The OASIS XML-based Access-Control Markup Language (XACML)

Document identifier: draft-xacml-v0.10

Location: http://www.oasis-open.org/committees/xacml/docs

Publication date: 8 Mar 2002

Maturity Level: Committee Draft

Send comments to: xacml-comment@lists.oasis-open.org

Editors:

Tim Moses, Entrust
Simon Godik, Simon Godik

Contributors:

Anne Anderson, Sun Microsystems
Bill Parducci, Bill Parducci
Carlisle Adams, Entrust
Ernesto Damiani, University of Milan
Hal Lockhart, Entegrity
Ken Yagen, Crosslogix
Michiharu Kudo, IBM, Japan
Pierangela Samarati, University of Milan
Polar Humenn, University of Syracuse
Sekhar Vajjhala, Sun Microsystems

Table of contents

1. Glossary (non-normative)

1.1. Preferred terms

Access - Performing an ***action*** on a ***resource***

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

Action - An operation that may be performed on a ***resource***

Applicable policy - The complete set of ***rules*** that governs ***access*** for a specific ***decision request***

Attribute - Characteristic of a ***subject***, ***resource, action, environment or rule*** that may be referenced by a ***rule***

Authorization decision - The result of evaluation of an ***applicable policy.*** A function that evaluates to "permit, deny or indeterminate", and (optionally) a set of ***post-conditions***

Condition - An expression of ***predicates*** over ***attributes*** of ***resource, subject*** and ***environment***

Context - The intended use of information revealed as a result of ***access***.

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

Environment - The set of ***attributes*** that may be referenced by ***rules*** and that are independent of a particular ***subject, resource, action or rule***

Information request - The request by a ***PDP*** to a ***PIP*** for one or more ***environment attributes***

Meta-policy - A procedure for combining ***rules*** to form a ***policy***

Meta-policy one - A ***meta-policy*** defined in the current version of XACML

Policy - A set of ***rules*** and an identifier for the ***meta-policy*** for combining the ***rules***

Policy administration point (PAP) - The system entity that creates a ***rule***

Policy decision point (PDP) - The system entity that evaluates ***applicable policy*** and renders an ***authorization decision***

Policy enforcement point (PEP) - The system entity that performs ***access control***, by enforcing ***authorization decisions***

Policy information point (PIP) - The system entity that acts as the source of ***environment attributes***

Policy mediation point (PMP) - The system entity that resolves conflict amongst ***rules***

P*olicy retrieval point (PRP)* - The system entity that creates ***applicable policy*** from ***rules*** by the application of ***meta-policy***

Post-condition - A process specified in a ***policy*** that must be completed in conjunction with ***access***. There are two types of post-condition: an ***internal post-condition*** must be performed by the ***PDP*** in conjunction with the issuance of a "permit" response, and an ***external post-condition*** must be performed by the ***PEP*** in conjunction with permitting ***access***

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

Resource - Data, service, or system component

Rule - A ***target*** and a set of ***conditions*** with an associated ***meta-policy*** identifier

Subject - An actor whose **attributes** may be referenced in a **rule**

Target - The set of ***decision requests***, identified by definitions of ***resource***, ***subject*** and ***action*** to which a ***rule*** applies

Target mapping - The process of confirming that a ***rule*** is applicable to a given ***target***.

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***, ***role***, ***privilege***, ***permission***, ***right***, ***authorization*** and ***entitlement***.

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***.

2. Introduction (non-normative)

This specification defines the syntax and semantics for XML-encoded access control rule statements and policy statements. The XACML schema is an extension schema for SAML.

The following sections describe how to understand the rest of this specification.

2.1. 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 schemas appear like this.
```

```
Example code listings appear like this.
```

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces (see Section 2.2) as follows, whether or not a namespace declaration is present in the example:

- The prefix `saml:` stands for the SAML assertion namespace.
- The prefix `ds:` stands for the W3C XML Signature namespace.
- The prefix `xs:` stands for the W3C XML Schema namespace in example listings. In schema listings, this is the default namespace and no prefix is shown.

This specification uses the following typographical conventions in text: `<SAMLElement>`, `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`.

2.2. Schema Organization and Namespaces

The XACML policy statement structures are defined in a schema associated with the following XML namespace:

`http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-09.xsd`

Comment: Page: 1
This will need to be edited before final release

Note: The XACML namespace names are temporary and may change when XACML 1.0 is finalized.

The SAML assertion schema is imported into the XACML schema. Also imported is the schema for XML Signature XMLSigXSD, which is associated with the following XML namespace:

`http://www.w3.org/2000/09/xmldsig#`

3. Example (non-normative)

3.1. Introduction to the example

This section contains an example of the application of XACML rules to a medical record. Four separate rules are created.

3.2. Example medical record instance

Following is an instance of a medical record to which the example rules can be applied. The "record" schema is defined in the registered namespace administered by "//medico.com".

```
<?xml version="1.0" encoding="UTF-8"?>
<record xmlns="medico.com/records.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="medico.com/records.xsd
D:\MYDOCU~1\Standards\XACML\record.xsd">
   <patient>
      <patientName>
         <first>Bartholomew</first>
         <last>Simpson</last>
      </patientName>
      <patientContact>
         <street>27 Shelbyville Road</street>
         <city>Springfield</city>
```

```
            <state>MA</state>
            <zip>12345</zip>
            <phone>555.123.4567</phone>
            <fax/>
            <email/>
         </patientContact>
         <patientDoB xsi:type="date">1992-03-21</patientDoB>
         <patientGender xsi:type="string">male</patientGender>
         <policyNumber xsi:type="string">555555</policyNumber>
      </patient>
      <parentGuardian>
         <parentGuardianName>
            <first>Homer</first>
            <last>Simpson</last>
         </parentGuardianName>
         <parentGuardianContact>
            <street>27 Shelbyville Road</street>
            <city>Springfield</city>
            <state>MA</state>
            <zip>12345</zip>
            <phone>555.123.4567</phone>
            <fax/>
            <email>homers@aol.com</email>
         </parentGuardianContact>
      </parentGuardian>
      <primaryCarePhysician>
         <physicianName>
            <first>Julius</first>
            <last>Hibbert</last>
         </physicianName>
         <physicianContact>
            <street>1 First St</street>
            <city>Springfield</city>
            <state>MA</state>
            <zip>12345</zip>
            <phone>555.123.9012</phone>
            <fax>555.123.9013</fax>
            <email/>
         </physicianContact>
         <registrationID>ABC123</registrationID>
      </primaryCarePhysician>
      <insurer>
         <name>Blue Cross</name>
         <street>1234 Main St</street>
         <city>Springfield</city>
         <state>MA</state>
         <zip>12345</zip>
         <phone>555.123.5678</phone>
         <fax>555.123.5679</fax>
         <email/>
      </insurer>
      <medical>
         <treatment>
            <drug>
               <name>methylphenidate hydrochloride</name>
               <dailyDosage>30mgs</dailyDosage>
               <startDate>1999-01-12</startDate>
            </drug>
            <comment>patient exhibits side-effects of skin coloration
and carpal degeneration</comment>
         </treatment>
         <result>
            <test>blood pressure</test>
```

