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

Working Draft 8, 12 December 2006

Document identifier:
xacml-3.0-profile-webservices-spec-v1.0-wd-8

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,P3P

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
- W3C Platform for Privacy Preferences 1.0 (P3P1.0)

Abstract:
This document specifies ways to use XACML in the context of Web Services for authorization, access control, and privacy policies. It specifies four 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) A policy Assertion type based on XACML elements for use with WS-Policy or other schemas and protocols; 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. This Profile specifies standard formats, matching semantics, and usage guidelines for two Assertions derived from this type: one for authorization policies and the other for privacy policies. 3) Some ways in which Attributes for a client MAY be passed to a Web Service as part of a SOAP message in such a way that they can be authenticated as having been issued by a trusted authority. These Attributes may be used by the Web Service in evaluating the internal XACML policies of a service or enterprise that are relevant to a given Web Services access. 4) How to express P3P policy preferences and match them using the new Assertion based on XACML.

Status:

This document was last revised or approved by the XACML TC 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[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.

There are four types of information specified in this Profile, designed to satisfy four different, but sometimes related Web Services requirements.

1. **XACML Authorization Token**: Web Services Security specifies formats for various security tokens, but does not address authorization tokens related to policies expressed using XACML. This Profile defines a format for XACML authorization decisions as authorization tokens or credentials in a Web Services environment; the token MAY be used to prove that a given access request satisfies an authorization policy without the requiring the consumer of the proof to re-evaluate the policy.

2. **XACML Assertion**: 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. This Profile may also be used to express Assertions in contexts other than WS-Policy. Two specific assertions derived from the XACML Assertion type are specified: an **XACMLAuthzAssertion** for use with authorization and access control policies, and an **XACMLPrivacyAssertion** for use with privacy policies.

3. **XACML P3P Policies:** The W3C Platform for Privacy Preferences (P3P) specifies a standard format for describing user-oriented privacy policies, but there has not been a satisfactory mechanism for matching a user's privacy preferences against those of a given service. This Profile defines how the **XACMLPrivacyAssertion** defined here may be used to specify both user and service privacy preferences and offerings with respect to P3P privacy policies, and to match them in such a way that the user's preferences can be satisfied if possible.

4. **XACML Attributes in SOAP Message Headers**: A client may need to convey as part of a Web Services message to a service the values of Attributes to be used in evaluating an access request related to the Web Services interaction. Such Attributes must be provably issued by an authority trusted by the PDP since otherwise rogue clients could submit bogus Attributes of their choosing. This Profile explains how to use SAML Assertions to convey such Attribute values in a SOAP message header.

1.1 Use Cases

The following are some sample use cases for the formats and behaviors specified in this Profile. In each use case, not all requirements need to be satisfied by this Profile – there may be other mechanisms available, but places where this Profile MAY be used are mentioned.

– **Services unable to evaluate authorization policies directly:** 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 service 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. The **XACML Authorization Token** MAY be used as the token in such a use case. An **XACMLAuthzAssertion** MAY be used by the service to publish its requirement that such a token be supplied, and MAY be used by a client to record that the client is capable of obtaining such an authorization token. Where both parties use such an Assertion, the semantics defined for matching requirements and capabilities in **XACML Assertions** MAY be used by a client, the service, or a proxy to determine whether the client will be able to satisfy the requirement.

- **Client sends authorization Attributes to a service:** 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. **XACML Attributes in SOAP Message Headers** MAY be used to convey the Attributes in such a use case.

- **Service requires a particular authorization Attribute for access:** 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. The **XACMLAuthzAssertion** MAY be used by the service to express its requirement, and MAY be used by the client to record its consortium membership. Where both parties use such an Assertion, the semantics defined for matching requirements and capabilities in **XACML Assertions** MAY be used by a client or a proxy to determine whether the client's recorded capabilities will be able to satisfy the requirement. If so, then **XACML Attributes in SOAP Message Headers** MAY be used by the client to convey the necessary Attribute to the service.

- **Service requires that Attributes be signed by a particular trusted 3rd party:** 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 a client may arrange to obtain `role` Attributes signed by that 3rd party, and then make them available. Again, the **XACMLAuthzAssertion** MAY be used by the service to express its requirement, and MAY be used by the client to record the trusted 3rd parties from which it is able to obtain `role` Attributes. Where both parties use such an Assertion, the semantics defined for matching requirements and capabilities in **XACML Assertions** MAY be used by a client or a proxy to determine whether the client's recorded capabilities are able to satisfy the requirement. If so, then **XACML Attributes in SOAP Message Headers** MAY be used by the client to convey the correctly signed `role` Attributes to the service.

