Web Services Profile of XACML (WS-XACML) Version 1.0

Working Draft 5, 9 October 2006

Document identifier:
xacml-3.0-profile-webservices-spec-v1.0-wd-5

OASIS identifier:
[OASIS document number]

Location:
Persistent: [persistent location]
This Version: [location of this version]
Previous Version: [location of previous version]

Technical Committee:
OASIS XACML TC

Chair(s):
Hal Lockhart
Bill Parducci

Editor:
Anne Anderson

Subject / Keywords:
web services, policy, authorization, access control, credential, token, WS-Policy, privacy

OASIS Conceptual Model Topic Area:
Security

Related Work:

This specification is related to:

- eXtensible Access Control Markup Language (XACML) Version 2.0
- eXtensible Access Control Markup Language (XACML) Version 3.0
- Privacy policy profile of XACML v2.0

Abstract:
This document specifies ways to use XACML in the context of Web Services for authorization, access control, and privacy policies. It specifies three types of information. 1) An authorization

token or credential based on XACML to be used in a Web Services context for conveying an authorization decision from a trusted 3rd party to a Web Service. 2) An Assertion based on XACML for use with WS-Policy; this Assertion may be used to convey both requirements and capabilities related to authorization, access control, and privacy for Web Service clients and for the Services themselves. The profile specifies standard formats, matching semantics, and usage guidelines for this Assertion. 3) Some ways in which authenticated Attributes for a client MAY be passed to a Web Service as part of a SOAP message. These Attributes may be used by the Web Service in evaluating internal XACML policies.

Status:

This document was last revised or approved by the XACML on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at www.oasis-open.org/committees/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 Technical Committee web page (www.oasis-open.org/committees/xacml/ipr.php.

The non-normative errata page for this specification is located at www.oasis-open.org/committees/xacml.

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.

Table of Contents

1 Introduction

[All text is normative unless otherwise indicated.]

The Web Services model based on SOAP and WSDL is increasingly important to vendors and customers. While specifications have been defined for many aspects of Web Services, the authorization, access control, and privacy policy aspects have not yet been addressed. Since XACML is the approved standard in those domains, there is need for a standard way to use XACML to address authorization, access control, and privacy policy in a Web Services environment. This profile defines that standard.

Web Services Security specifies formats for authentication tokens, but does not address authorization tokens. This profile defines a format for XACML authorization decisions as authorization tokens or credentials in a Web Services environment.

Web Services Policy 1.5 – Framework (WS-Policy) is the emerging standard for expressing policies in the Web Services model. WS-Policy provides a framework for expressing alternative sets of policy Assertions from various domains, such as security, and reliable messaging, that are supported by a service. But there are no WS-Policy Assertions defined for authorization, access control, or privacy policies. This profile defines a format for such Assertions and describes their use in Web Services policies.

Here are some use cases for the formats and behaviors specified in this profile:

– A Web Service may not have access to the authorization policies used within a security domain. Another Web Service may operate on a constrained device that is unable to run a Policy Decision Point. Instead, such a Services depends on some trusted 3rd party Policy Decision Point (PDP) to provide a signed authorization token to a client, indicating that the 3rd party has evaluated the client's request against current policies, and has determined that the request is permitted. The client then presents this token to the Web Service in the wsse:Security header of the SOAP message at the time the Service is invoked. The Service verifies that the token correctly describes the controlled access that the client invocation requires, and then, based on the signature of the 3rd party, verifies that the token indeed came from a trusted 3rd party PDP, and, based on the validity period of the token, that the token is currently valid.

– A Web Service client may obtain Attributes from an Attribute Authority that is trusted by the Web Service, and wants to convey these Attributes to the Web Service as part of a SOAP message. The Web Service can then use the Attributes as input to its evaluation of authorization, access control, and privacy policies related to the interaction.

– A Web Service may require that clients be members of the "`XYZ Consortium`", and includes this requirement in an authorization Assertion as part of its published Web Services policy. Knowing this information, a client who is a member of the consortium can obtain an authenticatable Attribute attesting to its membership status, and can make the Attribute available to the Web Service at the time the Service is invoked. A client who is not a member can avoid trying to invoke that Web Service.