```
          <value>120/80</value>
          <date>2001-06-09</date>
          <performedBy>Nurse Betty</performedBy>
       </result>
   </medical>
</record>
```

3.3. Example authorization decision request

The following example illustrates a SAML authorization decision request to which the example rules are intended to be applicable. It represents a request by Julius Hibbert to read the patient date of birth in the record of Bartholomew Simpson. It includes an authentication assertion and an attribute assertion.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request RequestID="47823081" MajorVersion="0" MinorVersion="24"
xmlns="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-protocol-24.xsd"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-protocol-24.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\draft-sstc-schema-protocol-
24.xsd">
   <AuthorizationDecisionQuery
Resource="//medico.com/record/patient[@patientName/first='Bartholome
w'][@patientName/last='Simpson']/patientDoB">
      <saml:Subject>
         <saml:NameIdentifier SecurityDomain="medico.com"
Name="Julius Hibbert"/>
      </saml:Subject>
      <saml:Actions>
         <saml:Action>read</saml:Action>
      </saml:Actions>
      <saml:Evidence>
         <Assertion AssertionID="64578390" Issuer="medico.com"
IssueInstant="2002-03-08T08:23:47-05:00" MajorVersion="0"
MinorVersion="24" xmlns="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-24.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\draft-sstc-schema-assertion-
24.xsd">
            <AuthenticationStatement AuthenticationInstant="2002-
03-08T08:23:45-05:00" AuthenticationMethod="http://www.oasis-
open.org/committees/security/docs/draft-sstc-core-24/password-sha1">
               <Subject>
                  <NameIdentifier SecurityDomain="medico.com"
Name="Julius Hibbert"/>
                  <SubjectConfirmation>
                     <ConfirmationMethod>http://www.oasis-
open.org/committees/security/docs/draft-sstc-core-
24/artifact</ConfirmationMethod>
                  </SubjectConfirmation>
               </Subject>
               <AuthenticationLocality IPAddress="217.57.95.242"/>
            </AuthenticationStatement>
         </Assertion>
      </saml:Evidence>
```

```
      <saml:Evidence>
         <Assertion MajorVersion="0" MinorVersion="24"
AssertionID="68938960" Issuer="medico.com" IssueInstant="2000-06-
15T15:02:39-05:00" xmlns="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-24.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\draft-sstc-schema-assertion-
24.xsd">
            <AttributeStatement>
               <Subject>
                  <NameIdentifier SecurityDomain="medico.com"
Name="Julius Hibbert"/>
               </Subject>
                  <Attribute AttributeName="role"
AttributeNamespace="//medico.com">
                        <AttributeValue>
                              <role
xmlns="http://www.w3.org/2001/XMLSchema/string">physician</role>
                        </AttributeValue>
                  </Attribute>
            </AttributeStatement>
         </Assertion>
      </saml:Evidence>
   </AuthorizationDecisionQuery>
</Request>
```

3.4. Example plain-language rules

The following plain-language rules are to be enforced:

1. A person may read any record for which he or she is the designated patient.

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

3. A physician may write any medical element for which he or she is the designated primary care physician.

4. An administrator may read and write any record during office hours and from a designated IP address sub-space.

These rules may be written by different PAPs, operating independently, or by a single PAP. Note that XACML instances are intended to be conveyed in SAML assertions, which provide elements for rule and policy meta-data, such as "issuer" and "issueInstant". However, this detail has been omitted from the following examples.

3.5. Example XACML rule instances

3.5.1. Rule 1

The following XACML ruleStatement instance expresses Rule 1.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ruleStatement ruleId="//medico.com/rules/rule1"
metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
policies/meta-policy-1" xmlns="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~4.XSD">
   <comment>A person may read any record for which he or she is the
designated patient</comment>
   <target>
      <subjects>
         <saml:Attribute AttributeName="RFC822Name"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>
               <rfc822Name
xmlns="http://www.w3.org/2001/XMLSchema/string">*</rfc822Name>
            </saml:AttributeValue>
         </saml:Attribute>
      </subjects>
      <resources>
         <saml:Attribute AttributeName="documentURI"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>
               <documentURI
xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
.*</documentURI>
            </saml:AttributeValue>
         </saml:Attribute>
      </resources>
      <actions>
         <saml:Actions>
            <saml:Action>read</saml:Action>
         </saml:Actions>
      </actions>
   </target>
   <effect>permitIf</effect>
   <condition>
      <equal>
         <saml:AttributeDesignator AttributeName="requestor"
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
         <saml:AttributeDesignator AttributeName="patientName"
AttributeNamespace="//medico.com/record/patient/"/>
      </equal>
   </condition>
</ruleStatement>
```

3.5.2. Rule 2

The following XACML ruleStatement instance expresses Rule 2.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ruleStatement ruleId="//medico.com/rules/rule2"
metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
policies/meta-policy-1" xmlns="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~4.XSD">
   <comment>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</comment>
   <target>
      <subjects>
         <saml:Attribute AttributeName="RFC822Name"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>
               <rfc822Name
xmlns="http://www.w3.org/2001/XMLSchema/string">*</rfc822Name>
            </saml:AttributeValue>
         </saml:Attribute>
      </subjects>
      <resources>
         <saml:Attribute AttributeName="documentURI"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>
               <documentURI
xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
.*</documentURI>
            </saml:AttributeValue>
         </saml:Attribute>
      </resources>
      <actions>
         <saml:Actions>
            <saml:Action>read</saml:Action>
         </saml:Actions>
      </actions>
   </target>
   <effect>permitIf</effect>
   <condition>
      <and>
         <equal>
            <saml:AttributeDesignator AttributeName="requestor"
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
            <saml:AttributeDesignator
AttributeName="parentGuardianName"
AttributeNamespace="//medico.com/record/parentGuardian/"/>
         </equal>
         <greaterOrEqual>
            <saml:AttributeDesignator AttributeName="patientDoB"
AttributeNamespace="//medico.com/record/patient/"/>
            <saml:Attribute AttributeName="dateOfBirth"
AttributeNamespace="//medico.com">
               <saml:AttributeValue>
                  <dateOfBirth>1986-03-08</dateOfBirth>
               </saml:AttributeValue>
            </saml:Attribute>
         </greaterOrEqual>
      </and>
   </condition>
</ruleStatement>
```

Notes:

The approach to the patient's age is clearly unsatisfactory as it stands, because it requires the rule to be updated daily with a new date. It will be necessary to include arithmetic operators in the language. Another option is to use an external function for the calculation.