- **Client needs to activate a particular role in order to interact with a service:** 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. Again, the **XACMLAuthzAssertion** MAY be used by the service to express its `role` requirement, and MAY be used by the client to record its capability of activating various `roles`. Where both parties use such an Assertion, the semantics defined for matching requirements and capabilities in **XACML Assertions** MAY be used by a client or a proxy to determine whether the client's recorded capabilities are able to satisfy the requirement and if so, to trigger role activation.

- **Client wants to know if its access request is likely to accepted by the service:** 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 wants to 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. An **XACMLAuthzAssertion** MAY be used by the client to express the Attributes it is capable of supplying in an access request. The service MAY publish its required authorization policy also using an **XACMLAuthzAssertion**. Where both parties use such Assertions, the semantics defined for matching requirements and capabilities in **XACML Assertions** MAY be used by the client or a proxy to determine whether its request is likely to satisfy the policy requirements of the service.

- **Service requires client to fulfill an obligation in order to obtain access:** 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. An **XACMLAuthzAssertion** or **XACMLPrivacyAssertion MAY** be used by the service to express its required obligations, and MAY be used by a client to record its capability of fulfilling various obligations. Where both parties use such an Assertion, the semantics defined for matching requirements and capabilities in **XACML Assertions** MAY be used by a client or a proxy to determine whether the client is able and willing to satisfy the required obligation.

- **Client requires service to fulfill an obligation regarding client's provided personal information:** 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. An **XACMLPrivacyAssertion** MAY be used by the client to express its required obligations, and MAY be used by the service to to record its capability of fulfilling various obligations. Where both parties use such an Assertion, the semantics defined for matching requirements and capabilities in **XACMLPrivacyAssertions** MAY be used by the service or a proxy to determine whether the service is able and willing to satisfy the required obligation.

- **Client and service want to know if they support mutually compatible P3P policies:** A service supports various **P3P privacy policies** associated with various resources. A prospective client for the service has various privacy preferences. The client and service need to determine whether the service is able to satisfy the preferences of the user with respect to a particular interaction, and, if so, which specific preferences should be supported. An **XACMLPrivacyAssertion** MAY be used by the client to record its P3P privacy policy requirements (preferences), and MAY be used by the service to record the P3P privacy policy preferences it is capable of supporting with respect to this type of interaction. Where both parties use such an Assertion, the semantics defined for matching requirements and capabilities in **XACML Assertions** MAY be used by the service or a proxy to determine whether the service is able and willing to conform to the required P3P policy requirements.

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, although this Profile does not specify a protocol for communicating such a mutually acceptable policy alternative to another party.

In many cases, a publicly available Web Service will want potential clients know exactly what is required in order to be authorized to use the service. In many other cases, a Web Service will not want to advertise the full authorization, access control, and privacy policies it will be applying to accesses 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. This subset can serve as a first-level filter by the service 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.

1.2 XACML Compatibility Requirements

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

1.3 Terminology

The namespace for the schema [SCHEMA] associated with this specification is

```
urn:oasis:names:tc:xacml:3.0:profile:webservices:v1.0:schema
```

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 .

capabilities: information and behaviors that the entity publishing the XACMLAssertion is willing and able to provide to another party with respect to the policy target and the policy domain of the XACMLAssertion, in conjunction with any requirements specified in the same XACMLAssertion. Capabilities are specified in the `Capabilities` element of the XACMLAssertion.

client: a Web Service client or consumer.

coincident: If two policy constraints constrain the same policy variable, then they are said to be **coincident**.

policy constraint: a way of describing and restricting the acceptable values of a policy variable. In this Profile, a policy constraint is implemented as an XACML `Apply` element, which is a Boolean function that compares the value of the policy variable, implemented as an XACML `AttributeDesignator` or `AttributeSelector`, to one or more acceptable values or to values that bracket a range of acceptable values. The function returns “True” if the policy variable has an acceptable value. Only a subset of the XACML standard functions may be used in policy constraints, as defined in Appendix A.

policy domain: a specific sphere of concern with respect to Web Services interactions. Examples of policy domains are authentication, reliable transactions, atomic transactions, trust establishment, and secure transactions. The policy domains addressed in this Profile are authorization (including access control and entitlements), and privacy (including confidentiality obligations).