– A Web Service may require that `role` Attributes associated with a client be signed by a certain trusted 3rd party, and includes this requirement in an authorization Assertion as part of its published Web Services policy. A client who is unable to obtain Attributes signed by that 3rd party can save the effort of invoking that Web Service. Another client whose `role` Attributes are signed by that 3rd party can make those Attributes available to the Web Service, or can obtain Attributes signed by that 3rd party.

– A Web Service may require that a client have a `role` attribute with the value "`purchasing officer`". A client who is authorized to activate such a role can request activation from the Role

Activation Authority prior to invoking the Web Service. The Web Service can then query the Role Activation Authority to determine that the client has such a role activated.

- A client may intend to request certain information from a Web Service, and will be able to supply certain Attributes as part of its access request. The client can match its request against the Web Service's published authorization policy to see if the request is likely to be authorized when the service is invoked.
- A Web Service may impose an obligation on the clients of a particular interface to keep copies of the data it sends to the client encrypted, and includes this obligation as a requirement in an authorization Assertion that is published as part of its Web Services policy. If a client is unwilling or unable to fulfill this obligation, the client can avoid invoking the Web Service. Other clients can indicate to the Web Service that they have chosen a Web Services policy alternative that includes this obligation, indicating they are willing and able to fulfill the obligation.
- A client may impose an obligation on the Web Service not to share the client's personal identifying information with 3rd parties and to delete such information once the transaction has been completed or terminated. The client can include this obligation as part of its authorization Assertion in its Web Services policy, and can match its policy against the policies of potential service providers. The client can then choose to invoke a Web Service whose published authorization Assertion indicates that it is willing and able to fulfill this obligation.

Knowing these types of authorization, access control, and privacy requirements and capabilities ahead of time can allow a client and Web Service to decide whether it is desirable or even possible to engage in an interaction. Agreeing on a mutually acceptable policy alternative, and communicating that to the other party in a secure manner, can assure the other party that obligations are understood and will be fulfilled.

In many cases, a publicly available Web Service will want potential clients know what is required for authorization to use the Service. In many other cases, a Web Service will not want to advertise its full authorization, access control, and privacy policies for security or privacy reasons., but even in this case, the Web Service may choose to advertise a subset of its actual internal access control policy for use as a first-leve filter to minimize unproductive exchanges. Even a subset can help clients find Services they are more likely to be able to use, and can help clients learn what information they will need to make available to be authorized to use the Service; the subset can serve as a first step in a controlled information exchange.

To satisfy these use cases, this profile specifies how to use existing XACML SAML Assertions in the context of Web Services, and also defines one new schema element: `XACMLAuthzAssertion`. The `XACMLAuthzAssertion` allows clients and services to express their Web Service authorization requirements and capabilities, and to match requirements and capabilities between clients and services. The format and usage of this element is described in this profile.

This Profile MAY be used with any version of XACML that is supported by the Web Service components.

1.1 Terminology

The keywords "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 .

1.2 Normative References

[RFC 2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[Reference] [reference citation]

1.3 Non-normative References

[Reference] [reference citation]

[Reference] [reference citation]

2 XACML Authorization Token

This section of the profile describes how to use the existing `xacml-saml:XACMLAuthzDecisionStatement` as a security and privacy authorization token as part of a SOAP message exchange in a Web Services context. This token MAY be used by a Web Service client to convey an authorization decision from a trusted 3rd party to a Web Service. A Web Service MAY use such a token to determine that the client is authorized to access information involved in the Web Service interaction.

The SAML 2.0 profile of XACML v2.0 describes the schema for an `xacml-saml:XACMLAuthzDecisionStatementType that MAY be used to convey` an authorization decision in the form of an XACML Response Context, optionally along with the corresponding XACML Request Context. The SAML 2.0 profile of XACML v2.0 describes how an instance of an `xacml-saml:XACMLAuthzDecisionStatementType` is enclosed in a SAML `Assertion`.

In a Web Services context, an instance of an `xacml-saml:XACMLAuthzDecisionStatementType` enclosed in a SAML `Assertion` MAY be used as an authorization token in the Web Services Security `wsse:Security` Header of a SOAP message. When used in this way, the `xacml-saml:XACMLAuthzDecisionStatementType` SHALL include the corresponding XACML Request Context. This allows the Web Service to determine whether the `Attributes` in the Request correspond to the access that the client requires as part of the Web Service interaction. The SAML `Assertion` containing the `xacml-saml:XACMLAuthzDecisionStatementType` instance SHOULD be signed by a Policy Decision Point trusted by the Web Service.