3.5.3. Rule 3

The following XACML ruleStatement instance expresses Rule 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleStatement ruleId="//medico.com/rules/rule3"
metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
policies/meta-policy-1" xmlns="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~4.XSD">
   <comment>A physician may write any medical element for which he
or she is the designated primary care physician</comment>
   <target>
      <subjects>
         <saml:Attribute AttributeName="role"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>
               <role
xmlns="http://www.w3.org/2001/XMLSchema/string">physician</role>
            </saml:AttributeValue>
         </saml:Attribute>
      </subjects>
      <resources>
         <saml:Attribute AttributeName="documentURI"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>
               <documentURI
xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
/medical.*</documentURI>
            </saml:AttributeValue>
         </saml:Attribute>
      </resources>
      <actions>
         <saml:Actions>
            <saml:Action>write</saml:Action>
         </saml:Actions>
      </actions>
   </target>
   <effect>permitIf</effect>
   <condition>
      <equal>
         <saml:AttributeDesignator AttributeName="requestor"
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
         <saml:AttributeDesignator AttributeName="physicianName"
AttributeNamespace="//medico.com/record/primaryCarePhysician/"/>
      </equal>
   </condition>
</ruleStatement>
```

Notes:

The post condition applies only to its immediately preceding sibling rule. While it may seem more desirable to have an element that explicitly encloses the post-condition with the rule to which it applies, this solution would lead to the inclusion of unnecessary tags in XACML instances when no post-condition is present.

3.5.4. Rule 4

The following XACML ruleStatement instance expresses Rule 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleStatement ruleId="//medico.com/rules/rule4"
metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
policies/meta-policy-1" xmlns="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~4.XSD">
   <comment>An administrator may read and write any record during
office hours and from a designated IP address sub-space</comment>
   <target>
      <subjects>
         <saml:Attribute AttributeName="role"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>
               <role
xmlns="http://www.w3.org/2001/XMLSchema/string">administrator</role>
            </saml:AttributeValue>
         </saml:Attribute>
      </subjects>
      <resources>
         <saml:Attribute AttributeName="documentURI"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>
               <documentURI
xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
.*</documentURI>
            </saml:AttributeValue>
         </saml:Attribute>
      </resources>
      <actions>
         <saml:Actions>
            <saml:Action>read write</saml:Action>
         </saml:Actions>
      </actions>
   </target>
   <effect>permitIf</effect>
   <condition>
      <and>
         <greaterOrEqual>
            <saml:AttributeDesignator AttributeName="timeOfDay"
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
            <saml:Attribute AttributeName="time"
AttributeNamespace="http://www.w3.org/2001/XMLSchema/">
               <saml:AttributeValue>
                  <timeOfDay>08:00</timeOfDay>
               </saml:AttributeValue>
            </saml:Attribute>
         </greaterOrEqual>
         <lessOrEqual>
```

```
            <saml:AttributeDesignator AttributeName="timeOfDay"
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
            <saml:Attribute AttributeName="time"
AttributeNamespace="http://www.w3.org/2001/XMLSchema/">
                <saml:AttributeValue>
                   <timeOfDay>17:00</timeOfDay>
                </saml:AttributeValue>
            </saml:Attribute>
         </lessOrEqual>
         <patternMatch>
            <saml:AttributeDesignator
AttributeName="authenticationLocality" AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
            <saml:Attribute AttributeName="authenticationLocality"
AttributeNamespace="//medico.com">
                <saml:AttributeValue>
                   <authenticationLocality
xmlns="http://www.w3.org/2001/XMLSchema/string">217.*</authenticatio
nLocality>
                </saml:AttributeValue>
            </saml:Attribute>
         </patternMatch>
      </and>
   </condition>
</ruleStatement>
```

Notes:

Pattern match uses the regular expression syntax.

4. Models (non-normative)

The context and schema of XACML are described in two models that describe different aspects of its operation. These models are: the data-flow model and the policy language model. They are described in the following sub-sections.

4.1. Data-flow model

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

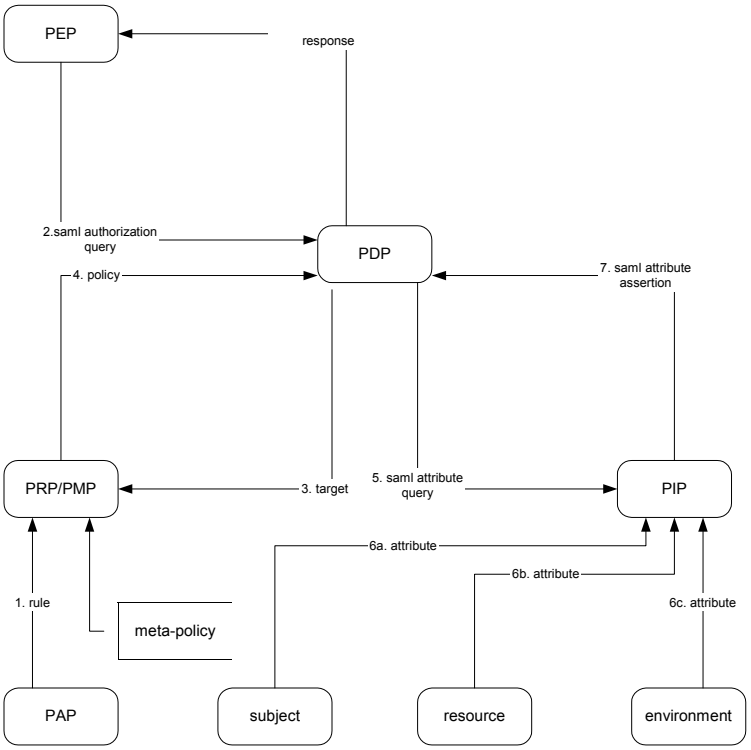

Figure 1 - Data-flow diagram

Some of the data-flows shown in the diagram may be facilitated by a repository. For instance, the communications between the PDP and the PIP may be facilitated by a repository, or the communications between the PDP and the PRP may be facilitated by a repository or the communication between the PAP and the PRP 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. PAPs write rules and make them available to the PRP. From the point of view of an individual PAP, its rule statements may represent the complete policy for a particular target. However, the PRP may be aware of other PAPs that it considers authoritative for the same target. In which case, it is the PRP's job to obtain all the rules and (if necessary) use a PMP to remove any conflict amongst those rules and combine the rules in accordance with a meta-policy. The result should be a self-consistent rule statement.

2. The PEP sends an authorization decision request to the PDP, in the form of a SAML [SAML] request. The decision request contains some or all of the attributes required by the PDP to render a decision, in accordance with policy.

3. The PDP locates and retrieves the policy statement applicable to the decision request from the PRP.

4. The PRP returns the complete policy to the PDP in the form of an XACML rule statement or policy instance. The PDP ensures that the decision request is in the scope of the policy statement or rule statement.

5. The PDP examines the authorization decision request and the policy statement to ascertain whether it has all the attribute values required to render an authorization decision. If it does not, then it requests attributes from suitable PIPs in the form of SAML requests of the attribute query type [SAML].

6. The PIP (which may be a SAML attribute authority) may locate and retrieve the requested attributes from other systems by a means, and in a form, that is out of scope for this specification.

7. The PIP returns the requested attributes to the PDP in the form of SAML responses containing SAML attribute assertions.

8. The PDP evaluates the policy statement.

9. If the policy statement were to evaluate to TRUE, then the PDP returns an authorization decision, in the form of a SAML response, to the PEP containing the "permit" decision attribute.

4.2. Rule combination

From the PRP's point of view, more than one rule may apply to a given target. For instance, taking the example rules described in Section 3, the "date of birth" element is governed by rules 1,2,3 and 4. Therefore, in plain language, the read rule applicable to "date of birth" is that at lease one of the following conditions must hold:

- The requestor is the patient;
- The requestor is the parent or guardian and the patient is under 16;
- The requestor is the primary care physician and a notification is sent to the patient; or
- The requestor is an administrator working from the office inside office hours.

XACML provides two alternative representations for combined rules. The first is a policy statement, which is a set of component rules and an identifier for the meta-policy, defining how the component rules are to be combined. The second is simply a rule statement containing the component rules explicitly combined as defined by the meta-policy.