policy target: the scope within which the policy applies. The scope may be an interaction, a resource access, a method invocation, etc. This Profile does not define how an XACMLAssertion is associated with a particular policy target. Typically, an assertion is used within a policy framework (such as WS-Policy) or a protocol that defines methods for associating the assertions used with policy targets.

policy variable: a policy variable is an element of the vocabulary used to express a policy. In XACML, a policy variable is either an `AttributeDesignator` specifying a named, typed XACML Attribute, or an `AttributeSelector` specifying an XPath expression selecting typed nodes in an XML document that contain information related to the policy.

policy vocabulary: a policy vocabulary is a set of policy variables used to describe information and behaviors relevant to a Web Service with respect to some policy domain. The policy vocabulary of an XACMLAssertion's `Requirements` element is the union of the sets of policy variables defined in the `Vocabulary` elements included in the `Requirements`. The policy vocabulary of an XACMLAssertion's `Capabilities` element is the union of the sets of policy variables defined in the `Vocabulary` elements included in the `Capabilities`.

requirements: with respect to an XACMLAssertion, “requirements” are information and behaviors that the entity publishing the XACMLAssertion has on another party with respect to the policy target and the policy domain of the XACMLAssertion. Requirements are specified in the `Requirements` element of the XACMLAssertion.

service: a Web Service provider.

vocabulary instance: a vocabulary instance is a specific set of values for a particular policy vocabulary.

XACMLAssertion: any element derived from the abstract `XACMLAssertionAbstractType`. The two XACMLAssertions defined in this Profile are the `XACMLAuthzAssertion` and the `XACMLPrivacyAssertion`.

1.4 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.

[P3P] W3C, *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*, W3C Recommendation 16 April 2002. http://www.w3.org/TR/2002/REC-P3P-20020416/.

[SCHEMA] OASIS, *Schema for the Web Service Profile of XACML (WS-XACML) Version 1.0, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml*

[WS-POLICY] W3C, *Web Services Policy 1.2 - Framework (WS-Policy)*, W3C Member Submission 25 April 2006. http://www.w3.org/Submission/WS-Policy/. [TO BE UPDATED TO STANDARD]

[WSS] OASIS, *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)*, OASIS Standard 1 February 2006. http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

[XACML] OASIS, *eXtensible Access Control Markup Language (XACML) Version 2.0*, OASIS Standard 1 February 2005. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#XACML20.

[XACML10] OASIS, *eXtensible Access Control Markup Language (XACML) Version 1.0,* OASIS Standard 6 February 2003. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#XACML10.

[XACML-SAML] OASIS, *SAML 2.0 profile of XACML v2.0*, OASIS Standard 1 February 2005. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

1.5 Non-normative References

None.

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 client to convey an authorization decision from a trusted 3rd party to a service. A service MAY use such a token to determine that the client is authorized to access information involved in the Web Services interaction.

The *SAML 2.0 profile of XACML v2.0* [XACML-SAML] 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 [WSS] `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 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 service.

A 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 service, and has reported an authorization decision. The service SHOULD verify that the signature on the `Assertion` is from a Policy Decision Point that the service trusts. The 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 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 XACMLAssertion Abstract Type

An XACMLAssertion is a description of an entity's Web Service's policy with respect to some policy domain, such as authorization or privacy; the name of the XACMLAssertion element indicates the domain to which it applies. The conceptual model for an XACMLAssertion is based on each policy domain having a set of variables whose values describe information and behaviors that are associated with a Web Services interaction with respect to that domain. Such a set of variables is called a "policy vocabulary", and a particular set of values for such a vocabulary is called a "vocabulary instance". An XACMLAssertion states an entity's policy with respect to that domain by describing instances of a policy vocabulary that are acceptable. Various XACML elements - `Policy`, `PolicySet`, `Request`, `Apply` – SHALL be used to describe such acceptable instances. With the exception of the XACML `Request`, these XACML elements are functions that accept a vocabulary instance or individual vocabulary variable value as input and return "`True`" or "`Permit`" if the instance or value is acceptable; "`Permit`" is interpreted as equivalent to "`True`" in this context, while any XACML combining algorithm result other than "`Permit`" is interpreted as "`False`". In an XACMLAssertion, an XACML `Request` SHALL be understood as a function that accepts any subset of the variable values contained in the `Request`. The policy variables in an XACMLAssertion SHALL be XACML Attributes (named, typed values) or typed nodes in an XML document that are selected using an XPath expression. Any instance of an XACMLAssertion is a complete description of one or more acceptable sets of values with respect to the related policy domain, but it may require multiple instances of that XACMLAssertion type to completely describe all acceptable sets of values for that domain. Thus it is sufficient, but not necessary, for a given set of values to satisfy at least one instance of the related named XACMLAssertion type.