A Web Service MAY use this token to determine that a trusted 3rd party has evaluated an XACML Request Context that is relevant to the invocation of the Web Service, and has reported an authorization decision. The Web Service SHOULD verify that the signature on the `Assertion` is from a Policy Decision Point that the Web Service trusts. The Web Service SHOULD verify that the validity period of the SAML `Assertion` includes the time at which the Web Service interaction will access the information or resource to which the Request Context applies. The Web Service SHOULD verify that the XACML `Attributes` contained in the XACML `Request` element correctly describe the information or resource access that needs to be authorized as part of this Web Service interaction.

3 XACMLAuthzAssertion

A Web Service MAY choose to publish all or a subset of the XACML authorization, access control, and privacy policy it will apply with respect to a particular Web Service target. Likewise, a Web Service client MAY choose to make its authorization, access control, and privacy policy available for comparison with potential Web Services. This function is served by including an `XACMLAuthzAssertion`, as described in this section, in a WS-Policy instance associated with the Web Service target.

An `XACMLAuthzAssertion` represents an XACML authorization, access control, or privacy policy that applies to the target of the `wsp:Policy` instance in which it appears. The Assertion MAY be used by a Web Service to express or publish its authorization, access control, or privacy requirements or its capability of complying with requirements imposed by a client. The Assertion MAY be used by a Web Services client to express or publish its authorization, access control, or privacy requirements requirements or its capability of complying with requirements imposed by a Web Service. Two instances of such an Assertion MAY be matched to determine whether they are compatible, and, if so, which requirements and capabilities are compatible.

```
<element name="XACMLAuthzAssertion"
xsi:type="XACMLAuthzAssertionType"/>
<complexType name="XACMLAuthzAssertionType">
   <complexContent>
      <sequence>
         <element ref="Requirements" minOccurs="0" />
         <element ref="Capabilities" minOccurs="0" />
      </sequence>
   </complexContent>
</complexType>
```

The following describes the elements contained in the schema above.

`/xacml-wsp:XACMLAuthzAssertion`

This element identifies an `XACMLAuthzAssertion`. If it contains no inner elements, it represents an acknowledgment or requirement that an XACML authorization, access control, or privacy policy will be applied to the target of the `wsp:Policy` instance in which this Assertion occurs.

`/xacml-wsp:XACMLAuthzAssertion/xacml-wsp:Requirements`

This element specifies authorization, access control, or privacy requirements of the entity publishing this `XACMLAuthzAssertion`.

`/xacml-wsp:XACMLAuthzAssertion/xacml-wsp:Capabilities`

This element specifies authorization, access control, or privacy capabilities of the entity publishing this `XACMLAuthzAssertion`.

An `XACMLPolicyAssertion` SHALL NOT contain any nested `wsp:Policy` elements. There SHALL be no more than one `XACMLPolicyAssertion` included in any one `wsp:Policy` alternative.

3.1 xacml-wsp:Requirements

The `Requirements` element in an `XACMLAuthzAssertion` specifies authorization, access control, or privacy requirements on any entity that may interact with the entity publishing this `XACMLAuthzAssertion`. These requirements are with respect to the target of the `wsp:Policy` instance of which the `XACMLAuthzAssertion` is a part. The requirements in this element MAY be

only a subset of the authorization, access control, or privacy policy that will be applied at the time another entity interacts with the publishing entity; in that sense, it is advisory to the other party.

An `XACMLAuthzAssertion` that contains no `Requirements` element is equivalent to an `XACMLAuthzAssertion` that contains a `Requirements` element that is empty; in either form, it indicates that the entity publishing the `XACMLAuthzAssertion` has no authorization, access control, or privacy requirements on the other entity that it is willing to publish.

```
<element name="Requirements" xsi:type="RequirementsType"/>
<complexType name="RequirementsType">
   <complexContent>
      <choice>
         <element ref="xacml:Policy" minOccurs="0" />
         <element ref="xacml:PolicySet" minOccurs="0" />
         <element ref="xacml:Apply" minOccurs="0"
   maxOccurs="unbounded" />
      </choice>
   </complexContent>
</complexType>
```

The following describes the elements contained in the schema above.

