names:tc:xacml:1.0:function:string-equal`" takes arguments of type "http://www.w3.org/2001/XMLSchema#string" and the `AttributeSelector` selects a bag of type "http://www.w3.org/2001/XMLSchema#string", "`urn:oasis:names:tc:xacml:1.0:function:string-one-and-only`" is used. This function guarantees that its argument evaluates to a bag containing exactly one value.

[a216] The `<AttributeSelector>` element selects a bag of values from the request context using a free-form XPath expression. In this case, it selects the value of the `patient-number` in the resource. Note that the namespace prefixes in the XPath expression are resolved with the standard XML namespace declarations.

[a223] Rule identifier.

[a224] Rule effect declaration. When a rule evaluates to 'True', it emits the value of the `Effect` attribute. This value is then combined with the `Effect` values of other rules according to the rule-combining algorithm.

[a225] – [a229] Free form description of the rule.

[a230] – [a263] A rule target defines a set of decision requests that the rule is intended to evaluate. In this example, the `<Subjects>` and `<Environments>` elements are omitted.

[a231] – [a249] The `<Resources>` element contains a disjunctive sequence of `<Resource>` elements. In this example, there is just one.

[a232] – [a248] The `<Resource>` element encloses the conjunctive sequence of `ResourceMatch` elements. In this example, there are two.

[a233] – [a240] The first `<ResourceMatch>` element compares its first and second child elements according to the matching function. A match is positive if the value of the first argument matches any of the values selected by the second argument. This match compares the target namespace of the requested document with the value of "`urn:example:med:schemas:record`".

[a233] The `MatchId` attribute names the matching function.

[a235] Literal attribute value to match.

[a237] – [a239] The `<ResourceAttributeDesignator>` element selects the target namespace from the resource contained in the request context. The attribute name is specified by the `AttributeId`.

[a241] – [a247] The second `<ResourceMatch>` element. This match compares the results of two XPath expressions. The second XPath expression is the location path to the requested XML element and the first XPath expression is the literal value "`/md:record`". The "xpath-node-match" function evaluates to "True" if the requested XML element is below the "`/md:record`" element.

[a250] – [a262] The `<Actions>` element contains a disjunctive sequence of `<Action>` elements. In this case, there is just one `<Action>` element.

[a251] – [a261] The `<Action>` element contains a conjunctive sequence of `<ActionMatch>` elements. In this case, there is just one `<ActionMatch>` element.

[a252] – [a260] The `<ActionMatch>` element compares its first and second child elements according to the matching function. The match is positive if the value of the first argument matches any of the values selected by the second argument. In this case, the value of the `action-id` action attribute in the request context is compared with the literal value "`read`".

[a264] – [a266] The `<Condition>` element. A condition must evaluate to "True" for the rule to be applicable. This condition contains a reference to a variable definition defined elsewhere in the policy.

