eXtensible Access Control Markup Language (XACML) Version 2.0

Committee draft 01, 16 Sep 2004

Document identifier: access_control-xacml-2.0-core-spec-cd-01

Location: http://docs.oasis-open.org/xacml/access_control-xacml-2.0-core-spec-cd-01.pdf

Editors:

Simon Godik, GlueCode Software
Tim Moses, Entrust

Abstract:

This specification defines version 2.0 of the extensible access-control markup language.

Status:

This version of the specification is an approved Committee Draft within the OASIS Access Control TC.

Access Control TC members should send comments on this specification to the xacml@lists.oasis-open.org list. Others may use the following link and complete the comment form: http://oasis-open.org/committees/comments/form.php?wg_abbrev=xacml.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Access Control TC web page (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml).

For any errata page for this specification, please refer to the Access Control TC web page (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml).

Table of contents

1. Introduction (non-normative)

1.1. Glossary

1.1.1 Preferred terms

Access - Performing an ***action***

Access control - Controlling ***access*** in accordance with a ***policy***

Action - An operation on a ***resource***

Applicable policy - The set of ***policies*** and ***policy sets*** that governs ***access*** for a specific ***decision request***

Attribute - Characteristic of a ***subject***, ***resource, action*** or ***environment*** that may be referenced in a ***predicate*** or ***target*** (see also – ***named attribute***)

Authorization decision - The result of evaluating ***applicable policy,*** returned by the ***PDP*** to the ***PEP.*** A function that evaluates to “Permit”, “Deny”, “Indeterminate” or “NotApplicable", and (optionally) a set of ***obligations***

Bag – An unordered collection of values, in which there may be duplicate values

Condition - An expression of ***predicates.*** A function that evaluates to "True", "False" or “Indeterminate”

Conjunctive sequence - a sequence of ***predicates*** combined using the logical ‘AND’ operation

Context - The canonical representation of a ***decision request*** and an ***authorization decision***

Context handler - The system entity that converts ***decision requests*** in the native request format to the XACML canonical form and converts ***authorization decisions*** in the XACML canonical form to the native response format

Decision – The result of evaluating a ***rule, policy*** or ***policy set***

Decision request - The request by a ***PEP*** to a ***PDP*** to render an ***authorization decision***

Disjunctive sequence - a sequence of ***predicates*** combined using the logical ‘OR’ operation

Effect - The intended consequence of a satisfied ***rule*** (either "Permit" or "Deny")

Environment - The set of ***attributes*** that are relevant to an ***authorization decision*** and are independent of a particular ***subject, resource*** or ***action***

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

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***

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

Policy administration point (PAP) - The system entity that creates a ***policy*** or ***policy set***

Policy-combining algorithm - The procedure for combining the ***decision*** and ***obligations*** from multiple ***policies***

Policy decision point (PDP) - The system entity that evaluates ***applicable policy*** and renders an ***authorization decision***. This term is defined in a joint effort by the IETF Policy Framework Working Group and the Distributed Management Task Force (DMTF)/Common Information Model (CIM) in [RFC3198]. This term corresponds to "Access Decision Function" (ADF) in [ISO10181-3].

Policy enforcement point (PEP) - The system entity that performs ***access control***, by making ***decision requests*** and enforcing ***authorization decisions***. This term is defined in a joint effort by the IETF Policy Framework Working Group and the Distributed Management Task Force (DMTF)/Common Information Model (CIM) in [RFC3198]. This term corresponds to "Access Enforcement Function" (AEF) in [ISO10181-3].

Policy information point (PIP) - The system entity that acts as a source of ***attribute*** values

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***

Predicate - A statement about ***attributes*** whose truth can be evaluated

Resource - Data, service or system component

Rule - A ***target***, an ***effect*** and a ***condition.*** A component of a ***policy***

Rule-combining algorithm - The procedure for combining ***decisions*** from multiple ***rules***

Subject - An actor whose ***attributes*** may be referenced by a ***predicate***

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

Type Unification - The method by which two type expressions are "unified". The type expressions are matched along their structure. Where a type variable appears in one expression it is then "unified" to represent the corresponding structure element of the other expression, be it another variable or subexpression. All variable assignments must remain consistent in both structures. Unification fails if the two expressions cannot be aligned, either by having dissimilar structure, or by having instance conflicts, such as a variable needs to represent both "xs:string" and "xs:integer". For a full explanation of ***type unification***, please see [**Hancock**].

1.1.2 Related terms

In the field of access control and authorization there are several closely related terms in common use. For purposes of precision and clarity, certain of these terms are not used in this specification.

For instance, the term ***attribute*** is used in place of the terms: group and role.

In place of the terms: privilege, permission, authorization, entitlement and right, we use the term ***rule.***

The term object is also in common use, but we use the term ***resourc***e in this specification.

Requestors and initiators are covered by the term ***subject***.

1.2. Notation

This specification contains schema conforming to W3C XML Schema and normative text to describe the syntax and semantics of XML-encoded policy statements.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 **[RFC2119]**

"they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)"

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

```
Listings of XACML schema appear like this.
```

```
[a01]  Example code listings appear like this.
```

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces as follows, whether or not a namespace declaration is present in the example:

- The prefix `xacml:` stands for the XACML policy namespace.
- The prefix `xacml-context:` stands for the XACML context namespace.
- The prefix `ds:` stands for the W3C XML Signature namespace **[DS]**.
- The prefix `xs:` stands for the W3C XML Schema namespace **[XS]**.
- The prefix `xf:` stands for the XQuery 1.0 and XPath 2.0 Function and Operators specification namespace **[XF]**.

This specification uses the following typographical conventions in text: `<XACMLElement>`, `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`. Terms in ***italic bold-face*** are intended to have the meaning defined in the Glossary.

1.3. Schema organization and namespaces

The XACML policy syntax is defined in a schema associated with the following XML namespace:

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

The XACML context syntax is defined in a schema associated with the following XML namespace:

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

2. Background (non-normative)

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.

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

2.1. 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 sections.

2.2. 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 re-used 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***.

Hinton et al [Hinton94] discuss the question of the compatibility of separate ***policies*** applicable to the same ***decision request***.

2.3. 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 specification may, if necessary, define their own combining algorithms.

2.4. 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.

2.5. 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 **[RBAC]**. 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*** (see Section 2.11 for an explanation of ***context***).

XACML provides a standard way to reference the ***attributes*** defined in the LDAP series of specifications **[LDAP-1, LDAP-2]**. 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.***

2.6. 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 (see Section A.3).

2.7. 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.

In cases where the information ***resource*** is not an XML document, specified ***attributes*** of the ***resource*** can be referenced, as described in Section 2.4.