This Profile defines an abstract `XACMLAssertionAbstractType` from which various specific named types of XACMLAssertion SHALL be derived. All derived XACMLAssertions share common semantics for evaluation and for matching against each other, although their policy vocabularies will usually differ. This Profile defines two such derived XACMLAssertions; an `XACMLAuthzAssertion` for the authorization domain and an `XACMLPrivacyAssertion` for the privacy domain. This Section of the Profile describes the common semantics for all XACMLAssertions, while Sections 4 and 5 describe the semantics particular to the two derived XACMLAssertion types.

An XACMLAssertion MAY be used to express the policy requirements, the policy capabilities, or both for an entity with respect to some policy target associated with the Assertion. Policy **requirements** are descriptions of information or behaviors that the entity requires from another party with respect to the target of the XACMLAssertion. Policy **capabilities** are descriptions of information or behaviors that the entity is willing and able to provide to another party with respect to the target of the XACMLAssertion, in connection with the stated requirements. A single XACMLAssertion describes one or more combinations of requirements and capabilities that are acceptable to the issuer of the XACMLAssertion: the stated capabilities are offered in connection with the stated requirements in each combination. The information and behaviors specified in the requirements and capabilities in an XACMLAssertion are described in terms of specified policy vocabularies.

An XACMLAssertion MAY be used by a service to express or publish its own requirements or its capabilities for complying with requirements imposed by a client. Likewise, an XACMLAssertion MAY be used by a client to express or publish its own requirements or its capabilities for complying with requirements imposed by a service. The requirements specified in an XACMLAssertion constrain the acceptable values of variables a policy vocabulary. The capabilities specified in an XACMLAssertion constrain the acceptable values of variables in the same or a different policy vocabulary.