II.2.4.2 Rule 2

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

```
[a269] <?xml version="1.0" encoding="UTF-8"?>
[a270] <Policy
[a271] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
[a272] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
[a273] xmlns:xf="urn:oasis:names:tc:xacml:2.0:data-types"
[a274] xmlns:md="http:www.med.example.com/schemas/record.xsd"
[a275] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
overrides">
[a276]  <PolicyDefaults>
[a277]   <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116</XPathVersion>
[a278]  </PolicyDefaults>
[a279]  <Target/>
[a280]  <VariableDefinition VariableId="17590035">
[a281]   <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:date-less-or-
equal">
[a282]    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-
only">
[a283]     <EnvironmentAttributeDesignator
[a284]     AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-date"
[a285]     DataType="http://www.w3.org/2001/XMLSchema#date"/>
[a286]    </Apply>
[a287]    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-
yearMonthDuration">
[a288]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-
only">
[a289]      <AttributeSelector RequestContextPath=
[a290]      "//md:record/md:patient/md:patientDoB/text()"
[a291]      DataType="http://www.w3.org/2001/XMLSchema#date"/>
[a292]     </Apply>
[a293]     <AttributeValue
[a294]     DataType="urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration">
[a295]      <xf:dt-yearMonthDuration>
[a296]       P16Y
[a297]      </xf:dt-yearMonthDuration>
[a298]     </AttributeValue>
[a299]    </Apply>
[a300]   </Apply>
[a301]  </VariableDefinition>
[a302]  <Rule
[a303]  RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:2"
[a304]  Effect="Permit">
[a305]   <Description>
[a306]    A person may read any medical record in the
[a307]    http://www.med.example.com/records.xsd namespace
[a308]    for which he or she is the designated parent or guardian,
[a309]    and for which the patient is under 16 years of age
[a310]   </Description>
[a311]   <Target>
[a312]    <Resources>
[a313]     <Resource>
[a314]      <ResourceMatch
[a315]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a316]       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
[a317]          http://www.med.example.com/schemas/record.xsd
[a318]        </AttributeValue>
[a319]       <ResourceAttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
[a320]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
```

```
[a321]       </ResourceMatch>
[a322]       <ResourceMatch
[a323]       MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
[a324]        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
[a325]          /md:record
[a326]        </AttributeValue>
[a327]        <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
[a328]        DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a329]       </ResourceMatch>
[a330]      </Resource>
[a331]     </Resources>
[a332]     <Actions>
[a333]      <Action>
[a334]       <ActionMatch
[a335]       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a336]        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
[a337]          read
[a338]        </AttributeValue>
[a339]        <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[a340]        DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a341]      </ActionMatch>
[a342]      </Action>
[a343]     </Actions>
[a344]    </Target>
[a345]    <Condition>
[a346]    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
[a347]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a348]      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-
and-only">
[a349]       <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:
[a350] parent-guardian-id"
[a351]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a352]      </Apply>
[a353]      <Apply
[a354]      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[a355]       <AttributeSelector
[a356]       RequestContextPath="//xacml-context:Resource/xacml-
context:ResourceContent/md:record/md:parentGuardian/md:parentGuardianId/text()"
[a357]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a358]       </Apply>
[a359]      </Apply>
[a360]      <VariableReference VariableId="17590035"/>
[a361]     </Apply>
[a362]    </Condition>
[a363]   </Rule>
[a364] </Policy>
```

[a280] – [a301] The `<VariableDefinition>` element contains part of the condition (i.e., is the patient under 16 years of age?). The patient is under 16 years of age if the current date is less than the date computed by adding 16 to the patient's date of birth.

[a281] – [a300] "`urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal`" is used to compute the difference of two date arguments.

[a282] – [a286] The first date argument uses "`urn:oasis:names:tc:xacml:1.0:function:date-one-and-only`" to ensure that the bag of values selected by its argument contains exactly one value of type "http://www.w3.org/2001/XMLSchema#date".

[a284] The current date is evaluated by selecting the "`urn:oasis:names:tc:xacml:1.0:environment:current-date`" environment attribute.

[a287] – [a299] The second date argument uses "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" to compute the date of the patient's sixteenth birthday by adding 16 years to the patient's date of birth. The first of its arguments is of type "http://www.w3.org/2001/XMLSchema#date" and the second is of type "urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration".

[a289] The <AttributeSelector> element selects the patient's date of birth by taking the XPath expression over the resource content.

[a293] – [a298] Year Month Duration of 16 years.

[a311] – [a344] Rule declaration and rule target. See Rule 1 in II.4.2.4.1 for the detailed explanation of these elements.

[a345] – [a362] The <Condition> element. The condition must evaluate to "True" for the rule to be applicable. This condition evaluates the truth of the statement: the requestor is the designated parent or guardian and the patient is under 16 years of age. It contains one embedded <Apply> element and one referenced <VariableDefinition> element.

[a346] The condition uses the "urn:oasis:names:tc:xacml:1.0:function:and" function. This is a boolean function that takes one or more boolean arguments (2 in this case) and performs the logical "AND" operation to compute the truth value of the expression.

[a347] – [a359] The first part of the condition is evaluated (i.e., is the requestor the designated parent or guardian?). The function is "urn:oasis:names:tc:xacml:1.0:function:string-equal" and it takes two arguments of type "http://www.w3.org/2001/XMLSchema#string".

[a348] designates the first argument. Since "urn:oasis:names:tc:xacml:1.0:function:string-equal" takes arguments of type "http://www.w3.org/2001/XMLSchema#string", "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" is used to ensure that the subject attribute "urn:oasis:names:tc:xacml:2.0:example:attribute:parent-guardian-id" in the request context contains exactly one value.