2.8. Operators

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

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

XACML includes a number of built-in functions and a method of adding non-standard functions. These functions may be nested to build arbitrarily complex expressions. This is achieved with the `<Apply>` element. The `<Apply>` element has an XML attribute called `FunctionId` that identifies the function to be applied to the contents of the element. Each standard function is defined for specific argument data-type combinations, and its return data-type is also specified. Therefore, data-type consistency of the ***policy*** can be checked at the time the ***policy*** is written or parsed. 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 RFC822 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.

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

2.9. 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. But, XACML does not describe any normative way to do this. Regardless of the means of distribution, ***PDPs*** are expected to confirm, by examining the ***policy's*** `<Target>` element that the policy is applicable to the ***decision request*** that it is processing.

`<Policy>` elements may be attached to the information ***resources*** to which they apply, as described by Perritt [Perritt93]. 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***.

2.10. 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 as defined by the XACML specification.

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.

The use of constraints limiting the applicability of a ***policy*** were described by Sloman [Sloman94].

2.11. 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, etc.. 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***. But, 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 **[XSLT]**.

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

2.12. Actions performed in conjunction with enforcement

In many applications, policies specify actions that MUST be performed, either instead of, or in addition to, actions that MAY be performed. This idea was described by Sloman [Sloman94]. XACML provides facilities to specify actions that MUST be performed in conjunction with policy evaluation in the `<Obligations>` element. This idea was described as a provisional action by Kudo [Kudo00]. 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.

3. Models (non-normative)

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

3.1. Data-flow model

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

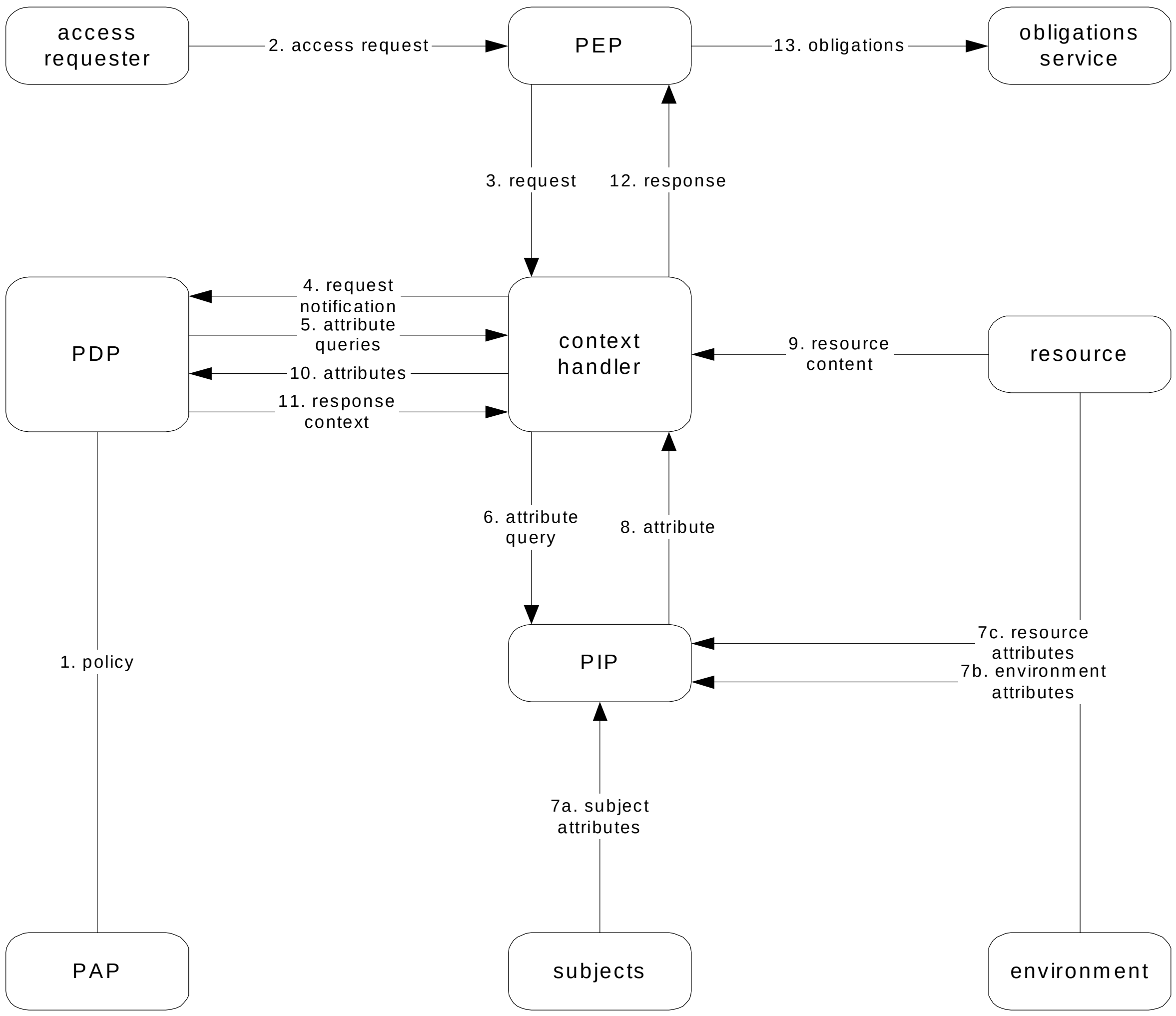

Figure 1 - Data-flow diagram

Note: some of the data-flows shown in the diagram may be facilitated by a repository. For instance, the communications between the ***context*** handler and the ***PIP*** or the communications between the ***PDP*** and the ***PAP*** may be facilitated by a repository. The XACML specification 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. ***PAP***s 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*** fulfills the ***obligations***.

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

3.2. XACML context

XACML is intended to be suitable for a variety of application environments. The core language is insulated from the application environment by the XACML ***context***, as shown in Figure 2, in which the scope of the XACML specification 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, J2SE, CORBA, and so on) and the ***attribute*** representations in the XACML ***context***. How this is achieved is outside the scope of the XACML specification. In some cases, such as SAML, this conversion may be accomplished in an automated way through the use of an XSLT transformation.

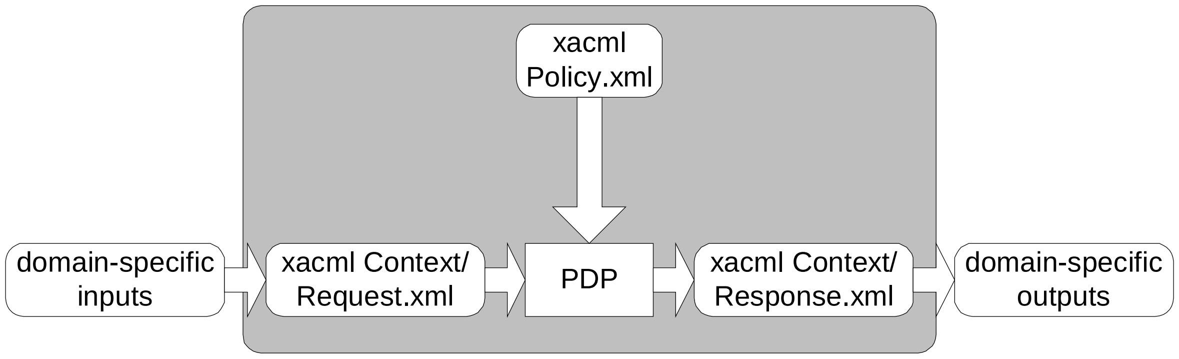

Figure 2 - XACML context

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

See Section 7.2.5 for a more detailed discussion of the request ***context***.