`/xacml-wsp:XACMLAuthzAssertion/xacml-wsp:Requirements/xacml:Policy`

This element contains an XACML `Policy` instance.

`/xacml-wsp:XACMLAuthzAssertion/xacml-wsp:Requirements/xacml:PolicySet`

This element contains an XACML `PolicySet` instance.

`/xacml-wsp:XACMLAuthzAssertion/xacml-wsp:Requirements/xacml:Apply`

This element contains any number of XACML `Apply` elements, as described in Section 3.5. Each `Apply` element represents a constraint on some authorization, access control, or privacy policy `Attribute` that must be satisfied in order to meet the requirements of the entity publishing this `XACMLAuthzAssertion`. The constraints are logically `AND`ed together; that is, all constraints must be satisfied in order for the `Requirements` element to be satisfied.

A `Requirements` element using `Apply` elements is semantically equivalent to an `xacml:Policy` instance of the following form:

```
<Policy PolicyId="policyId"
RuleCombiningAlg="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides">
  <Rule>
   <Condition>
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
       ...sequence of Apply elements contained in Requirements...
     </Apply>
   </Condition>
  </Rule>
</Policy>
```

3.2 xacml-wsp:Capabilities

The `Capabilities` element in an `XACMLAuthzAssertion` specifies authorization, access control, or privacy capabilities of the entity publishing this `XACMLAuthzAssertion` with respect to the target of the `wsp:Policy` of which the `XACMLAuthzAssertion` is a part. These represent information the entity is able and willing to provide, or obligations the entity is able and willing to fulfill as part of a Web Service interaction.

```
<element name="Capabilities" xsi:type="CapabilitiesType"/>
<complexType name="CapabilitiesType">
   <complexContent>
      <choice>
         <element ref="xacml-context:Request" minOccurs="0" />
         <element ref="xacml:Apply" minOccurs="0"
   maxOccurs="unbounded" />
      </choice>
   </complexContent>
</complexType>
```

The following describes the elements contained in the schema above.

`/xacml-wsp:XACMLAuthzAssertion/xacml-wsp:Capabilities/xacml:Request`

This element contains an XACML `Request Context` instance. An instance of this element is semantically equivalent to a `Capabilities` element where, for each `Attribute` *i* in the Request Context, the `Capabilities` element contains an `Apply` element of the following form:

```
<Apply FunctionId="type-is-in">
   <AttributeValue DataType="DataType of Attribute[i]">
      value of Attribute[i]
   </AttributeValue>
   <CategoryAttributeDesignator AttributeId="AttributeId of
Attribute[i]" Datatype="DataType of Attribute[i]"/>
</Apply>
```

`/xacml-wsp:XACMLAuthzAssertion/xacml-wsp:Capabilities/xacml:Apply`

This element contains any number of XACML `Apply` element instances. Each instance represents a set of values for a particular Attribute that the entity publishing this `XACMLAuthzAssertion` is able to provide in conjunction with any of the values represented by the other `Apply` element instances.

3.3 Apply element format

Each `Apply` element in a `Capabilities` or `Requirements` represents a constraint on the permitted literal values of a policy vocabulary variable.In order to support the Assertion matching algorithm described in this profile, each such Apply element MUST conform to the following requirements.

Exactly one policy vocabulary variable MUST be referenced in each `Apply` element. That is, there MUST be exactly one `AttributeDesignator` or `AttributeSelector` instance in the Apply element.

`Apply` element instances MUST NOT be nested except for the purpose of converting a bag to a single value or for purposes of limiting the scope of a set of constraints using the `limit-scope` function.

The `FunctionId` attribute in an `Apply` element MUST be one of those listed in Appendix A. This profile does not define the intersection of two `Apply` elements using functions that are not in Appendix A.

3.4 Obligations

Obligations in an `XACMLAuthzAssertion` SHALL be represented by `Attributes`. Imposing an obligation SHALL be expressed by including an `Apply` element equating the obligation `Attribute` with the value associated with the obligation in the `XACMLAuthzAssertion/Requirements` element. An entity able and willing to fulfill an obligation SHALL include an instance of the obligation Attribute having the appropriate value in the `XACMLAuthzAssertion/Capabilities/Request` element, or SHALL include an `Apply` element equating the obligation `Attribute` to the appropriate value in its `XACMLAuthzAssertion/Capabilities` element.