[a353] designates the second argument. The value of the subject attribute "urn:oasis:names:tc:xacml:2.0:example:attribute:parent-guardian-id" is selected from the request context using the <SubjectAttributeDesignator> element.

[a354] As above, the "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" is used to ensure that the bag of values selected by its argument contains exactly one value of type "http://www.w3.org/2001/XMLSchema#string".

[a355] The second argument selects the value of the <md:parentGuardianId> element from the resource content using the <AttributeSelector> element. This element contains a free-form XPath expression, pointing into the request context. Note that all namespace prefixes in the XPath expression are resolved with standard namespace declarations. The AttributeSelector evaluates to the bag of values of type "http://www.w3.org/2001/XMLSchema#string".

[a360] references the <VariableDefinition> element, where the second part of the condition is defined.

II.2.4.3 Rule 3

Rule 3 illustrates the use of an obligation. The XACML <Rule> element syntax does not include an element suitable for carrying an obligation, therefore Rule 3 has to be formatted as a <Policy> element.

```
[a365] <?xml version="1.0" encoding="UTF-8"?>
[a366] <Policy
[a367]  xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
[a368]  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a369]  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-
os.xsd"
[a370] xmlns:md="http:www.med.example.com/schemas/record.xsd"
[a371]  PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:3"
[a372]  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides">
[a373]  <Description>
[a374]   Policy for any medical record in the
[a375]   http://www.med.example.com/schemas/record.xsd namespace
[a376]  </Description>
[a377]  <PolicyDefaults>
```

```
[a378]   <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
19991116</XPathVersion>
[a379]  </PolicyDefaults>
[a380]  <Target>
[a381]   <Resources>
[a382]    <Resource>
[a383]     <ResourceMatch
[a384]     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a385]      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a386]       urn:example:med:schemas:record
[a387]      </AttributeValue>
[a388]      <ResourceAttributeDesignator AttributeId=
[a389]      "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
[a390]      DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a391]     </ResourceMatch>
[a392]    </Resource>
[a393]   </Resources>
[a394]  </Target>
[a395]  <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:3"
[a396]  Effect="Permit">
[a397]   <Description>
[a398]    A physician may write any medical element in a record
[a399]    for which he or she is the designated primary care
[a400]    physician, provided an email is sent to the patient
[a401]   </Description>
[a402]   <Target>
[a403]    <Subjects>
[a404]     <Subject>
[a405]      <SubjectMatch
[a406]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a407]       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a408]         physician
[a409]        </AttributeValue>
[a410]        <SubjectAttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:2.0:example:attribute:role"
[a411]        DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a412]      </SubjectMatch>
[a413]     </Subject>
[a414]    </Subjects>
[a415]    <Resources>
[a416]     <Resource>
[a417]      <ResourceMatch
[a418]      MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-
match">
[a419]       <AttributeValue
[a420]        DataType="http://www.w3.org/2001/XMLSchema#string">
[a421]          /md:record/md:medical
[a422]        </AttributeValue>
[a423]       <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
[a424]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a425]      </ResourceMatch>
[a426]     </Resource>
[a427]    </Resources>
[a428]    <Actions>
[a429]     <Action>
[a430]      <ActionMatch
[a431]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a432]       <AttributeValue
[a433]       DataType="http://www.w3.org/2001/XMLSchema#string">
[a434]         write
[a435]       </AttributeValue>
[a436]       <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[a437]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a438]      </ActionMatch>
[a439]     </Action>
[a440]    </Actions>
[a441]   </Target>
```

```
[a442]   <Condition>
[a443]   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
[a444]    <Apply
[a445]    FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-
only">
[a446]     <SubjectAttributeDesignator
[a447]     AttributeId="urn:oasis:names:tc:xacml:2.0:example:
attribute:physician-id"
[a448]     DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a449]    </Apply>
[a450]    <Apply
[a451]    FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-
only">
[a452]     <AttributeSelector RequestContextPath=
[a453]     "//xacml-context:Resource/xacml-
context:ResourceContent/md:record/md:primaryCarePhysician/md:registrationID
/text()"
[a454]     DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a455]    </Apply>
[a456]     </Apply>
[a457]   </Condition>
[a458]  </Rule>
[a459]  <Obligations>
[a460]   <Obligation
ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
[a461]   FulfillOn="Permit">
[a462]    <AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:mailto"
[a463]    DataType="http://www.w3.org/2001/XMLSchema#string">
[a464]     &lt;AttributeSelector RequestContextPath=
[a465]     "//md:/record/md:patient/md:patientContact/md:email"
[a466]     DataType="http://www.w3.org/2001/XMLSchema#string"/&gt ;
[a467]    </AttributeAssignment>
[a468]    <AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:text"
[a469]    DataType="http://www.w3.org/2001/XMLSchema#string">
[a470]      Your medical record has been accessed by:
[a471]    </AttributeAssignment>
[a472]    <AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:text"
[a473]    DataType="http://www.w3.org/2001/XMLSchema#string">
[a474]     &lt;SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[a475]     DataType="http://www.w3.org/2001/XMLSchema#string"/&gt;
[a476]    </AttributeAssignment>
[a477]   </Obligation>
[a478]  </Obligations>
[a479] </Policy>
```