3.3. Policy language model

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

- ***Rule***;
- ***Policy***; and
- ***Policy set***.

These are described in the following sub-sections.

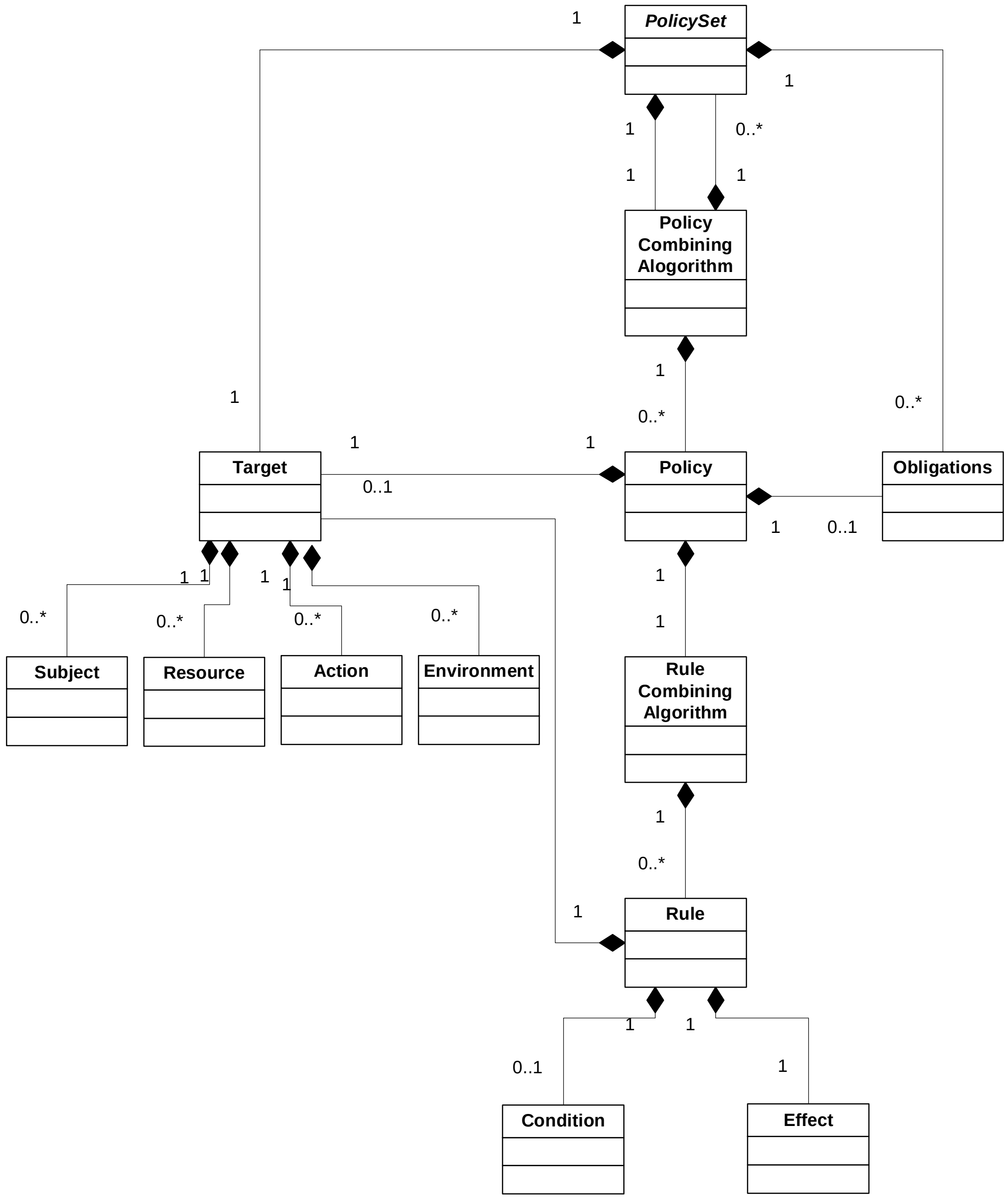

Figure 3 - Policy language model

3.3.1 Rule

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

- a ***target***;
- an ***effect*** and
- a ***condition***.

These are discussed in the following sub-sections.

3.3.1.1. Rule target

The ***target*** defines the set of:

- ***resource***s;
- ***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 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 RFC822 name "med.example.com" is a legal RFC822 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” (see Appendix A).

3.3.1.2. Effect

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

3.3.1.3. Condition

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

3.3.2 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***.

Rules are described above. The remaining components are described in the following sub-sections.

3.3.2.1. 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.

3.3.2.2. Rule-combining algorithm

The ***rule-combining algorithm*** specifies the procedure by which the results of evaluating the component ***rules*** are combined when evaluating the ***policy***, i.e. the `Decision` value placed in the response ***context*** by 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***.

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

3.3.2.3. 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***. Section 7.14 explains which ***obligations*** are to be returned.

3.3.3 Policy set

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

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

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

3.3.3.1. Policy-combining algorithm

The ***policy-combining algorithm*** specifies the procedure by which the results of evaluating the component ***policies*** are combined when evaluating the ***policy set***, i.e. the `Decision` value placed in the response ***context*** by 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***.

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

3.3.3.2. 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***. Section 7.14 explains which ***obligations*** are to be returned.

4. Examples (non-normative)

This section 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***.

4.1. Example one

4.1.1 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:cd"
[a05]  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a06]  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd.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 Appendix 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 *AND*ed or *OR*ed together).

[a37] closes the ***policy***. As mentioned above, this ***policy*** has only one ***rule***, but more complex ***policies*** may have any number of ***rules***.

4.1.2 Example request context

Let's 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:cd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a40] xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:cd
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-cd.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***[1]. 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`”.

[1] Some exceptions are described in the XACML Profile for Multiple Resources [MULT].

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".

4.1.3 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:cd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:cd
http://docs.oasis-open.org/xacml/xacml-core-2.0-context-schema-cd.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***.

4.2. Example two

This section 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***.