> **Non-normative example:** If a Web service imposes an obligation on the client not to disclose information obtained through the Web services interaction to any 3rd parties, this can be expressed as follows. Define an XACML `Attribute` with `AttributeId` "*namespace*`:DisclosureOfInformation`" and a value of "`None to 3rd parties`". The Web service can include an `Apply` element equating the `Attribute` with `AttributeId` "*namespace*`:DisclosureOfInformation`" to the value "`None to 3rd parties`" in its `XACMLAuthzAssertion/Requirements`. A client willing and able to accept this obligation can either include an instance of the `Attribute` with `AttributeId` "*namespace*`:DisclosureOfInformation`" having the value "`None to 3rd parties`" in its `XACMLAuthzAssertion/Capabilities/Request` element or can include an `Apply` element equating the `Attribute` with `AttributeId` "*namespace*`:DisclosureOfInformation`" to the value "`None to 3rd parties`" in its `XACMLAuthzAssertion/Capabilities/` element.

3.5 Assertion Matching

An `XACMLAuthzAssertion` in one `wsp:Policy` instance matches an `XACMLAuthzAssertion` in another `wsp:Policy` instance if all of the `Requirements` of each entity are satisfied by the `Capabilities` of the other entity. An empty `XACMLAuthzAssertion` always matches another `XACMLAuthzAssertion`: it represents an acknowledgment or requirement that an XACML authorization, access control, and privacy policy will be applied to the target of the `wsp:Policy` instance in which this Assertion occurs when a Web Service interaction occurs.

The following match semantics are defined between the `Requirements` of one entity and the `Capabilities` of the other entity.

Requirements Format	Capabilities Format	Match algorithm
xacml:Policy or xacml:PolicySet	xacml-context:Request	"True" if Request evaluated against the Policy or PolicySet according to the corresponding XACML Standard returns "Permit"; otherwise "False".
xacml:Policy or xacml:PolicySet	xacml:Apply	Not defined
xacml:Apply	xacml-context:Request	Consider the Requirements to be an XACML Policy as described in Section 2.1. "True" if Request evaluated against the Policy or PolicySet according to the corresponding XACML Standard returns "Permit"; otherwise "False".
xacml:Apply	xacml:Apply	"True" if, for each Apply element in the Requirements, there is at least one Apply element in the Capabilities for which the intersection defined in Section 3.6 is non-empty; otherwise "False".

3.6 Intersection of two Apply elements

Two Apply elements that reference the same vocabulary variable are **coincident.**

The intersection of two **coincident** Apply elements is either the empty set, the two Apply elements or a single Apply element. The result of taking the intersection of two **coincident** Apply elements and the FunctionId and <AttributeValue> of any single resulting Apply element are specified in Table 1 of Appendix A. If the intersection of the two Apply elements is the empty set, then the Apply elements are incompatible and do not match.

3.7 Intersection of two XACMLAuthzAssertion instances

The intersection of two XACMLAuthzAssertion instances represents an agreement concerning the policy to be applied between the two parties publishing the XACMLAuthzAssertion. The intersection is defined only for the case where, for both Assertions, the Requirements and Capabilities contain Apply elements, or the Capabilities contain a Request element that is considered as a sequence of Apply elements as described in Section 3.2.

The intersection of two XACMLAuthzAssertion instances is the empty set, represented by an error indicating "no intersection", unless the two Assertions match according to Section 3.5. If the two Assertions match according to Section 3.5, then the result of the intersection is two new XACMLAuthzAssertions, one for each party. The Capabilities section for a given party contains the intersection of the Requirements of the other party with the Capabilities of the given party as defined in Section 3.5. The Requirements section for a given party contains the intersection of the Requirements of the given party with the Capabilities of the other party as defined in Section 3.5.

Note that an intersection MAY support multiple policy alternatives. A Web Service MAY propose a specific alternative to the other party by conveying a new XACMLAuthzAssertion in which each Apply

element in the Requirementsand Capabilities sections is replaced by an equality function between the AttributeDesignator or AttributeSelector from the original Apply element and one literal value that satisfies the original Apply element.