[a366] – [a372] The `<Policy>` element includes standard namespace declarations as well as policy specific parameters, such as `PolicyId` and `RuleCombiningAlgId`.

[a371] Policy identifier. This parameter allows the policy to be referenced by a policy set.

[a372] The Rule combining algorithm identifies the algorithm for combining the outcomes of rule evaluation.

[a373] – [a376] Free-form description of the policy.

[a379] – [a394] Policy target. The policy target defines a set of applicable decision requests. The structure of the `<Target>` element in the `<Policy>` is identical to the structure of the `<Target>` element in the `<Rule>`. In this case, the policy target is the set of all XML resources that conform to the namespace "`urn:example:med:schemas:record`".

[a395] The only `<Rule>` element included in this `<Policy>`. Two parameters are specified in the rule header: `RuleId` and `Effect`.

[a402] – [a441] The rule target further constrains the policy target.

[a405] – [a412] The `<SubjectMatch>` element targets the rule at subjects whose "`urn:oasis:names:tc:xacml:2.0:example:attribute:role`" subject attribute is equal to "`physician`".

[a417] – [a425] The <ResourceMatch> element targets the rule at resources that match the XPath expression "/md:record/md:medical".

[a430] – [a438] The <ActionMatch> element targets the rule at actions whose "urn:oasis:names:tc:xacml:1.0:action:action-id" action attribute is equal to "write".

[a442] – [a457] The <Condition> element. For the rule to be applicable to the decision request, the condition must evaluate to "True". This condition compares the value of the "urn:oasis:names:tc:xacml:2.0:example:attribute:physician-id" subject attribute with the value of the <registrationId> element in the medical record that is being accessed.

[a459] – [a478] The <Obligations> element. Obligations are a set of operations that must be performed by the PEP in conjunction with an authorization decision. An obligation may be associated with a "Permit" or "Deny" authorization decision. The element contains a single obligation.

[a460] – [a477] The <Obligation> element consists of the ObligationId attribute, the authorization decision value for which it must be fulfilled, and a set of attribute assignments. The PDP does not resolve the attribute assignments. This is the job of the PEP.

[a460] The ObligationId attribute identifies the obligation. In this case, the PEP is required to send email.

[a461] The FulfillOn attribute defines the authorization decision value for which this obligation must be fulfilled. In this case, when access is permitted.

[a462] – [a467] The first parameter indicates where the PEP will find the email address in the resource.

[a468] – [a471] The second parameter contains literal text for the email body.

[a472] – [a476] The third parameter indicates where the PEP will find further text for the email body in the resource.