4.2.1 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>
```

4.2.2 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:cd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
urn:oasis:names:tc:xacml:2.0:context:schema:cd http://docs.oasis-
open.org/xacml/access_control-xacml-2.0-context-schema-cd.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.

4.2.3 Example plain-language rules

The following plain-language rules are to be enforced:

Rule 1: A person, identified by his or her patient number, may read any record for which he or she is the designated patient.

Rule 2: A person may read any record for which he or she is the designated parent or guardian, and for which the patient is under 16 years of age.

Rule 3: A physician may write to any medical element for which he or she is the designated primary care physician, provided an email is sent to the patient.

Rule 4: An administrator shall not be permitted to read or write to medical elements of a patient record.

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

4.2.4 Example XACML rule instances

4.2.4.1. 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:cd" xmlns:xacml-
context="urn:oasis:names:tc:xacml:2.0:context:schema:cd"
[a200] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
urn:oasis:names:tc:xacml:2.0:policy:schema:cd http://docs.oasis-
open.org/xacml/access_control-xacml-2.0-context-schema-cd.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 the 1.0 version of the XPath specification.

[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***.

4.2.4.2. 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:cd" xmlns:xacml-
context="urn:oasis:names:tc:xacml:2.0:context:schema:cd"
[a272] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd.xsd"
[a273] xmlns:xf="http://www.w3.org/TR/2002/WD-xquery-operators-20020816/#"
[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="http://www.w3.org/TR/2002/WD-xquery-operators-
20020816#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 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#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 Section 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 it’s 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.

4.2.4.3. 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:cd" xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:cd"
[a368]  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a369]  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd.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.

4.2.4.4. 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:cd"
[a483] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd.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.

`[a558] [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.

4.2.4.5. Example PolicySet

This section uses the examples of the previous sections 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 Section 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.

```
[a559] <?xml version="1.0" encoding="UTF-8"?>
[a560]  <PolicySet
[a561]  xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:cd"
[a562]  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd.xsd"
[a563]  PolicySetId=
[a564]  "urn:oasis:names:tc:xacml:2.0:example:policysetid:1"
[a565]  PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:
[a566]  policy-combining-algorithm:deny-overrides">
[a567]   <Description>
[a568]    Example policy set.
[a569]   </Description>
[a570]   <Target>
[a571]    <Resources>
[a572]     <Resource>
[a573]      <ResourceMatch
[a574]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a575]       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
[a576]        urn:example:med:schema:records
[a577]       </AttributeValue>
```

```
[a578]       <ResourceAttributeDesignator AttributeId=
[a579]       "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
[a580]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a581]      </ResourceMatch>
[a582]     </Resource>
[a583]    </Resources>
[a584]   </Target>
[a585]  <PolicyIdReference>
[a586]   urn:oasis:names:tc:xacml:2.0:example:policyid:3
[a587]  </PolicyIdReference>
[a588]  <Policy
[a589]  PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
[a590]  RuleCombiningAlgId=
[a591] "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
[a592]   <Target/>
[a593]   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1"
[a594]   Effect="Permit">
[a595]   </Rule>
[a596]   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:2"
[a597]   Effect="Permit">
[a598]   </Rule>
[a599]   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:4"
[a600]   Effect="Deny">
[a601]   </Rule>
[a602]  </Policy>
[a603] </PolicySet>
```

[a560] - [a566] The `<PolicySet> element` declaration. Standard XML namespace declarations are included.

[a563] The `PolicySetId` attribute is used for identifying this ***policy set*** for possible inclusion in another ***policy set***.

[a565]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.

[a567] - [a569] Free form description of the ***policy set***.

[a570] - [a584] The ***policy set*** `<Target>` element defines the set of ***decision requests*** that are applicable to this `<PolicySet>` element.

[a585] `PolicyIdReference` includes a ***policy*** by id.

[a589] `Policy 2` is explicitly included in this ***policy set***. The ***rules*** in `Policy 2` are omitted for clarity.

5. Policy syntax (normative, with the exception of the schema fragments)

5.1. Element <PolicySet>

The `<PolicySet>` element is a top-level element in the XACML policy schema. `<PolicySet>` is an aggregation of other ***policy sets*** and ***policies***. ***Policy sets*** MAY be included in an enclosing `<PolicySet>` element either directly using the `<PolicySet>` element or indirectly using the

`<PolicySetIdReference>` element. ***Policies*** MAY be included in an enclosing `<PolicySet>` element either directly using the `<Policy>` element or indirectly using the `<PolicyIdReference>` element.

A `<PolicySet>` element MAY be evaluated, in which case the evaluation procedure defined in Section 7.11 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***. See Section 7.11.

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 fulfill, any of the ***obligations***, then it MUST act as if the ***PDP*** had returned a “Deny” ***authorization decision*** value. See Section 7.14.

```
<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. Standard ***policy-combining algorithms*** are listed in Appendix C. Standard ***policy-combining algorithm*** identifiers are listed in Section B.10.

`<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. See Section 7.14 for a description of how the set of ***obligations*** to be returned by the ***PDP*** shall be determined.

`<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>`.

5.2. 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"/>
```

5.3. 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.

5.4. Element <XPathVersion>

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

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

The URI for the XPath 1.0 specification 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.

5.5. 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***.

5.6. 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]

See Section 5.7.

5.7. 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. See Section 5.8.

5.8. 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 documented in Section 7.5.

`<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***.

5.9. 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 Section 5.10.

5.10. 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. See Section 5.11.

5.11. 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 documented in Section 7.5.

`<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***.

5.12. 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 Section 5.13.

5.13. 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 Section 5.14.

5.14. 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 documented in Section 7.5.

<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***.

5.15. 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 Section 5.16.

5.16. 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.

5.17. 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 documented in Section A.3.

`<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***.

5.18. 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 specification.

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

```
        <xs:attribute name="xacml:LatestVersion" type="xacml:VersionMatchType"
use="optional"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

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

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

`Version` [Optional]

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

`EarliestVersion` [Optional]

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

`LatestVersion` [Optional]

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

The matching operation is defined in Section 5.21. 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.

5.19. 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 specification.

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

Element <PolicyIdReference> is of **xacml:IdReferenceType** complex type (see Section 5.18) .

5.20. 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.

5.21. Simple type VersionMatchType

Elements of this type SHALL contain a restricted regular expression matching a version number (see Section 5.20). 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.+'.

5.22. 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 Section 7.10 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***. See Section 7.10.

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. Standard ***rule-combining algorithms*** are listed in Appendix C. Standard ***rule-combining algorithm*** identifiers are listed in Section B.10.

`<Description>` [Optional]

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

`<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***. See Section 7.14 for a description of how the set of ***obligations*** to be returned by the ***PDP*** SHALL be determined.

5.23. 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.

5.24. 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 occurence 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. See Section 5.25.

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

5.25. 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.

5.26. 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 occurence 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 optional.

5.27. 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 occurence of the `<PolicyCominberParameters>` 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 optional.