4.2.1. Policy Statement

The combined rules may be formed by referencing the individual rules in a policy statement.

```
<?xml version="1.0" encoding="UTF-8"?>
<policyStatement policyId="//medico.com/policies/policy7"
metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
policies/meta-policy-1" xmlns="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~4.XSD">
   <comment>The set of rules that govern acess to a patient date of
birth</comment>
   <target>
      <subjects>
```

```
          <saml:Attribute AttributeName="RFC822Name"
AttributeNamespace="//medico.com">
             <saml:AttributeValue>
                <rfc822Name
xmlns="http://www.w3.org/2001/XMLSchema/string">*</rfc822Name>
             </saml:AttributeValue>
          </saml:Attribute>
       </subjects>
       <resources>
          <saml:Attribute AttributeName="documentURI"
AttributeNamespace="//medico.com">
             <saml:AttributeValue>
                <documentURI
xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
/patient/patientDoB</documentURI>
             </saml:AttributeValue>
          </saml:Attribute>
       </resources>
       <actions>
          <saml:Actions>
             <saml:Action>read</saml:Action>
          </saml:Actions>
       </actions>
    </target>
    <rule ruleID="//medico.com/rules/rule1"
authority="//medico.com"/>
    <rule ruleID="//medico.com/rules/rule2"
authority="//medico.com"/>
    <rule ruleID="//medico.com/rules/rule3"
authority="//medico.com"/>
    <rule ruleID="//medico.com/rules/rule4"
authority="//medico.com"/>
</policyStatement>
```

4.2.2. Rule statement

Alternatively, the policy may be formed by creating a new rule statement that includes all the individual rule conditions, combined according to the meta-policy.

```
<?xml version="1.0" encoding="UTF-8"?>
<policyStatement policyId="//medico.com/policies/policy1"
metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
policies/meta-policy-1" xmlns="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~1.XSD">
    <comment>The set of policies that govern acess to a patient date
of birth</comment>
    <target>
       <subjects>
          <saml:Attribute AttributeName="RFC822Name"
AttributeNamespace="//medico.com">
             <saml:AttributeValue>
                <rfc822Name
xmlns="http://www.w3.org/2001/XMLSchema/string">*</rfc822Name>
             </saml:AttributeValue>
          </saml:Attribute>
       </subjects>
```

```
      <resources>
         <saml:Attribute AttributeName="documentURI"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>
               <documentURI
xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
/patient/patientDoB</documentURI>
            </saml:AttributeValue>
         </saml:Attribute>
      </resources>
      <actions>
         <saml:Actions>
            <saml:Action>read</saml:Action>
         </saml:Actions>
      </actions>
   </target>
   <effect>permitIf</effect>
   <condition>
      <or>
         <equal>
            <saml:AttributeDesignator AttributeName="requestor"
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
            <saml:AttributeDesignator AttributeName="patientName"
AttributeNamespace="//medico.com/record/patient/"/>
         </equal>
         <and>
            <equal>
               <saml:AttributeDesignator AttributeName="requestor"
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
               <saml:AttributeDesignator
AttributeName="parentGuardianName"
AttributeNamespace="//medico.com/record/parentGuardian/"/>
            </equal>
            <greaterOrEqual>
               <saml:AttributeDesignator
AttributeName="patientDoB"
AttributeNamespace="//medico.com/record/patient/"/>
               <saml:Attribute AttributeName="dateOfBirth"
AttributeNamespace="//medico.com">
                  <saml:AttributeValue>
                        <dateOfBirth>1986-03-08</dateOfBirth>
                  </saml:AttributeValue>
               </saml:Attribute>
            </greaterOrEqual>
         </and>
         <equal>
            <saml:AttributeDesignator AttributeName="requestor"
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
            <saml:AttributeDesignator
AttributeName="physicianName"
AttributeNamespace="//medico.com/record/primaryCarePhysician/"/>
         </equal>
         <and>
            <greaterOrEqual>
               <saml:AttributeDesignator AttributeName="timeOfDay"
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
               <saml:Attribute AttributeName="time"
AttributeNamespace="http://www.w3.org/2001/XMLSchema/">
                  <saml:AttributeValue>
```

```
                        <timeOfDay>08:00</timeOfDay>
                    </saml:AttributeValue>
                </saml:Attribute>
            </greaterOrEqual>
            <lessOrEqual>
                <saml:AttributeDesignator AttributeName="timeOfDay"
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
                <saml:Attribute AttributeName="time"
AttributeNamespace="http://www.w3.org/2001/XMLSchema/">
                    <saml:AttributeValue>
                        <timeOfDay>17:00</timeOfDay>
                    </saml:AttributeValue>
                </saml:Attribute>
            </lessOrEqual>
            <patternMatch>
                <saml:AttributeDesignator
AttributeName="authenticationLocality" AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/"/>
                <saml:Attribute
AttributeName="authenticationLocality"
AttributeNamespace="//medico.com">
                    <saml:AttributeValue>
                        <authenticationLocality
xmlns="http://www.w3.org/2001/XMLSchema/string">217.*</authenticatio
nLocality>
                    </saml:AttributeValue>
                </saml:Attribute>
            </patternMatch>
        </and>
    </or>
  </condition>
</policyStatement>
```

4.3. Policy language model

The policy language model is shown in Figure 2.

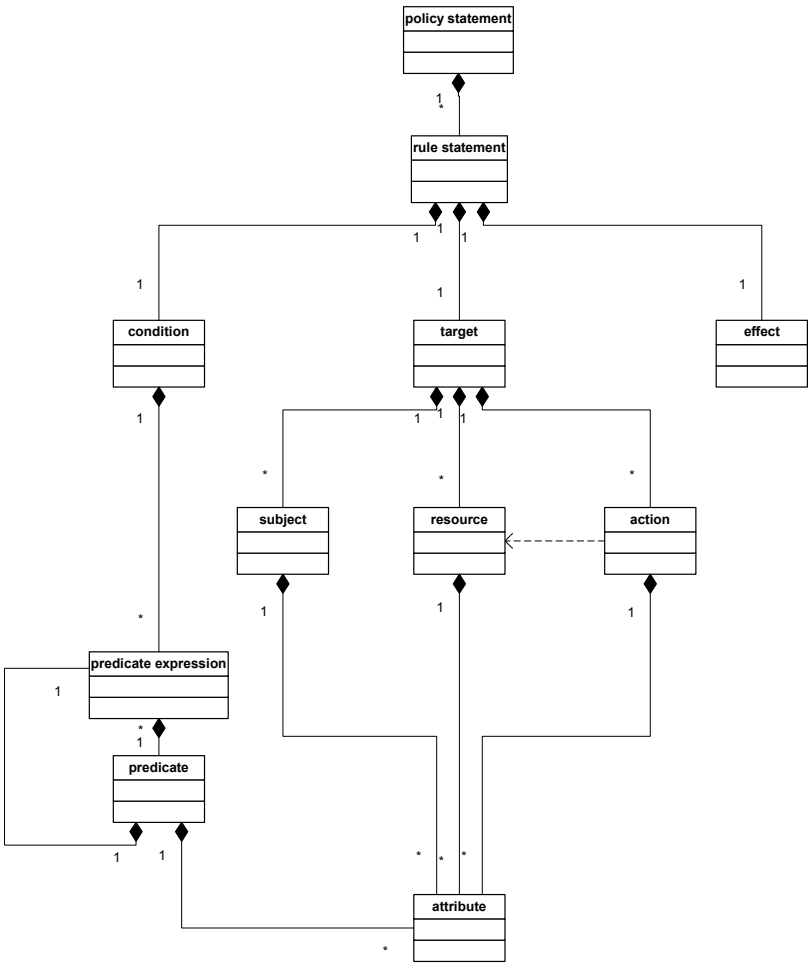

Figure 2 - Policy language model

A rule statement contains:

- a ruleId;
- a target;
- an effect;
- a metaPolicyRef and
- a condition.