Non-normative example: the following Apply element:

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-
greater-than">
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-
one-and-only">
    <SubjectAttributeDesignator AttributeId="namespace:A"
DataType="http://www.w3.org/2001/XMLSchema#integer"/>
  </Apply>
  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"/>5</AttributeVa
lue>
</Apply>
```

may be replaced by the following Apply element:

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-
equal">
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-
one-and-only">
    <SubjectAttributeDesignator AttributeId="namespace:A"
DataType="http://www.w3.org/2001/XMLSchema#integer"/>
  </Apply>
  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"/>8</AttributeVa
lue>
</Apply>
```

The value “8” used in this example could have been any integer value greater than 5; that is, any value that would have satisfied the original Apply element.

This Profile does not specify a protocol by which such a specific policy alternative is conveyed to the other party.

3.8 Evaluation of an XACMLAuthzAssertion instance

An `XACMLAuthzAssertion` SHALL evaluate to “True”, indicating that the Assertion is satisfied, if and only if all Requirements in the Assertion evaluate to “True” against the information related to an actual instance of the target of the WS-Policy instance with which the Assertion is associated. Otherwise, an `XACMLAuthzAssertion` SHALL evaluate to “False”, indicating that the Assertion is not satisfied.

4 Conveying XACML Attributes in a SOAP Message

At the time a Web Service is invoked, the Web Service MAY need to determine whether the client is authorized to invoke the Service or to access resources that are involved in the Service invocation. A Web Service MAY use XACML to make such an authorization decision.

When a Web Service evaluates an XACML authorization, access control, or privacy policy related to a SOAP message, it MAY obtain the `Attributes` required for the evaluation from various sources, including databases, registries, trusted Attribute Authorities, and so on. This work is done in the application-dependent XACML Context Handler that provides `Attributes` to the PDP on request. A Web Services client or intermediary MAY include XACML `Attributes` in a `wsse:Security` SOAP Header for use by this Context Handler. This section of this profile describes two ways in which such `Attributes` MAY be provided.

4.1 xacml-samlp:XACMLAuthzDecisionQuery

The first way in which `Attributes` may be provided to a Web Service is by including an instance of an `xacml-samlp:XACMLAuthzDecisionQuery` in the `wsse:Security` Header of a SOAP message. This query contains an XACML Request Context that SHOULD contain XACML `Attributes` related to access that the client will need in order to interact successfully with the Web Service. The `xacml-samlp:XACMLAuthzDecisionQuery` SHOULD be signed by an entity that the Web Service trusts to authenticate the enclosed XACML `Attributes`.

The Web Service MAY use the Attributes in such an `xacml-samlp:XACMLAuthzDecisionQuery` as part of evaluating XACML policies related to the Web Service interaction. The Web Service SHOULD verify that the query is signed by an entity that the Web Service trusts to authenticate the enclosed XACML Attributes. It SHOULD verify that the IssueInstant of the `xacml-samlp:XACMLAuthzDecisionQuery` is close enough the the current time to meet the validity requirements of the Web Service.

The `xacml-samlp:XACMLAuthzDecisionQuery` is fully described in the SAML 2.0 profile of XACML v2.0 [].

4.2 saml:Attribute

A second way in which `Attributes` may be provided to a Web Service is in the form of `saml:Attribute` elements in a `saml:AttributeStatement` that is part of a `saml:Assertion` in the `wsse:Security` Header of a SOAP message. These `Attributes` MAY be converted to XACML `Attributes` as described in the SAML 2.0 profile of XACML v2.0 [] by an XACML Context Handler for use by a PDP associated with the Web Service in evaluating XACML policies related to the Web Service interaction.

5 New XACML Functions

For use in specifying XACMLAuthzAssertions, the following new XACML functions are defined.

5.1 [namespace:]must-be-present

This function SHALL take one <AttributeDescriptor> or an <AttributeValue> with data type "&xml;string" as its argument and SHALL return a Boolean result. The <AttributeValue> string SHALL be interpreted as an XPath expression. The result SHALL be "true" if the bag containing the values of the specified <AttributeDescriptor> or the nodeset representing the XPath expression contains at least one value; the result SHALL be "false" otherwise.

This function is equivalent to applying the "bag-size" function to the <AttributeDescriptor> and comparing the result using the "&function;integer-greater-than-or-equal" function to the integer value "1".