5.28. 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 occurence 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 optional.

5.29. 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 Section 7.9 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 Section 7.6 for details.

`<Condition>` [Optional]

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

5.30. 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>
```

5.31. 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.

5.32. 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.

5.33. 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>.

5.34. 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”. Evaluation of the <Condition> element is described in Section 7.8.

5.35. 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. See Section 5.33.

```
<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 Appendix A.

`<Expression>` [Optional]

Arguments to the function, which may include other functions.

5.36. 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. The higher-order ***bag*** functions are described in Section A3A.3.12.

```
<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.

5.37. 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***. See Section 7.2.4.

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***. See Section 7.2.5. Also see Sections 7.15.2 and 7.15.3.

5.38. 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". See Section 7.2.5.

The **SubjectAttributeDesignatorType** extends the match semantics of the **AttributeDesignatorType** (See Section 5.37) 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. Standard values of the `SubjectCategory` are listed in Section B.2.

5.39. 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”. See Section 7.2.5.

A *named resource attribute* SHALL match a ***resource attribute*** as per the match semantics specified in the **AttributeDesignatorType** complex type. See Section 5.37.

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.

5.40. 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". See Section 7.2.5.

A *named* ***action attribute*** SHALL match an ***action attribute*** as per the match semantics specified in the **AttributeDesignatorType** complex type. See Section 5.37.

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.

5.41. 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". See Section 7.2.5.

A *named* ***environment attribute*** SHALL match an ***environment attribute*** as per the match semantics specified in the **AttributeDesignatorType** complex type. See Section 5.37.

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.

5.42. 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 **[XF]** or **[XS]**, then the value returned by the XPath expression SHALL be converted to the DataType specified in the <AttributeSelector> using the constructor function below [**XF** Section 4] that corresponds to the DataType. If an error results from using the constructor function, 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()
xf:yearMonthDuration()
xf:dayTimeDuration()

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. See also Section 5.4.

`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. See Section 7.2.5. Also see Sections 7.15.2 and 7.15.3.

5.43. 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.

5.44. 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***. See Section 5.45.

5.45. 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 Section 7.14 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***.

5.46. 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. See Section 7.14. Section 4.2.4.3 provides a number of examples of arguments included in ***obligations***.

```
<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.

6. Context syntax (normative with the exception of the schema fragments)

6.1. 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 document describes ***policy*** evaluation in terms of operations on the ***context***. However a conforming ***PDP*** is not required to actually instantiate the ***context*** in the form of an XML document. But, any system conforming to the XACML specification 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[2]. 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.

[2] The conditions under which multiple `<Resource>` elements are allowed are described in the XACML Profile for Multiple Resources [MULT].

`<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***.

6.2. 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.

6.3. 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.

6.4. 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.

6.5. 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***.

6.6. 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.

6.7. 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***. See Appendix B.

`DataType` [Required]

The data-type of the contents of the `<xacml-context:AttributeValue>` element. This SHALL be either a primitive type defined by the XACML 2.0 specification 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.

6.8. 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.

6.9. 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 the XACML specification 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 [MULT]. 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. See Section 6.10.

6.10. 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 fulfill 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 fulfill an ***obligation***, then it MUST act as if the ***PDP*** had denied ***access*** to the requested ***resource***. See Section 7.14 for a description of how the set of ***obligations*** to be returned by the PDP is determined.

6.11. 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***.

6.12. 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.

6.13. 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 Section B.9 for a list of values.

`<StatusCode>` [Any Number]

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

6.14. 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.

6.15. 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.

6.16. 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. Functional requirements (normative)

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

7.1. Policy enforcement point

This section 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 ***authorization decision*** as described in one of the following sub-sections

7.1.1. 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*** behavior is undefined.

If the ***decision*** is “Indeterminate”, then the ***PEP’s*** behavior is undefined.

7.1.2. 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.

7.1.3. 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.

7.2. 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.

7.2.1. Structured attributes

`<xacml:AttributeValue>` and `<xacml-context:AttributeValue>` elements MAY contain an instance of a structured XML data-type, for example `<ds:KeyInfo>`. XACML 2.0 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 “regexp-string-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 Section A.3. This method requires support by the ***PDP*** for the optional XPath expressions and XPath functions features.

See also Section 8.2.

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

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

7.2.3. 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.

7.2.4. 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`.

7.2.5. 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*** as described in Section 7.13. However, a ***PDP*** MAY choose not to return such information for security reasons.

7.2.6. Environment Attributes

Standard ***environment attributes*** are listed in Section B.8. 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***".

7.3. 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>`

7.4. Arithmetic evaluation

IEEE 754 [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 (IEEE 854 §7) with the exception of the “division-by-zero” trap enabler, which SHALL be set to 1

precision - is set to the designated double precision

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

7.5. 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:regexp-string-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:regexp-string-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. 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 1.

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"

Table 1 - Target match table

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 2. The ***resources***, ***actions*** and ***environments*** match tables are analogous.

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

Table 2 - Subjects match table

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 3. The ***resource***, ***action*** and ***environment*** match tables are analogous.

`<SubjectMatch>` values	`<Subject>` Value
All “True”	“Match”
No “False” and at least one “Indeterminate”	“Indeterminate”
At least one “False”	“No match”

Table 3 - Subject match table

7.7. 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.

7.8. 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 `<Condtion>` element evaluates to "Indeterminate."

7.9. 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 4.

Target	Condition	Rule Value
“Match”	“True”	Effect
“Match”	“False”	“NotApplicable”
“Match”	“Indeterminate”	“Indeterminate”
“No-match”	Don’t care	“NotApplicable”
“Indeterminate”	Don’t care	“Indeterminate”

Table 4 - Rule truth table

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.

7.10. 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 5.

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"

Table 5 - Policy truth table

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"; see Section 7.9). 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 XACML 2.0 take 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.

7.11. 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 6.

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”

Table 6 – Policy set truth table

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 take 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.

7.12. 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. These are described in the XACML Profile for Hierarchical Resources [HIER] and in the XACML Profile for Requests for Multiple Resources [MULT].

7.13. 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 Sections 5 and 7. 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.

7.14. 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.

Also, see Section 7.1.

7.15. Exception handling

XACML specifies behaviour for the ***PDP*** in the following situations.

7.15.1. 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.

7.15.2. 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".

7.15.3. 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, as described in Sections 5.37 and 5.42. 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 (see Section 6.16). 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.

8. XACML extensibility points (non-normative)

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

8.1. 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`.

See Section 5 for definitions of these attribute types.

8.2. Structured attributes

`<xacml:AttributeValue>` and `<xacml-context:AttributeValue>` elements MAY contain an instance of a structured XML data-type. Section 7.2.1 describes a number of standard techniques to identify data items within such a structured attribute. Listed here are some additional techniques that require XACML extensions.