Target defines the set of names for the:

- resources;
- subjects and

- actions

to which the rule statement applies. If the rule statement applies to all entities of a particular type, then the target definition is the root of the applicable name space. An XACML PDP MUST verify that the resources, subjects and actions identified in an authorization decision request are each included in the target of the rule statement that it uses to evaluate the request.

MetaPolicyRef specifies the meta-policy by which the rule statement MAY be combined with other rule statements.

Condition is a general expression of predicates of attributes. It SHOULD NOT duplicate the exact predicates implied by target. Therefore, it MAY be null.

Rule statements may be combined provided that their metaPolicyRefs are identical and that their targets are in identical name spaces. This SHALL be done by including the conditions from the component rule statements in the condition of the combined rule statement, in accordance with the meta-policy. (Section 7 for an example of a meta-policy).

The target of the combined rule statement can be calculated from the targets of the component rule statements. Two approaches are permitted:

- the target of the combined rule statement MAY contain the unions of the target definitions for resource, subject and action that are contained in the component rule statements; or
- the target of the combined rule statement MAY contain the intersections of the definitions.

In the former case, the targets from the source rule statements MUST be included in the form of conditions in the combined rule statement. In the latter case, this step MAY be omitted. In the former case, inapplicable predicates should not be encountered. In the latter case, inapplicable predicates may be encountered.

The effect of the combined rule statement is defined by the meta-policy.

The syntax of a combined rule statement is identical to that of a component rule statement.

There is an alternative representation for a combined rules, called a policy statement. It contains:

- a policyId;
- a target;
- a metaPolicyRef and
- a set of rule references.

A rule reference identifies a rule statement by meta-data, (e.g. ruleID).

A policy statement can be converted to a rule statement at any time, by applying the meta-policy identified in the rule statements to the rules. The resulting rule statement MUST contain a target computed from the target values in the actual versions of the component rule statements used in the conversion, whereas, the target in the policy statement SHOULD be computed from the target values in the actual versions of the component rule statements at the time the policy statement was prepared.

The significance of the two representations is that, in the first case, the combined rule statement is generated at the time of preparing the representation, whereas, in the second case, the combined rule statement can be generated (using the latest versions of the component rule statements) immediately prior to evaluation. PDPs SHOULD support both representations.

4.3.1. Target

The target of a rule statement identifies the set of authorization decision requests that the rule statement is intended to evaluate. It contains definitions for resource, subject and action. In order for a rule statement to apply to a specific authorization decision request, the resource, subject and action identified in the request MUST be contained in the resource, subject and action definitions in the target. Target definitions are discrete, in order that they may be indexed by the PDP.

Targets conform to a data model. XACML does not specify the data model, but it MUST be agreed between the PAP and the PDP. The data model MUST be semi-hierarchical. That is, it MUST have one or more disjoint trees for resources and one or more for subjects. Actions are leaf nodes of the resource node to which they apply. Each level in the tree is identified with an attribute type. A "path" is a list of attribute-type/value pairs linking a node to the root. The form of a target is a set of paths, one or more for each tree in the data model.

A node MAY have more than one target associated with it.

An authorization decision request also specifies a set of paths by (directly or indirectly) providing resource, subject and action attribute values.

A rule statement is applicable to a request if and only if every path in the target is part of a path in the request.

A PAP MAY store a rule statement in a repository, whose data model is congruent with that of the target data model, by storing it at the lowest node of every path in the target. In practice, the rule statements may be referenced from these nodes rather than stored at them.

When rule statements are combined, the combined target must be computed, and the repository must be updated. Rule statements that conform to different target data models MUST NOT be combined.

The combined target SHALL be computed by separately combining trees of the same type from each of the original targets. The combination may be in the form of a union or an intersection.

A union combination retains all of the original paths. If, as the result, all possible paths containing a particular node are retained, then the path may be truncated at that node.

An intersection combination retains a path from one target if and only if it includes a path from the other target.

The combined rule statement SHOULD be stored at the lowest node of every retained path.

Some attributes have an internal tree structure (e.g. DNS names). A sub-tree of this structure should be represented by a regular expression. When such an attribute defines a level in a target tree, the sub-tree defined by each node at that level SHALL be attached at that node.

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

5.1. Element <policytStatement>

The <policyStatement> element is a top-level element in the XACML schema.

```
<xs:element name="policyStatement" type="xacml:PolicyStatementType"/>
```

5.2. Element <ruleStatement>

The <ruleStatement> element is a top-level element in the XACML schema.

```
    <xs:element name="ruleStatement" type="xacml:RuleStatementType"/>
```

5.3. Complex type PolicyStatementType>

Elements of type PolicyStatementType extend the saml:StatementAbstractType so that they can be included in a <saml:Assertion>. The <saml:Assertion> element contains some policy meta-data. The main elements of this type definition are the <rule> and <target> elements and the metaPolicyRef attribute. The <rule> element contains references to the set of rules that are to be combined in a policy. The <target> element is computed from the target elements of the referenced <rule> elements either as an intersection or as a union. And the metaPolicyRef attribute contains a reference to the meta-policy by which this <policyStatement> element may be combined with other <policyStatement> elements.

```
    <xs:complexType name="PolicyStatementType">
       <xs:complexContent>
          <xs:extension base="saml:StatementAbstractType">
             <xs:sequence>
                <xs:element name="comment" type="xs:string" minOccurs="0"/>
                <xs:element name="target" type="xacml:TargetType"/>
                <xs:element name="rule" type="xacml:RuleRefType"
maxOccurs="unbounded"/>
             </xs:sequence>
             <xs:attribute name="policyId" type="xs:anyURI" use="required"/>
             <xs:attribute name="policyName" type="xs:string" use="optional"/>
             <xs:attribute name="metaPolicyRef" type="xs:anyURI" use="required"/>
          </xs:extension>
       </xs:complexContent>
    </xs:complexType>
```

5.4. Complex type RuleStatementType

Like the PolicyStatementType, the RuleStatementType extends the saml:StatementAbstractType. The main elements of this type definition are <target>, <effect>, <condition> and the metaPolicyRef attribute. The metaPolicyRef attribute identifies the meta-policy by which an element of this type may be combined with other elements of the type.

```
    <xs:complexType name="RuleStatementType">
       <xs:complexContent>
          <xs:extension base="saml:StatementAbstractType">
             <xs:sequence>
                <xs:element name="comment" type="xs:string" minOccurs="0"/>
                <xs:element name="target" type="xacml:TargetType"/>
                <xs:element name="effect" type="xacml:EffectType"/>
                <xs:element name="condition"
type="xacml:PredicateExpressionType" minOccurs="0"/>
             </xs:sequence>
             <xs:attribute name="ruleId" type="xs:anyURI" use="required"/>
             <xs:attribute name="ruleName" type="xs:string" use="optional"/>
             <xs:attribute name="metaPolicyRef" type="xs:anyURI" use="required"/>
          </xs:extension>
       </xs:complexContent>
    </xs:complexType>
```