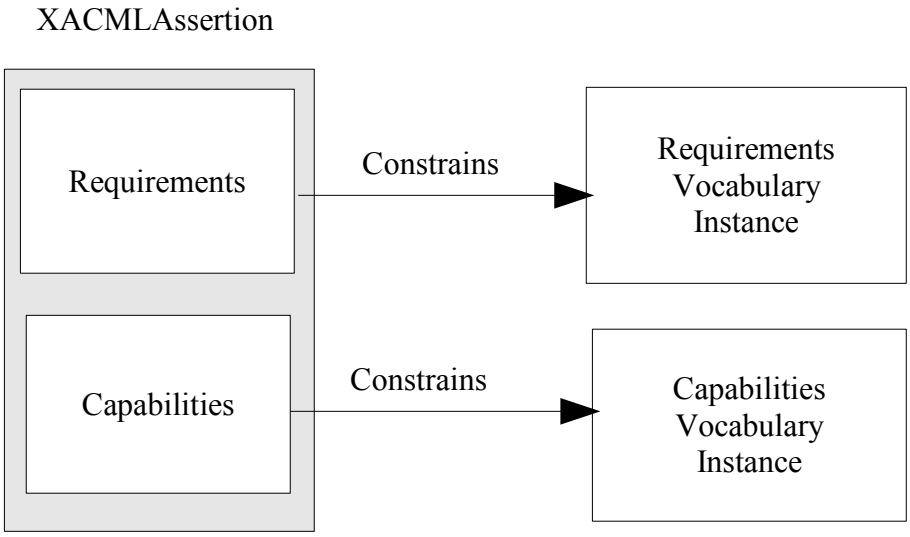

Figure 1: An XACMLAssertion constrains vocabulary instances

Non-normative XACMLAssertion example

Service A has an `XACMLPrivacyAssertion` associated with a Web Services interface allowing a client to request a price list for parts sold by Service A's owner. The **requirements** in the `XACMLPrivacyAssertion` constrain the values of a vocabulary that describes a client's responsibilities for preserving the confidentiality of the price list that Service A will be providing to the client through this interface. This vocabulary contains a service application-defined XACML Attribute named "`urn:serviceA:attributes:maximum-retention-days`", and another XACML Attribute specified as the P3P schema element: "`//P3P10/POLICIES/POLICY/STATEMENT/RECIPIENT/ours`". The first Attribute is defined by the Service A application to be an integer specifying the maximum number of days the client may keep the price list before destroying all copies of it; in this `XACMLPrivacyAssertion` it has the value "`30`". The second Attribute is defined by P3P to indicate that the client is not permitted to share the information received (the price list) with a 3rd party. The **capabilities** in Service A's `XACMLPrivacyAssertion` constrain instances of a P3P schema element to values indicating Service A will collect only that personal information necessary for the current transaction ("`//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/current`") and that this personal information will not be released to 3rd parties ("`//P3P10/POLICIES/POLICY/STATEMENT/RECIPIENT/ours`"). Both the requirements and the capabilities constrain the same P3P schema element, but in the requirements the element represents a requirement on the client's behavior, while in the capabilities it represents a service behavior that the service promises as part of this Web Services interaction.

XACMLAssertion scope

The scope of any specific XACMLAssertion is defined along three dimensions.

1) The XACMLAssertion is defined with respect to policy vocabularies that are specified in the requirements and capabilities sections of the XACMLAssertion. The XACMLAssertion SHALL apply only to those specified vocabularies.

2) The XACMLAssertion is defined with respect to one or more specific policy targets. An XACMLAssertion, when included in a WS-Policy instance, represents requirements and capabilities that apply to the targets of the `wsp:Policy` instance in which it appears. Each other context in which an XACMLAssertion is used SHALL specify how the policy targets associated with the XACMLAssertion are determined.

3) The capabilities expressed in an XACMLAssertion are defined in conjunction with the requirements in the same XACMLAssertion.

Assertion evaluation

The requirements specified in an XACMLAssertion MAY be evaluated against instances of the policy vocabulary associated with those requirements: the requirements will evaluate to “true” or “Permit” if the vocabulary instance is compatible with the XACMLAssertion. Assertion evaluation is described in more detail in Section 3.8.