1. For a given structured data-type, a community of XACML users MAY define new attribute identifiers for each leaf sub-element of the structured data-type that has a type conformant with one of the XACML-defined primitive data-types. Using these new attribute identifiers, 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.

9. Security and privacy considerations (non-normative)

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

9.1. 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 specification.

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 specification, 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 sections detail specific compromise scenarios that may be relevant to an XACML system.

9.1.1. 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.

9.1.2. 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.

9.1.3. 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.

9.1.4. 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.

9.1.5. 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.

9.1.6. 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. Note, however, that according to Section 7.1, a ***PEP*** must deny ***access*** unless it receives an explicit “Permit” ***authorization decision***.

9.1.7. 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.

9.2. Safeguards

9.2.1. 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.

9.2.2. 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.

9.2.3. 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.

9.2.3.1. 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.

9.2.3.2. Statement level confidentiality

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

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

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

9.2.4. 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 document.

9.2.5. 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 specification 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.

9.2.6. 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.

9.2.7. 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.

10. Conformance (normative)

10.1. Introduction

The XACML specification addresses the following aspect of conformance:

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

10.2.Conformance tables

This section lists those portions of the specification that MUST be included in an implementation of a ***PDP*** that claims to conform with XACML v2.0. A set of test cases has been created to assist in this process. These test cases are hosted by Sun Microsystems and can be located from the XACML Web page. The site hosting the test cases contains a full description of the test cases and how to execute them.

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

10.2.1. 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
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

10.2.2. 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

10.2.3. 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

10.2.4. 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

10.2.5. Attributes

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

10.2.6. 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 are transparent to the ***PDP***.

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

3873 ### 10.2.7. Data-types

3874 The implementation MUST support the data-types associated with the following identifiers marked
3875 "M".

Data-type	M/O
http://www.w3.org/2001/XMLSchema#string	M
http://www.w3.org/2001/XMLSchema#boolean	M
http://www.w3.org/2001/XMLSchema#integer	M
http://www.w3.org/2001/XMLSchema#double	M
http://www.w3.org/2001/XMLSchema#time	M
http://www.w3.org/2001/XMLSchema#date	M
http://www.w3.org/2001/XMLSchema#dateTime	M
http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration	M
http://www.w3.org/TR/2002/WD-xquery-operators-20020816#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

3876 ### 10.2.8. Functions

3877 The implementation MUST properly process those functions associated with the identifiers marked
3878 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
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
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:regexp-string-match	M
urn:oasis:names:tc:xacml:1.0:function:regexp-uri-match	M
urn:oasis:names:tc:xacml:1.0:function:regexp-ipAddress-match	M
urn:oasis:names:tc:xacml:1.0:function:regexp-dnsName-match	M
urn:oasis:names:tc:xacml:1.0:function:regexp-rfc822Name-match	M
urn:oasis:names:tc:xacml:1.0:function:regexp-x500Name-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
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
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

11. References

[DS] D. Eastlake et al., *XML-Signature Syntax and Processing*, **http://www.w3.org/TR/xmldsig-core/**, World Wide Web Consortium.

[Hancock] Hancock, "Polymorphic Type Checking", in Simon L. Peyton Jones, "Implementation of Functional Programming Languages", Section 8, Prentice-Hall International, 1987

[Haskell] Haskell, a purely functional language. Available at **http://www.haskell.org/**

[Hier] Anderson A, ed., *The XACML Profile for Hierarchical Resources,* OASIS Access Control TC, Committee Draft 01, 16 Sep 2004, http://www.oasis-open.org/committees/xacml

[Hinton94] Hinton, H, M, Lee,, E, S, The Compatibility of Policies, Proceedings 2nd ACM Conference on Computer and Communications Security, Nov 1994, Fairfax, Virginia, USA.

[IEEE754]	IEEE Standard for Binary Floating-Point Arithmetic 1985, ISBN 1-5593-7653-8, IEEE Product No. SH10116-TBR
[ISO10181-3]	ISO/IEC 10181-3:1996 Information technology – Open Systems Interconnection -- Security frameworks for open systems: Access control framework.
[Kudo00]	Kudo M and Hada S, XML document security based on provisional authorization, Proceedings of the Seventh ACM Conference on Computer and Communications Security, Nov 2000, Athens, Greece, pp 87-96.
[LDAP-1]	RFC2256, A summary of the X500(96) User Schema for use with LDAPv3, Section 5, M Wahl, December 1997 **http://www.ietf.org/rfc/rfc2798.txt**
[LDAP-2]	RFC2798, Definition of the inetOrgPerson, M. Smith, April 2000 **http://www.ietf.org/rfc/rfc2798.txt**
[MathML]	Mathematical Markup Language (MathML), Version 2.0, W3C Recommendation, 21 February 2001. Available at: **http://www.w3.org/TR/MathML2/**
[Multi]	Anderson A, ed., *The XACML Profile for Requests for Multiple Resources*, OASIS Access Control TC, Working Draft 02, 4 June 2004, http://www.oasis-open.org/committees/xacml
[Perritt93]	Perritt, H. Knowbots, Permissions Headers and Contract Law, Conference on Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, April 1993. Available at: **http://www.ifla.org/documents/infopol/copyright/perh2.txt**
[RBAC]	Role-Based Access Controls, David Ferraiolo and Richard Kuhn, 15th National Computer Security Conference, 1992. Available at: **http://csrc.nist.gov/rbac**
[RegEx]	XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001, Appendix D. Available at: **http://www.w3.org/TR/xmlschema-0/**
[RFC2119]	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, **http://www.ietf.org/rfc/rfc2119.txt**, IETF RFC 2119, March 1997
[RFC2396]	Berners-Lee T, Fielding R, Masinter L, Uniform Resource Identifiers (URI): Generic Syntax. Available at: http://www.ietf.org/rfc/rfc2396.txt
[RFC2732	Hinden R, Carpenter B, Masinter L, Format for Literal IPv6 Addresses in URL's. Available at: http://www.ietf.org/rfc/rfc2732.txt
[RFC3198]	IETF RFC 3198: Terminology for Policy-Based Management, November 2001. http://www.ietf.org/rfc/rfc3198.txt
[SAML]	Security Assertion Markup Language available from **http://www.oasis-open.org/committees/security/#documents**
[Sloman94]	Sloman, M. Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, Volume 2, part 4. Plenum Press. 1994.
[XACMLv1.0]	Extensible access control markup language (XACML) Version 1.0. OASIS Standard. 18 February 2003. Available at: **http://www.oasis-open.org/apps/org/workgroup/xacml/download.php/940/oasis-xacml-1.0.pdf**
[XACMLv1.1]	Extensible access control markup language (XACML) Version 1.1. OASIS Committee Specification. 7 August 2003. Available at: **http://www.oasis-open.org/apps/org/workgroup/xacml/download.php/4104/cs-xacml-specification-1.1.pdf**