5.5. Complex type TargetType

Target identifies the set of samlp:Requests of type samlp:AuthorizationDecisionQuery that the <ruleStatement> is intended to address. It contains definition for subject, resource and action. If the subject,

resource and action identified in the <samlp:Request> are contained within the definition, then the <ruleStatement> MAY be used to evaluate the <samlp:Request>.

```
<xs:complexType name="TargetType">
   <xs:sequence>
      <xs:element name="subjects" type="xacml:SubjectsType"/>
      <xs:element name="resources" type="xacml:ResourcesType"/>
      <xs:element name="actions" type="xacml:ActionsType"/>
   </xs:sequence>
</xs:complexType>
```

5.6. Complex type SubjectType

Elements of type SubjectType identify a set of subjects by a set of <saml:Attribute> elements.

```
<xs:complexType name="SubjectsType">
   <xs:sequence maxOccurs="unbounded">
      <xs:element ref="saml:Attribute"/>
   </xs:sequence>
</xs:complexType>
```

5.7. Complex type ResourceType

Elements of type ResourceType identify a set of resources by a list of <saml:Attribute> elements.

```
<xs:complexType name="ResourcesType">
   <xs:sequence maxOccurs="unbounded">
      <xs:element ref="saml:Attribute"/>
   </xs:sequence>
</xs:complexType>
```

5.8. Complex type ActionType

Elements of type ActionType identify a set of actions by a list of <saml:Action> elements.

```
<xs:complexType name="ActionsType">
   <xs:sequence>
      <xs:element ref="saml:Actions"/>
   </xs:sequence>
</xs:complexType>
```

5.9. Simple type EffectType

EffectType is an enumeration of "permitIf", "permitOnlyIf" and "denyIf". The attribute indicates the effect of the <ruleStatement> should it evaluate to TRUE. The relationship between the SAML Decision attribute issued by the PDP, the effect of the <ruleStatement> and the value of the <ruleStatement> is determined by meta-policy.

```
<xs:simpleType name="EffectType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="permitIf"/>
      <xs:enumeration value="permitOnlyIf"/>
      <xs:enumeration value="denyIf"/>
   </xs:restriction>
</xs:simpleType>
```

5.10. Complex type PredicateExpressionType

Elements of type PredicateExpressionType contain either a <predicateExpression> or <predicate> element. These elements are both of abstract type. So, an element of this type may only contain an element in one of their substitution groups.

```
    <xs:complexType name="PredicateExpressionType">
        <xs:choice>
            <xs:element ref="xacml:predicateExpression"/>
            <xs:element ref="xacml:predicate"/>
        </xs:choice>
    </xs:complexType>
```

5.11. Element <predicateExpression>

This is the base element for predicate expression elements, such as <and> and <or>.

```
    <xs:element name="predicateExpression"
type="xacml:PredicateExpressionAbstractType" abstract="true"/>
```

5.12. Complex type PredicateExpressionAbstractType

```
    <xs:complexType name="PredicateExpressionAbstractType"/>
```

5.13. Element <predicate>

This is the base element for predicate elements, such as <equal> and <greaterOREqual>.

```
    <xs:element name="predicate" type="xacml:PredicateAbstractType"
abstract="true"/>
```

5.14. Complex type PredicateAbstractType

This is an XACML extensibility point. New predicates may be added in the substitution group of <predicate>.

```
    <xs:complexType name="PredicateAbstractType"/>
```

5.15. Element <and>

The <and> element is an element of type AndType.

```
    <xs:element name="and" type="xacml:AndType"
substitutionGroup="xacml:predicateExpression"/>
```

5.16. Element <or>

The <or> element is an element of type OrType.

```
    <xs:element name="or" type="xacml:OrType"
substitutionGroup="xacml:predicateExpression"/>
```

5.17. Element <orderedOr>

The <orderedOr> element is an element of type OrderedOrType.

```
    <xs:element name="orderedOr" type="xacml:OrderedOrType"
substitutionGroup="xacml:predicateExpression"/>
```

5.18. Element <nOf>

The <nOf> element is an element of type NOfType.

```
    <xs:element name="nOf" type="xacml:NOfType"
substitutionGroup="xacml:predicateExpression"/>
```

5.19. Element <not>

The <not> element is an element of type NotType.

```
   <xs:element name="not" type="xacml:NotType"
substitutionGroup="xacml:predicateExpression"/>
```

5.20. Complex type AndType

Elements of type AndType can be evaluated with range true, false and indeterminate. They contain any combination of <predicateExpression> elements and <predicate> elements. Elements of this type evaluate to true if and only if all the elements contained in them evaluate to true.

```
   <xs:complexType name="AndType">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="xacml:predicateExpression"/>
            <xs:element ref="xacml:predicate"/>
         </xs:choice>
      </xs:sequence>
   </xs:complexType>
```

5.21. Complex type OrType

Elements of type OrType can be evaluated with range true, false and indeterminate. They contain any combination of <predicateExpression> elements and <predicate> elements. Elements of this type evaluate to true if one or more of the elements contained in them evaluate to true.

```
   <xs:complexType name="OrType">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="xacml:predicateExpression"/>
            <xs:element ref="xacml:predicate"/>
         </xs:choice>
      </xs:sequence>
   </xs:complexType>
```

5.22. Complex type OrderedOrType

Elements of type OrderedOrType can be evaluated with range true, false and indeterminate. They contain any combination of <predicateExpression> elements and <predicate> elements. Elements of this type evaluate to true if one or more of the elements contained in them evaluate to true. The <predicateExpression> elements and <predicate> elements MUST be evaluated in the order in which they appear, and evaluation MAY halt once one has evaluated to true.

```
   <xs:complexType name="OrderedOrType">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="xacml:predicateExpression"/>
            <xs:element ref="xacml:predicate"/>
         </xs:choice>
      </xs:sequence>
   </xs:complexType>
```

5.23. Complex type NofType

Elements of type NOfType can be evaluated with range true, false and indeterminate. They contain any combination of <predicateExpression> elements and <predicate> elements and a quorum attribute. Elements of this type evaluate to true if a number of the elements contained in them equal to the value of quorum evaluate to true.

```
   <xs:complexType name="NOfType">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="xacml:predicateExpression"/>
            <xs:element ref="xacml:predicate"/>
         </xs:choice>
      </xs:sequence>
      <xs:attribute name="quorum" type="xs:positiveInteger"/>
   </xs:complexType>
```

5.24. Complex type NotType

Elements of type NotType can be evaluated with range true, false and indeterminate. They contain a choice of a <predicateExpression> element or a <predicate> element. Elements of this type evaluate to true if the element contained in them evaluates to false.

```
   <xs:complexType name="NotType">
      <xs:choice>
         <xs:element ref="xacml:predicateExpression" minOccurs="0"/>
         <xs:element ref="xacml:predicate" minOccurs="0"/>
      </xs:choice>
   </xs:complexType>
```

5.25. Element <present>

The <present> element is in the substitution group of <predicate>.

```
   <xs:element name="present" type="xacml:PresentType"
substitutionGroup="xacml:predicate"/>
```

5.26. Element <equal>

The <equal> element is in the substitution group of <predicate>. It is of type CompareType. It SHALL evaluate to true if and only if the attributes it contains are equal.

```
   <xs:element name="equal" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
```