II.2.4.4 Rule 4

Rule 4 illustrates the use of the "Deny" Effect value, and a <Rule> with no <Condition> element.

```
[a480] <?xml version="1.0" encoding="UTF-8"?>
[a481] <Policy
[a482] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
[a483] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-
os.xsd"
[a484] xmlns:md="http:www.med.example.com/schemas/record.xsd"
[a485] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:4"
[a486] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides">
[a487]  <PolicyDefaults>
[a488]   <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
19991116</XPathVersion>
[a489]  </PolicyDefaults>
[a490]  <Target/>
[a491]  <Rule
[a492]  RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:4"
[a493]  Effect="Deny">
[a494]   <Description>
[a495]    An Administrator shall not be permitted to read or write
[a496]    medical elements of a patient record in the
[a497]    http://www.med.example.com/records.xsd namespace.
[a498]   </Description>
[a499]   <Target>
[a500]    <Subjects>
[a501]     <Subject>
[a502]      <SubjectMatch
[a503]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a504]       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a505]        administrator
[a506]       </AttributeValue>
[a507]       <SubjectAttributeDesignator AttributeId=
[a508]       "urn:oasis:names:tc:xacml:2.0:example:attribute:role"
[a509]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a510]      </SubjectMatch>
```

```
[a511]     </Subject>
[a512]    </Subjects>
[a513]    <Resources>
[a514]     <Resource>
[a515]      <ResourceMatch
[a516]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a517]       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a518]        urn:example:med:schemas:record
[a519]       </AttributeValue>
[a520]       <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
[a521]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a522]      </ResourceMatch>
[a523]      <ResourceMatch
[a524]      MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-
match">
[a525]       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a526]        /md:record/md:medical
[a527]       </AttributeValue>
[a528]       <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
[a529]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a530]      </ResourceMatch>
[a531]     </Resource>
[a532]    </Resources>
[a533]    <Actions>
[a534]     <Action>
[a535]      <ActionMatch
[a536]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a537]       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a538]        read
[a539]       </AttributeValue>
[a540]       <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[a541]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a542]      </ActionMatch>
[a543]     </Action>
[a544]     <Action>
[a545]      <ActionMatch
[a546]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a547]       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a548]        write
[a549]       </AttributeValue>
[a550]       <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[a551]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a552]      </ActionMatch>
[a553]     </Action>
[a554]    </Actions>
[a555]   </Target>
[a556]  </Rule>
[a557] </Policy>
```

[a492] – [a493] The `<Rule>` element declaration.

[a493] Rule `Effect`. Every rule that evaluates to "True" emits the rule effect as its value. This rule `Effect` is "Deny" meaning that according to this rule, access must be denied when it evaluates to "True".

[a494] – [a498] Free form description of the rule.

[a499] – [a555] Rule target. The Rule target defines the set of decision requests that are applicable to the rule.

[a502] – [a510] The `<SubjectMatch>` element targets the rule at subjects whose "`urn:oasis:names:tc:xacml:2.0:example:attribute:role`" subject attribute is equal to "administrator".

[a513] – [a532] The <Resources> element contains one <Resource> element, which (in turn) contains two <ResourceMatch> elements. The target matches if the resource identified by the request context matches both resource match criteria.

[a515] – [a522] The first <ResourceMatch> element targets the rule at resources whose "urn:oasis:names:tc:xacml:2.0:resource:target-namespace" resource attribute is equal to "urn:example:med:schemas:record".

[a523] – [a530] The second <ResourceMatch> element targets the rule at XML elements that match the XPath expression "/md:record/md:medical".

[a533] – [a554] The <Actions> element contains two <Action> elements, each of which contains one <ActionMatch> element. The target matches if the action identified in the request context matches either of the action match criteria.

[a535] – [a552] The <ActionMatch> elements target the rule at actions whose "urn:oasis:names:tc:xacml:1.0:action:action-id" action attribute is equal to "read" or "write".

This rule does not have a <Condition> element.

II.2.4.5 Example PolicySet

This clause uses the examples of the previous clauses to illustrate the process of combining policies. The policy governing read access to medical elements of a record is formed from each of the four rules described in II.4.2.3. In plain language, the combined rule is:

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

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

```
[a558] <?xml version="1.0" encoding="UTF-8"?>
[a559]  <PolicySet
[a560]  xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
[a561]  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-
os.xsd"
[a562]  PolicySetId=
[a563]  "urn:oasis:names:tc:xacml:2.0:example:policysetid:1"
[a564]  PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:
[a565]  policy-combining-algorithm:deny-overrides">
[a566]   <Description>
[a567]    Example policy set.
[a568]   </Description>
[a569]   <Target>
[a570]    <Resources>
[a571]     <Resource>
[a572]      <ResourceMatch
[a573]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a574]       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
[a575]        urn:example:med:schema:records
[a576]       </AttributeValue>
[a577]       <ResourceAttributeDesignator AttributeId=
[a578]       "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
[a579]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a580]      </ResourceMatch>
[a581]     </Resource>
[a582]    </Resources>
[a583]   </Target>
[a584]  <PolicyIdReference>
[a585]   urn:oasis:names:tc:xacml:2.0:example:policyid:3
[a586]  </PolicyIdReference>
[a587]  <Policy
[a588]  PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
```

```
[a589]  RuleCombiningAlgId=
[a590] "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
overrides">
[a591]   <Target/>
[a592]   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1"
[a593]   Effect="Permit">
[a594]   </Rule>
[a595]   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:2"
[a596]   Effect="Permit">
[a597]   </Rule>
[a598]   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:4"
[a599]   Effect="Deny">
[a600]   </Rule>
[a601]  </Policy>
[a602] </PolicySet>
```

[a559] – [a565] The `<PolicySet>` element declaration. Standard XML namespace declarations are included.