[XF]	XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Working Draft 16 August 2002. Available at: **http://www.w3.org/TR/2002/WD-xquery-operators-20020816**
[XS]	XML Schema, parts 1 and 2. Available at: **http://www.w3.org/TR/xmlschema-1/** and **http://www.w3.org/TR/xmlschema-2/**
[XPath]	XML Path Language (XPath), Version 1.0, W3C Recommendation 16 November 1999. Available at: **http://www.w3.org/TR/xpath**
[XSLT]	XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999. Available at: **http://www.w3.org/TR/xslt**

Appendix A. Data-types and functions (normative)

A.1. Introduction

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

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

A.2. Data-types

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

- http://www.w3.org/2001/XMLSchema#string
- http://www.w3.org/2001/XMLSchema#boolean
- http://www.w3.org/2001/XMLSchema#integer
- http://www.w3.org/2001/XMLSchema#double
- http://www.w3.org/2001/XMLSchema#time
- http://www.w3.org/2001/XMLSchema#date
- http://www.w3.org/2001/XMLSchema#dateTime
- http://www.w3.org/2001/XMLSchema#anyURI
- http://www.w3.org/2001/XMLSchema#hexBinary
- http://www.w3.org/2001/XMLSchema#base64Binary
- http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration
- http://www.w3.org/TR/2002/WD-xquery-operators-20020816#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:1.0:data-type:ipAddress
- urn:oasis:names:tc:xacml:1.0: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. For integers and doubles, XACML SHALL use the conversions described in [IEEE754].

XACML defines three data-types; these are:

“urn:oasis:names:tc:xacml:1.0:data-type:x500Name”,

“urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name”

“urn:oasis:names:tc:xacml:1.0:data-type:ipAddress”

“urn:oasis:names:tc:xacml:1.0:data-type:dnsName” and

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

X.500 directory name

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

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, Section 4.1.2, Command Argument Syntax, under the term "Mailbox".

IP address

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

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

For an IPv4 address, the address and mask are formatted in accordance with the syntax for a "host" in IETF RFC 2396 "Uniform Resource Identifiers (URI): Generic Syntax", section 3.2.
For an IPv6 address, the address and mask are formatted in accordance with the syntax for an "ipv6reference" in IETF RFC 2732 "Format for Literal IPv6 Addresses in URL's". (Note that an IPv6 address or mask, in this syntax, is enclosed in literal "[" "]" brackets.)

DNS name

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

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

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

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

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

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

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”. It SHALL perform its evaluation on integers according to IEEE 754 [IEEE 754].

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

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

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

  This function SHALL take two arguments of data-type “http://www.w3.org/2001/XMLSchema#date” and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL perform its evaluation according to the “op:date-equal” function [**XF** Section 8.3.11].

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

  This function SHALL take two arguments of data-type “http://www.w3.org/2001/XMLSchema#time” and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL perform its evaluation according to the “op:time-equal” function [**XF** Section 8.3.14].

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

  This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to the "op:dateTime-equal" function [**XF** Section 8.3.8].

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

  This function SHALL take two arguments of data-type "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation according to the "op:dayTimeDuration-equal" function [**XF** Section 8.3.5]. Note that the lexical representation of each argument MUST be converted to a value expressed in fractional seconds [**XF** Section 8.2.2].

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

  This function SHALL take two arguments of data-type "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation according to the "op:yearMonthDuration-equal" function [**XF** Section 8.3.2]. Note that the lexical representation of each argument MUST be converted to a value expressed in integer months [**XF** Section 8.2.1].

- 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". It SHALL perform its evaluation according to the "op:anyURI-equal" function [**XF** Section 10.2.1].

- 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"[3].

  1. Normalize the two arguments according to IETF RFC 2253 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names".

  2. If any RDN contains multiple attributeTypeAndValue pairs, re-order the Attribute ValuePairs in that RDN in ascending order when compared as octet strings (described in ITU-T Rec. X.690 (1997 E) Section 11.6 "Set-of components").

  3. Compare RDNs using the rules in IETF RFC 3280 "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Section 4.1.2.4 "Issuer".

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

  This function SHALL take two arguments of data-type "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the two arguments are equal. Otherwise, it SHALL

[3] ITU-T Rec. X.520 contains rules for matching X500 names, but these are very complex and require knowledge of the syntax of various AttributeTypes. IETF RFC 3280 contains simplified matching rules that the XACML x500Name-equal function uses.

return “False”. An RFC822 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 [**XS** Section 8.2.15].

- 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 [**XS** Section 8.2.16].

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. Each function evaluation SHALL proceed as specified by their logical counterparts in IEEE 754 [IEEE 754]. 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”.

- 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 section 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. They SHALL comply with the rules governed by IEEE 754 [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 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration". It SHALL return a result of "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL return the value by adding the second argument to the first argument according to the specification of adding durations to date and time [**XS** 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 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration". It SHALL return a result of "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL return the value by adding the second argument to the first argument according to the specification of adding durations to date and time [**XS** 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 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#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 the specification [**XS** 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 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#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 the specification [**XS** 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 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration". It SHALL return a result of "http://www.w3.org/2001/XMLSchema#date". This function SHALL return the value by adding the second argument to the first argument according to the specification of adding duration to date [**XS** 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 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#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 specification [**XS** 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" [**XS** Section 3.2.8]. Otherwise, it SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

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

  This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first argument is greater than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" [**XS** Section 3.2.8]. Otherwise, it SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

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

  This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first argument is less than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" [**XS** Section 3.2.8]. Otherwise, it SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

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

  This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first argument is less than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" [**XS** Section 3.2.8]. Otherwise, it SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.

- urn:oasis:names:tc:xacml:1.0:function:time-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 [**XF** Section 3.2.7]. Otherwise, it

SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described in **[XF]**.

- 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 [**XF** Section 3.2.7**]**. Otherwise, it SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described in **[XF]**.

- 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 [**XF** Section 3.2.7]. Otherwise, it SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described in **[XF]**.

- 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 [**XF** Section 3.2.7]. Otherwise, it SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described in **[XF]**.

- 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” by [**XF** Section 3.2.9]. Otherwise, it SHALL return “False”. Note: if a date value does not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described in **[XF]**.

- 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” by [**XF** Section 3.2.9]. Otherwise, it SHALL return “False”. Note: if a date value does not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described in **[XF]**.

- 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” by [**XF** Section 3.2.9]. Otherwise, it SHALL return “False”. Note: if a date value does not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described in **[XF]**.

- 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” by [**XF** Section 3.2.9]. Otherwise, it SHALL return “False”. Note: if a date value does not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described in **[XF]**.

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 section describes functions in XACML that perform operations on ***bags*** such that functions may be applied to the ***bags*** in general.