5.27. Element <greaterOrEqual>

The <greaterOrEqual> element is in the substitution group of <predicate>. It is of type CompareType. It MUST evaluate to TRUE only if the attribute value referenced by the first element of the CompareType is greater than or equal to the attribute value referenced by the second element. The elements must be of the same type, which may be string, normalizedString, byte, unsignedByte, base64Binary, hexBinary, integer, positiveInteger, negativeInteger, nonNegativeInteger, nonPositiveInteger, int, unsignedInt, long, unsignedLong, short, unsignedShort, decimal, float, double, time, dateTime, duration, date, gMonth, gYear, gYearMonth, gDay, gMonthDay, Name or Qname.

```
   <xs:element name="greaterOrEqual" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
```

5.28. Element <lessOrEqual>

The <lessOrEqual> element is in the substitution group of <predicate>. It is of type CompareType. It MUST evaluate to TRUE only if the attribute value referenced by the first element of the CompareType is less than or equal to the attribute value referenced by the second element. The elements must be of the same type, which may be string, normalizedString, byte, unsignedByte, base64Binary, hexBinary, integer, positiveInteger, negativeInteger, nonNegativeInteger, nonPositiveInteger, int, unsignedInt, long, unsignedLong, short, unsignedShort, decimal, float, double, time, dateTime, duration, date, gMonth, gYear, gYearMonth, gDay, gMonthDay, Name or Qname.

```
   <xs:element name="lessOrEqual" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
```

5.29. Element <subset>

The <subset> element is in the substitution group of <predicate>. It is of type CompareType. It MUST evaluate to TRUE only if the value referenced by the first element of the CompareType is amongst the set of values referenced by the second element.

```
   <xs:element name="subset" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
```

5.30. Element <superset>

The <superset> element is in the substitution group of <predicate>. It is of type CompareType. It MUST evaluate to TRUE only if the value referenced by the first element of the CompareType contains the set of values referenced by the second element.

```
         <xs:element name="supersetOf" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
```

5.31. Element <patternMatch>

The <patternMatch> element is in the substitution group of <predicate>. It is of type CompareType. It MUST evaluate to TRUE only if the string referenced by the first element of the CompareType matches the pattern defined in the string referenced by the second element. The pattern must be defined as a regular expression.

```
   <xs:element name="patternMatch" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
```

5.32. Element <nonNullSetInterscetion>

The <nonNullSetIntersection> element is in the substitution group of <predicate>. It contains an element of type CompareType. It MUST evaluate to TRUE only if the set of values referenced by the two elements of the CompareType have at least one value in common.

```
   <xs:element name="nonNullSetIntersection" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
```

5.33. Complex type PresentType

Elements of type PresentType contain an element of type <saml:AttributeDesignator>. It MUST evaluate to TRUE only if the element obtained by resolving the <saml:AttributeDesignator> element exists.

```
   <xs:complexType name="PresentType">
      <xs:complexContent>
         <xs:extension base="xacml:PredicateAbstractType">
            <xs:sequence>
               <xs:element ref="saml:AttributeDesignator"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
```

5.34. Complex type CompareType

Elements of the type CompareType contain a pair of elements. The first is a <saml:AttributeDesignator> element and the second is either a <saml:AttributeDesignator> or a <saml:Attribute> element. The two elements MUST be of identical type and include an xsi:type attribute.

```
   <xs:complexType name="CompareType">
      <xs:complexContent>
         <xs:extension base="xacml:PredicateAbstractType">
            <xs:sequence>
               <xs:element ref="saml:AttributeDesignator"/>
               <xs:choice>
                  <xs:element ref="saml:AttributeDesignator"/>
                  <xs:element ref="saml:Attribute"/>
               </xs:choice>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
      <!-- XML operands in "set" operations MUST be of type xs:list -->
      <!-- XML operands in "inequality" operations MUST contain an xsi:type
attribute for which
   XACML defines a comparison algorithm -->
   </xs:complexType>
```

5.35. Complex type RuleRefType

Elements of type RuleRefType contain attributes by which instances of <ruleStatement> may be referenced.

```
   <xs:complexType name="RuleRefType">
      <xs:attribute name="ruleID" type="xs:anyURI"/>
      <xs:attribute name="authority" type="xs:anyURI"/>
   </xs:complexType>
```

6. XACML identifiers (normative)

This section defines standard identifiers for commonly-used entities.

6.1. Requestor

6.2. Time of day

6.3. Authentication locality

6.4. Meta-policy one

7. Meta-policy one (non-normative)

"Meta-policy one" is a meta-policy for defining the combining of rules into a policy. Only <ruleStatements> that have the value "//www.oasis-open.org/committees/xacml/docs/meta-policies/meta-policy-1" for their metaPolicyRef attribute SHALL be combined using this meta-policy.

```
<metaPolicyStatement metaPolicyRef="//www.oasis-
open.org/committees/xacml/docs/meta-policies/meta-policy-1">
   <and>
      <or>
         <policyRef>
            <policyId>//saml:assertion/ruleStatement
               [@metaPolicyRef="metaPolicy1"][@effect="permitIf"] |
[@effect="permitOnlyIf"]/condition
            </policyId>
         </policyRef>
      </or>
      <not>
         <or>
```

```
            <policyRef>
               <policyId>//saml:assertion/ruleStatement
                  [@metaPolicyRef="metaPolicy1"][@effect="denyIf"]/condition
            </policyId>
            </policyRef>
         </or>
      </not>
      <and>
         <policyRef>
            <policyId>//saml:assertion/ruleStatement
               [@metaPolicyRef="metaPolicy1"][@effect="permitOnlyif"]/condition
               </policyId>
         </policyRef>
      </and>
   </and>
   <effect>permitIf<effect>
</metaPolicyStatement>
```

8. Profiles (non-normative)

Information in this section is normative, but not mandatory to implement.

8.1. XACML

Describes subsets of XACML appropriate to general classes of problem

8.2. SAML

Describes the subset of SAML that is relevant to XACML

We need to specify SAML status codes for situations specific to XACML, such as:

- PDP has no policy for the requested target
- PDP cannot retrieve the required attributes

8.3. XML Digital Signature

Describes how XACML instances shall be integrity-protected in the case where XML DSig is used. PAPs MAY sign applicable policy. When a PRP combines applicable policies, it MAY sign the resulting applicable policy.

Issue: Should the identities and/or signatures of the PAPs be preserved in the composed policy?

8.4. LDAP

Describes an LDAP schema for the case where LDAP is used to distribute XACML

9. XACML extension points (non-normative)

Describes the points within the XACML model and schema where extensions can be added

10. Security and privacy (non-normative)

Vulnerabilities and safeguards

11. References

[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

[SAML] Security Assertion Markup Language available from http://www.oasis-open.org/committees/security/#documents

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

[XMLSig-XSD] XML Signature Schema available from http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd.