[a562] The `PolicySetId` attribute is used for identifying this policy set for possible inclusion in another policy set.

[a564] The policy combining algorithm identifier. Policies and policy sets in this policy set are combined according to the specified policy combining algorithm when the authorization decision is computed.

[a566] – [a568] Free form description of the policy set.

[a569] – [a583] The policy set `<Target>` element defines the set of decision requests that are applicable to this `<PolicySet>` element.

[a584] `PolicyIdReference` includes a policy by id.

[a588] `Policy 2` is explicitly included in this policy set. The rules in `Policy 2` are omitted for clarity.

Appendix III

Example description of higher order bag functions

III.1 Example of higher-order bag functions

This appendix describes functions in XACML that perform operations on bags such that functions may be applied to the bags in general.

For example purposes, a general-purpose functional language called Haskell (see [Haskell]) is used to specify the semantics of these functions. Although the English description is adequate, a proper specification of the semantics is helpful.

For a quick summary, in the following Haskell notation a function definition takes the form of clauses that are applied to patterns of structures, namely lists. The symbol "[]" denotes the empty list, whereas the expression "(x:xs)" matches against an argument of a non-empty list of which "x" represents the first element of the list, and "xs" is the rest of the list, which may be an empty list. We use the Haskell notion of a list, which is an ordered collection of elements, to model the XACML bags of values.

A simple Haskell definition of a familiar function "urn:oasis:names:tc:xacml:1.0:function:and" that takes a list of values of type boolean is defined as follows:

```
and:: [Bool]      -> Bool
and []            = True
and (x:xs)        = x && (and xs)
```

The first definition line denoted by a "::" formally describes the data-type of the function, which takes a list of booleans, denoted by "[Bool]", and returns a boolean, denoted by "Bool". The second definition line is a clause that states that the function "and" applied to the empty list is "True". The third definition line is a clause that states that for a non-empty list, such that the first element is "x", which is a value of data-type Bool, the function "and" applied to x shall be combined with, using the logical conjunction function, which is denoted by the infix symbol "&&", the result of recursively applying the function "and" to the rest of the list. Of course, an application of the "and" function is "True" if, and only if, the list to which it is applied is empty or every element of the list is "True". For example, the evaluation of the following Haskell expressions,

(and []), (and [True]), (and [True,True]), (and [True,True,False])

evaluate to "True", "True", "True", and "False", respectively.

1) `urn:oasis:names:tc:xacml:1.0:function:any-of`

In Haskell, the semantics of this operation is as follows:

```
any_of :: ( a -> b -> Bool )        -> a -> [b] -> Bool
any_of  f  a  []                            = False
any_of  f  a  (x:xs)                        = (f  a  x) || (any_of  f  a  xs)
```

In the above notation, "f" is the function to be applied, "a" is the primitive value, and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression shall return "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
      <Function
            FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-   equal"/>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            Paul
      </AttributeValue>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                  John
            </AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                  Paul
            </AttributeValue>
            <AttributeValue
                  DataType="http://www.w3.org/2001/XMLSchema#string">George
            </AttributeValue>
            <AttributeValue
                  DataType="http://www.w3.org/2001/XMLSchema#string">
                  Ringo
            </AttributeValue>
      </Apply>
</Apply>
```

This expression is "True" because the first argument is equal to at least one of the elements of the bag, according to the function.

2) `urn:oasis:names:tc:xacml:1.0:function:all-of`

In Haskell, the semantics of this operation is as follows:

```
all_of :: ( a -> b -> Bool ) -> a -> [b] -> Bool
all_of  f  a  []                            = True
all_of  f  a  (x:xs)                        = (f  a  x) && (all_of  f  a  xs)
```

In the above notation, "f" is the function to be applied, "a" is the primitive value, and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression shall evaluate to "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of">
      <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
      greater"/>
      <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
            <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#integer">9</AttributeValue>
            <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
            <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
            <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
      </Apply>
</Apply>
```

This expression is "True" because the first argument (10) is greater than all of the elements of the bag (9,3,4 and 2).