5.2 [namespace:]must-not-be-present

This function SHALL take one <AttributeDescriptor> or an <AttributeValue> with data type "&xml;string" as its argument and SHALL return a Boolean result. The <AttributeValue> string SHALL be interpreted as an XPath expression. The result SHALL be "true" if the bag containing the values of the specified <AttributeDescriptor> or the nodeset representing the XPath expression contains no values (i.e. is an empty bag or set); the result SHALL be "false" otherwise.

This function is equivalent to applying the "bag-size" function to the <AttributeDescriptor> and comparing the result using the "function:integer-equal" function to the integer value "0".

5.3 [namespace:]limit-scope

This function is used to group **constraints** that must all be satisfied by a single element in an XML schema instance.

This function SHALL take two or more parameters, where the first parameter is and <AttributeValue> with data type "&datatype;string", and the remaining parameters are **policy constraints**. It SHALL return a Boolean result. The <AttributeValue> used as the first parameters SHALL be interpreted as an XPath expression. The result SHALL be "true" if the **constraints** are all "true" when applied to at least one single **node** in the nodeset selected by the <AttributeValue>'s XPath expression. That is, all **constraints** must be satisfied by the same node, although there may be multiple such **nodes** where all **constraints** are satisfied.

Instances of the "limit-scope" function SHALL be treated for purposes of **policy** intersection as if all the **constraints** were individual **constraints** specified separately without the "limit-scope" function. This function may be nested arbitrarily deeply, but all **constraints** are treated as if they were specified at the top-most **policy set constraint** level.

Non-normative example:

```
<Apply FunctionId="[namespace:]limit-scope">
    <AttributeValue DataType="&xml;string">/security/key-info</AttributeValue>
    <Apply FunctionId="&function;integer-equal">
        <AttributeSelector ElementId="/key-length" DataType="&xml;integer">
        <AttributeValue DataType="&xml;integer">96</AttributeValue>
```

```
    </Apply>
    <Apply FunctionId="&function;string-equal">
        <AttributeSelector ElementId="/algorithm" DataType="&xml;string">
        <AttributeValue DataType="&xml;string">DES-CBC</AttributeValue>
    </Apply>
</Apply>
```

This function returns "`true`" only if there is at least one "`/security/key-info`" element that has a "`key-length`" child with value `96` and an "`algorithm`" child with value "`DES-CBC`". [XPath syntax might treat parent "limit-scope" as new root, or we might require evaluator to follow only paths that lead into same element.]

This function specifies two **policy constraints**. For intersection purposes, these **constraints** are considered as if they were enclosed with an AND operator at the **policy sets** level.

Appendix A. Supported constraint functions and their intersections (normative)

The following table lists the XACML functions that are supported for use in **policy constraints**, and describes how to compute the intersection of any two **constraints.**

If two **constraints** constrain the same **vocabulary item**, then they are said to be **coincident.** If two **constraints** in XACMLAuthzAssertions that are being matched are not **coincident,** then their intersection is the two original **constraints.**

If two **coincident constraints** are not compatible according to the "Compatibility test" column of Table 1 below, then the two **constraints** are incompatible, and the XACMLAuthzAssertions containing them do not match.

If two **coincident constraints** are compatible according to the "Compatibility test" column, then their intersection is the **constraint** specified in the "Replacement constraint" column of Table 1.

Table 1 is to be interpreted according to the following key.

Columns one, two and four contain shorthand versions of an XACML `<Apply>` element. The portion before the open parenthesis (e.g. "type-equal" in the first row) represents the `<Apply>` element's `FunctionId` attribute value. The "type-" portion represents any of the type-specific parts of the standard XACML function identifiers.

Alphabetic symbols (e.g. "a" in the first row) represent XACML `<AttributeDesignator>`, `<AttributeSelector>` or `<AttributeValue>` elements. Where a **constraint** `FunctionId` takes a single value rather than a bag for an argument where an `<AttributeDesignator>` or `<AttributeSelector>` is used, the `<AttributeDesignator>` or `<AttributeSelector>` SHALL be enclosed in an inner `<Apply>` element having as its `FunctionId` the appropriate "type`-one-and-only`" function.