12. Schema (normative)

This appendix contains the XACML schema definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.oasis-
open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
xmlns:xacml="http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-
policy-10.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-
assertion-24.xsd" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:import namespace="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-24.xsd"
schemaLocation="D:\My Documents\Standards\Xacml\v10 schema\draft-sstc-schema-
assertion-24.xsd"/>
   <xs:element name="policyStatement" type="xacml:PolicyStatementType"/>
   <xs:element name="ruleStatement" type="xacml:RuleStatementType"/>
   <xs:complexType name="PolicyStatementType">
      <xs:complexContent>
         <xs:extension base="saml:StatementAbstractType">
            <xs:sequence>
               <xs:element name="comment" type="xs:string" minOccurs="0"/>
               <xs:element name="target" type="xacml:TargetType"/>
               <xs:element name="rule" type="xacml:RuleRefType"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="policyId" type="xs:anyURI" use="required"/>
            <xs:attribute name="policyName" type="xs:string" use="optional"/>
            <xs:attribute name="metaPolicyRef" type="xs:anyURI" use="required"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="RuleStatementType">
      <xs:complexContent>
         <xs:extension base="saml:StatementAbstractType">
            <xs:sequence>
               <xs:element name="comment" type="xs:string" minOccurs="0"/>
               <xs:element name="target" type="xacml:TargetType"/>
               <xs:element name="effect" type="xacml:EffectType"/>
               <xs:element name="condition"
type="xacml:PredicateExpressionType" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="ruleId" type="xs:anyURI" use="required"/>
            <xs:attribute name="ruleName" type="xs:string" use="optional"/>
            <xs:attribute name="metaPolicyRef" type="xs:anyURI" use="required"/>
         </xs:extension>
      </xs:complexContent>
```

```
   </xs:complexType>
   <xs:complexType name="TargetType">
      <xs:sequence>
         <xs:element name="subjects" type="xacml:SubjectsType"/>
         <xs:element name="resources" type="xacml:ResourcesType"/>
         <xs:element name="actions" type="xacml:ActionsType"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="SubjectsType">
      <xs:sequence maxOccurs="unbounded">
         <xs:element ref="saml:Attribute"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="ResourcesType">
      <xs:sequence maxOccurs="unbounded">
         <xs:element ref="saml:Attribute"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="ActionsType">
      <xs:sequence>
         <xs:element ref="saml:Actions"/>
      </xs:sequence>
   </xs:complexType>
   <xs:simpleType name="EffectType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="permitIf"/>
         <xs:enumeration value="permitOnlyIf"/>
         <xs:enumeration value="denyIf"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="PredicateExpressionType">
      <xs:choice>
         <xs:element ref="xacml:predicateExpression"/>
         <xs:element ref="xacml:predicate"/>
      </xs:choice>
   </xs:complexType>
   <xs:element name="predicateExpression"
type="xacml:PredicateExpressionAbstractType" abstract="true"/>
   <xs:complexType name="PredicateExpressionAbstractType"/>
   <xs:element name="and" type="xacml:AndType"
substitutionGroup="xacml:predicateExpression"/>
   <xs:element name="or" type="xacml:OrType"
substitutionGroup="xacml:predicateExpression"/>
   <xs:element name="orderedOr" type="xacml:OrderedOrType"
substitutionGroup="xacml:predicateExpression"/>
   <xs:element name="nOf" type="xacml:NOfType"
substitutionGroup="xacml:predicateExpression"/>
   <xs:element name="not" type="xacml:NotType"
substitutionGroup="xacml:predicateExpression"/>
   <xs:complexType name="AndType">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="xacml:predicateExpression"/>
            <xs:element ref="xacml:predicate"/>
         </xs:choice>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="OrType">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="xacml:predicateExpression"/>
            <xs:element ref="xacml:predicate"/>
         </xs:choice>
```

```
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="OrderedOrType">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="xacml:predicateExpression"/>
            <xs:element ref="xacml:predicate"/>
         </xs:choice>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="NOfType">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="xacml:predicateExpression"/>
            <xs:element ref="xacml:predicate"/>
         </xs:choice>
      </xs:sequence>
      <xs:attribute name="quorum" type="xs:positiveInteger"/>
   </xs:complexType>
   <xs:complexType name="NotType">
      <xs:choice>
         <xs:element ref="xacml:predicateExpression" minOccurs="0"/>
         <xs:element ref="xacml:predicate" minOccurs="0"/>
      </xs:choice>
   </xs:complexType>
   <xs:element name="predicate" type="xacml:PredicateAbstractType"
abstract="true"/>
   <!--This is an XACML extensibility point.  New predicates may be added in the
      substitution group of "predicate"-->
   <xs:complexType name="PredicateAbstractType"/>
   <xs:element name="present" type="xacml:PresentType"
substitutionGroup="xacml:predicate"/>
   <xs:element name="equal" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
   <xs:element name="greaterOrEqual" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
   <xs:element name="lessOrEqual" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
   <xs:element name="subset" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
   <xs:element name="superset" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
   <xs:element name="patternMatch" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
   <xs:element name="nonNullSetIntersection" type="xacml:CompareType"
substitutionGroup="xacml:predicate"/>
   <xs:complexType name="PresentType">
      <xs:complexContent>
         <xs:extension base="xacml:PredicateAbstractType">
            <xs:sequence>
               <xs:element ref="saml:AttributeDesignator"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="CompareType">
      <xs:complexContent>
         <xs:extension base="xacml:PredicateAbstractType">
            <xs:sequence>
               <xs:element ref="saml:AttributeDesignator"/>
               <xs:choice>
                  <xs:element ref="saml:AttributeDesignator"/>
                  <xs:element ref="saml:Attribute"/>
```

```
                </xs:choice>
             </xs:sequence>
          </xs:extension>
       </xs:complexContent>
       <!-- XML operands in "set" operations MUST be of type xs:list -->
       <!-- XML operands in "inequality" operations MUST contain an xsi:type
attribute for which
    XACML defines a comparison algorithm -->
    </xs:complexType>
    <xs:complexType name="RuleRefType">
       <xs:attribute name="ruleID" type="xs:anyURI"/>
       <xs:attribute name="authority" type="xs:anyURI"/>
    </xs:complexType>
</xs:schema>
```

13. Conformance (normative)

Not the test cases themselves, but a description of how the test cases should be used. The test cases will be a set of files on the XACML Web site

Conformance claims MAY be made by either one of two components in the XACML model:

1. An implementation of a policy administration points that produces policy statements that conform with the XACML schema; and

2. An implementation of a policy decision point that produces decisions in response to decision requests on the basis of XACML policy statements that conform with the XACML schema.

In the current version of the specification, implementations of a policy retrieval point that produce policy statements that confomr with the XACML schema by combining XACML applicable policies are treated in the same way as policy administration points, from the point of view of conformance.

Policy administration points MAY claim conformance with the XACML specification provided merely that they produce schema-compliant policy statements.

Policy decision points MAY claim conformance with the XACML specification provided that they correctly execute the XACML conformance test suite provided:

http://www.oasis-open.org/ …

XACML Test Suite

The test suite comprises three directories:

- Decision requests
- Policies
- Authentication and attribute assertions
- Decision assertions

The decision requests directory contains a set of text/xml/samlp files that are valid SAML authorization decision request messages.

The polices directory contains precisely one XACML policy file whose target includes includes each of the decision requests.

The assertions directory contains an unordered set of text/xml/saml files containing the attributes required to evaluates the policies in the policies directory.

The decisions directory contains an unordered set of tect/xml/samlp files that are valid SAML authorization decision responses.

A conformant XACML PDP implementation shall create a decision assertion in response to each and every decision request. The decision responses are linked to the corresponding decision requests by the request ID attribute.

XACML implementations that target an application domain other than SAML may use a tool or process that is not an integral part of the implementation to convert between the SAML test vectors and its private data representation.

Disclaimer: Implementors SHALL NOT consider the test cases provided in the XACML conformance test suite as providing 100% test coverage. OASIS does not represent that a conformant implementation will operate correctly in all respects nor that it is fit for its purpose.

Appendix A. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.