In this section, a general-purpose functional language called Haskell **[Haskell]** is used to formally specify the semantics of these functions. Although the English description is adequate, a formal 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.

- 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".

  In Haskell, the semantics of this operation are 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.

- 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".

In Haskell, the semantics of this operation are as follows:

```
all_of :: ( a -> b -> Bool ) -> a -> [b] -> Bool
all_of  f  a  []    = "False"
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).

- 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 are 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***.

In Haskell, taking advantage of the "any_of" function defined above, the semantics of the "any_of_any" function are 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***.

- 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”.

In Haskell, taking advantage of the “any_of” function defined in Haskell above, the semantics of the “all_of_any” function are as follows:

```
all_of_any :: ( a -> b -> Bool ) -> [a ]-> [b] -> Bool
all_of_any   f  []   ys = "False"
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***.

- 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 ”rn: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”.

In Haskell, taking advantage of the “all_of” function defined in Haskell above, the semantics of the “any_of_all” function are 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.

- 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 are that the result of the expression is "True" if and only if the applied predicate is "True" for *all* elements of the first ***bag*** compared to *all* the elements of the second ***bag***.

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 = "False"
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***.

- 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.

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!".

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:regexp-string-match

  This function decides a regular expression match. It SHALL take two arguments of "http://www.w3.org/2001/XMLSchema#string" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular expression and the second argument SHALL be a general string. The function specification SHALL be that of the "xf:matches" function with the arguments reversed [XF Section 6.3.15].

- urn:oasis:names:tc:xacml:1.0:function:regexp-uri-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:regexp-string-match".

- urn:oasis:names:tc:xacml:1.0:function:regexp-ipAddress-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: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:regexp-string-match".

- urn:oasis:names:tc:xacml:1.0:function:regexp-dnsName-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: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:regexp-string-match".

- urn:oasis:names:tc:xacml:1.0:function:regexp-rfc822Name-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:regexp-string-match".

- urn:oasis:names:tc:xacml:1.0:function:regexp-x500Name-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:regexp-string-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 the following specification.

  An RFC822 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.[4]

  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. For example, if the first argument is "Anderson@sun.com", this matches a value in the second argument of "Anderson@sun.com" and "Anderson@SUN.COM", but not "Anne.Anderson@sun.com", "anderson@sun.com" or "Anderson@east.sun.com".

  In order to match any address at a particular domain in the second argument, the first argument must specify only a domain name (usually a DNS name). For example, if the first argument is "sun.com", this matches a value in the first argument of "Anderson@sun.com" or "Baxter@SUN.COM", but not "Anderson@east.sun.com".

  In order to match any address in a particular domain in the second argument, the first argument must specify the desired domain-part with a leading ".". For example, if the first argument is ".east.sun.com", this matches a value in the second argument of "Anderson@east.sun.com" and "anne.anderson@ISRG.EAST.SUN.COM" but not "Anderson@sun.com".

4 According to IETF RFC822 and its successor specifications [RFC2821], case is significant in the *local-part.* Many mail systems, as well as the IETF PKIX specification, treat the *local-part* as case-insensitive. This anomaly is considered an error by mail-system designers and is not encouraged. For this reason, rfc822Name-match treats *local-part* as case sensitive.

A.3.15 XPath-based functions

This section 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, according to the “op:node-equal” function [**XF** Section 13.1.6], any of the XML nodes in the node-set matched by the second argument.

- 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, according to "op:node-equal" [XF Section 13.1.6], 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, according to "op:node-equal" [XF Section 13.1.6], to any of the XML nodes in the node-set matched by the second argument.

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.

Appendix B. XACML identifiers (normative)

This section 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:cd`

Request and response ***contexts*** are defined using this identifier.

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

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 Section A.2.

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

The following data-type identifiers are defined by XML Schema **[XS]**.

```
http://www.w3.org/2001/XMLSchema#string
http://www.w3.org/2001/XMLSchema#boolean
http://www.w3.org/2001/XMLSchema#integer
```

```
http://www.w3.org/2001/XMLSchema#double
http://www.w3.org/2001/XMLSchema#time
http://www.w3.org/2001/XMLSchema#date
http://www.w3.org/2001/XMLSchema#dateTime
http://www.w3.org/2001/XMLSchema#anyURI
http://www.w3.org/2001/XMLSchema#hexBinary
http://www.w3.org/2001/XMLSchema#base64Binary
```

The following data-type identifiers correspond to the dayTimeDuration and yearMonthDuration data-types defined in [**XF** Sections 8.2.2 and 8.2.1, respectively].

```
http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration
http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration
```

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 **[SAML]**.

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 **[LDAP-1, LDAP-2]**, the XACML identifier SHALL be formed by adding the ***attribute*** name to the URI of the LDAP specification. For example, the ***attribute*** name for the userPassword defined in the RFC 2256 SHALL be:

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

B.6. 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.7. 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.8. 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.9. Status codes

The following status code values are defined.

This identifier indicates success.

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

This identifier indicates that all the attributes necessary to make a policy decision were not available (see Section 6.16).

`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.10.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`

Appendix C. Combining algorithms (normative)

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

C.1. Deny-overrides

The following specification 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;
}
```

The following specification 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 Section 7.14.

C.2. Ordered-deny-overrides

The following specification defines the "Ordered-deny-overrides" ***rule-combining algorithm*** of a ***policy***.

> The behavior 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 specification defines the "Ordered-deny-overrides" ***policy-combining algorithm*** of a ***policy set***.

> The behavior 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

The following specification 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;
}
```

The following specification 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 Section 7.14.

C.4. Ordered-permit-overrides

The following specification defines the "Ordered-permit-overrides" ***rule-combining algorithm*** of a ***policy***.

> The behavior 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 specification defines the "Ordered-permit-overrides" ***policy-combining algorithm*** of a ***policy set***.

> The behavior 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

The following specification 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;
}
```

The following specification 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 Section 7.14.

C.6. Only-one-applicable

The following specification 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;
  }
}
```

Appendix D. Acknowledgments

The following individuals contributed to the development of the specification:

Anne Anderson
Anthony Nadalin
Bill Parducci
Carlisle Adams
Daniel Engovatov
Don Flinn
Ed Coyne
Ernesto Damiani
Frank Siebenlist
Gerald Brose
Hal Lockhart
James MacLean
John Merrells
Ken Yagen
Konstantin Beznosov
Michiharu Kudo
Michael McIntosh
Pierangela Samarati
Pirasenna Velandai Thiyagarajan
Polar Humenn
Rebekah Lepro
Ron Jacobson
Satoshi Hada
Sekhar Vajjhala
Seth Proctor
Simon Godik
Steve Anderson
Steve Crocker
Suresh Damodaran
Tim Moses
Von Welch

Appendix E. Revision history

Rev	Date	By whom	What
CD 01	16 Sep 2004	Access Control TC	First committee draft

Appendix F. Notices