Where "Keep both constraints" appears in the "Replacement constraint" column, there is no single replacement `<Apply>` element: the **predicates** are compatible, but not combinable. In these cases, the original `<Apply>` elements MUST NOT be modified by this step in the procedure.

any-constraint(a) is "true" if there is any **constraint** in the **policy set** that applies to **vocabulary item** a.
no-constant(a) is "true" if there is no **constraint** in the **policy set** that applies to **vocabulary item** a.

∩ means set intersection.

⊆ means "is a subset of".

	First constraint	Second constraint	Compatibility test	Replacement constraint
1	type-equal(a,b)	type-equal(a,c)	b == c	type-equal(a,b)
2	type-equal(a,b)	type-greater-than(a,c)	b > c	type-equal(a,b)
3	type-equal(a,b)	type-greater-than-or-equal(a,c)	b ≥ c	type-equal(a,b)
4	type-equal(a,b)	type-less-than(a,c)	b < c	type-equal(a,b)

5	type-equal(a,b)	type-less-than-or-equal(a,c)	b ≤ c	type-equal(a,b)	
6	type-greater-than(a,b)	type-greater-than(a,c)		type-greater-than(a,max(b,c))	
7	type-greater-than(a,b)	type-greater-than-or-equal(a,c)		Where b ≥ c	type-greater-than(a,b)
8				Where b < c	type-greater-than-or-equal(a,c)
9	type-greater-than-or-equal(a,b)	type-greater-than-or-equal(a,c)		type-greater-than-or-equal(a,max(b,c))	
10	type-less-than(a,b)	type-less-than(a,c)		type-less-than(a,min(b,c))	
11	type-less-than(a,b)	type-less-than-or-equal(a,c)		Where b > c	type-less-than-or-equal(a,c)
12				Where b ≤ c	type-less-than(a,b)
13	type-less-than-or-equal(a,b)	type-less-than-or-equal(a,c)		type-less-than-or-equal(a,min(b,c))	
14	type-greater-than(a,b)	type-less-than(a,c)	b < c	Keep both constraints	
15	type-greater-than(a,b)	type-less-than-or-equal(a,c)	b < c	Keep both constraints	
16	type-greater-than-or-equal(a,b)	type-less-than(a,c)	b < c	Keep both constraints	
17	type-greater-than-or-equal(a,b)	type--less-than-or-equal(a,c)	b < c	Keep both constraints	
18	set-equals(a,b)	set-equals(a,c)	b == c	set-equals(a,b)	
19	set-equals(a,b)	subset(a,c)	b ⊆ c	set-equals(a,b)	
20	subset(a,b)	subset(a,c)	∩ (b,c) ≠ 0	subset (a,∩ (b,c))	
24	time-in-range(a,b,c)	time-equal(a,d)	b ≤ d ≤ c	time-equal(a, d)	
25	time-in-range(a,b,c)	time-greater-than-or-equal(a,d)	b ≤ d ≤ c	time-in-range(a, d, c)	
26	time-in-range(a,b,c)	time-less-than-or-equal(a,d)	b ≤ d ≤ c	time-in-range(a, b, d)	
27	must-be-present(a)	Any constraint other than must-not-be-present(a)		The second constraint	
28	must-not-be-present(a)	must-not-be-present(a)		must-not-be-present(a)	
29	any-constraint(a)	no constraint(a)		any-constraint(a)	

Table 1 - Intersection of coincident constraints

Appendix B. Acknowledgments

The semantics for intersection of XACMLAuthzAssertions is taken from the *XACML profile for web services*, edited by Tim Moses and largely developed by him. We acknowledge this contribution with particular gratitude.

The new functions, the use of sequences of Apply elements, and new intersection algorithms are taken from the *XACML-Based Web Services Policy Constraint Language (WS-PolicyConstraints)* authored by Sun Microsystems. This material is being contributed by Sun to OASIS as part of this specification.

The following individuals have participated in the creation of this specification and are gratefully acknowledged

Participants: TBD (the usual list of XACML TC voting members over time)

- [Participant name, affiliation | Individual member]
- [Participant name, affiliation | Individual member]
- [Participant name, affiliation | Individual member]

Appendix C. Revision History

[optional; should not be included in OASIS standards]

Rev	Date	By Whom	What
01	9 Oct 2006	Anne Anderson	Initial version. Appendix A derived from XACML profile for Web-services Working Draft.

Appendix D. Non-Normative Text