3) `urn:oasis:names:tc:xacml:1.0:function:any-of-any`

In Haskell, taking advantage of the "any_of" function defined above, the semantics of the "any_of_any" function is as follows:

```
any_of_any :: ( a -> b -> Bool )    -> [a]-> [b] -> Bool
any_of_any  f  []                   ys              = False
any_of_any  f  (x:xs)               ys              = (any_of  f  x  ys) || (any_of_any  f  xs  ys)
```

In the above notation, "f" is the function to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression shall evaluate to "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
       <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal"/>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Mary</AttributeValue>
       </Apply>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
       </Apply>
</Apply>
```

This expression is "True" because at least one of the elements of the first bag, namely "Ringo", is equal to at least one of the elements of the second bag.

4) `urn:oasis:names:tc:xacml:1.0:function:all-of-any`

In Haskell, taking advantage of the "any_of" function defined in Haskell above, the semantics of the "all_of_any" function is as follows:

```
all_of_any :: ( a -> b -> Bool )        -> [a]-> [b] -> Bool
all_of_any   f  []          ys                  = True
all_of_any   f  (x:xs)      ys                  = (any_of  f  x  ys) && (all_of_any f  xs  ys)
```

In the above notation, "f" is the function to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression shall evaluate to "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-any">
       <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
greater"/>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">20</AttributeValue>
       </Apply>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
```

```
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">19</AttributeValue>
      </Apply>
</Apply>
```

This expression is "True" because each of the elements of the first bag is greater than at least one of the elements of the second bag.

5) `urn:oasis:names:tc:xacml:1.0:function:any-of-all`

In Haskell, taking advantage of the "all_of" function defined in Haskell above, the semantics of the "any_of_all" function is as follows:

```
any_of_all :: ( a -> b -> Bool )            -> [a]-> [b] -> Bool
any_of_all  f  []          ys                    = False
any_of_all  f  (x:xs)      ys                    = (all_of  f  x  ys) || ( any_of_all  f  xs  ys)
```

In the above notation, "f" is the function name to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression shall evaluate to "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-all">
      <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
greater"/>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
      </Apply>
</Apply>
```

This expression is "True" because, for all of the values in the second bag, there is a value in the first bag that is greater.

6) `urn:oasis:names:tc:xacml:1.0:function:all-of-all`

In Haskell, taking advantage of the "all_of" function defined in Haskell above, the semantics of the "all_of_all" function is as follows:

```
all_of_all :: ( a -> b -> Bool )      -> [a] -> [b] -> Bool
all_of_all   f  []          ys              = True
all_of_all   f  (x:xs)      ys              = (all_of  f  x  ys) && (all_of_all  f  xs  ys)
```

In the above notation, "f" is the function to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression shall evaluate to "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-all">
      <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
greater"/>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">6</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
      </Apply>
```

```
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
      </Apply>
</Apply>
```

This expression is "True" because all elements of the first bag, "5" and "6", are each greater than all of the integer values "1", "2", "3", "4" of the second bag.

7) `urn:oasis:names:tc:xacml:1.0:function:map`

In Haskell, this function is defined as follows:

map:: (a -> b) -> [a] -> [b]

map f [] = []

map f (x:xs) = (f x) : (map f xs)

In the above notation, "f" is the function to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression,

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:map">
      <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
normalize-to-lower-case">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Hello</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">World!</AttributeValue>
      </Apply>
</Apply>
```

evaluates to a bag containing "hello" and "world!".

BIBLIOGRAPHY

[Haskell] THOMPSON (S.): Haskell: The Craft of Functional Programming (2nd Edition), *Addison Wesley*, ISBN 0-201-34275-8, 1996.

[IEEE 754] IEEE 754-1985, *Binary Floating-Point Arithmetic*, ISBN 1-5593-7653-8, IEEE Product No. SH10116-TBR.

[RBAC] ANSI INCITS 359-2004, *Information technology – Role Based Access Control*, http://csrc.nist.gov/rbac/.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	**Data networks, open system communications and security**
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2007