Assertion matching

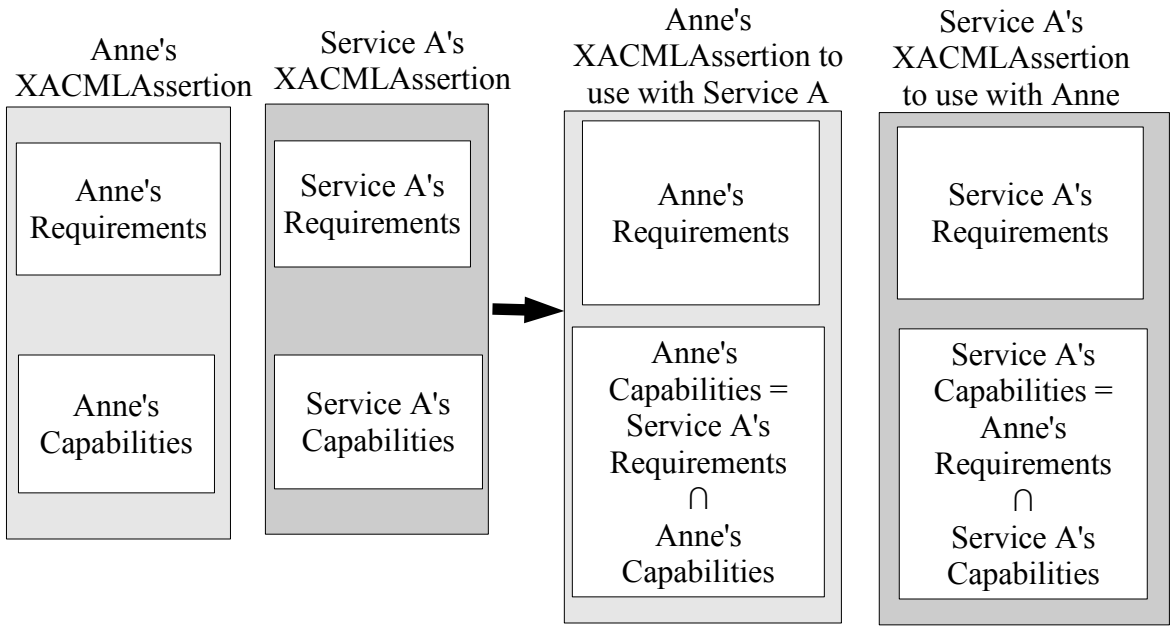

Figure 2: Matching two XACMLAssertions

Two XACMLAssertions with the same name, such as one from a service's policy and one from a client's policy, MAY be matched to determine whether they are compatible, and, if so, exactly which requirements and capabilities are mutually compatible. Matching is done by computing the intersection of the requirements in each XACMLAssertion with the capabilities in the other XACMLAssertion. For each of the original XACMLAssertions, the result of the intersection is a new XACMLAssertion containing in its requirements the intersection of the original requirements with the original capabilities of the other XACMLAssertion, and containing in its capabilities the intersection of the original capabilities with the original requirements of the other XACMLAssertion. The resulting requirements describe one or more instances of the requirements vocabulary that the issuer of the XACMLAssertion can expect from the other entity in association with the policy target; the resulting capabilities describe one or more instances of the capabilities vocabulary that the issuer of the XACMLAssertion can provide to the other entity in association with the policy target. If either intersection is empty, then the two XACMLAssertions do not match and are not compatible: there is no instance of the requirements and capabilities policy

vocabularies that can satisfy the requirements and capabilities of both entities. Any one of those vocabulary instances may be selected for use with the target of the two original policies. How a particular instance is selected and which entity does the selecting are not addressed by this Profile. Assertion matching is described in more detail in Section 3.8.

3.1 ws-xacml:XACMLAssertionAbstractType

```
<complexType name="XACMLAssertionAbstractType" abstract="true">
    <sequence>
        <element ref="ws-xacml:Requirements" minOccurs="0"
   maxOccurs="unbounded"/>
        <element ref="ws-xacml:Capabilities" minOccurs="0"
   maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

The following describes the types and elements contained in the schema above.

`/ws-xacml:XACMLAssertionAbstractType`

This type SHALL be used to derive named XACMLAssertions. If an XACMLAssertion contains no inner elements, it SHALL represents an acknowledgment or requirement that an XACMLAssertion of the specified name will be applied in connection with the policy target associated with the XACMLAssertion; in this case, the XACMLAssertion is advisory.

`/ws-xacml:`*XACMLAssertion*`/ws-xacml:Requirements`

This element specifies requirements of the entity publishing this XACMLAssertion with respect to the policy target. There may be any number of `Requirements` elements in a single XACMLAssertion.

`/ws-xacml:`*XACMLAssertion*`/ws-xacml:Capabilities`

This element specifies capabilities of the entity publishing this XACMLAssertion with respect to the policy target and in connection with the requirements specified in the `Requirements` element. There may be any number of `Capabilities` elements in a single XACMLAssertion.

When used in a WS-Policy instance, an XACMLAssertion SHALL NOT contain any nested `wsp:Policy` elements. There SHALL be no more than one instance of any XACMLAssertion of a given name included in any one `wsp:Policy` alternative.

Where multiple `Requirements` or `Capabilities` elements are present, they SHALL be treated as equivalent to multiple logically OR'd XACMLAssertions, each containing one unique combination of the `Requirements` and `Capabilities` elements. Every set of `Capabilities` MUST be compatible with every set of `Requirements` in the same XACMLAssertion and every set of `Requirements` MUST be compatible with every set of `Capabilities` in the same XACMLAssertion.

3.2 ws-xacml:Requirements

The `Requirements` element in an XACMLAssertion specifies requirements on any entity that may interact with the entity publishing the XACMLAssertion with respect to the policy target associated with the XACMLAssertion; the requirements are stated in terms of a policy vocabulary. The specification for each named XACMLAssertion SHALL specify whether or not instances of the named XACMLAssertion SHALL be complete specifications of requirements with respect to the stated policy vocabulary. If the specification is not required to be complete, then the requirements in this element MAY be only a subset

of the requirements that will be applied at the time another entity actually interacts with the publishing entity and thus the requirements MAY be only advisory to the other party. Likewise, if the specification is not required to be complete, the policy vocabulary specified in this element MAY be only a subset of the vocabulary that will be applied at the time another entity actually interacts with the publishing entity and thus the policy vocabulary MAY also be only advisory to the other party.

An XACMLAssertion that contains no `Requirements` element SHALL indicate that the entity publishing the XACMLAssertion has no requirements on the other entity that it is willing to publish. Such an XACMLAssertion SHALL always match any instance of an XACMLAssertion of the same name in another policy.

```
<element name="Requirements" xsi:type="RequirementsType"/>
<complexType name="RequirementsType">
   <sequence>
      <element name="Vocabulary" type="anyURI" minOccurs="1"
maxOccurs="unbounded"/>
      <choice>
         <element ref="xacml:Policy" minOccurs="0" />
         <element ref="xacml:PolicySet" minOccurs="0" />
         <element ref="xacml:Apply" minOccurs="0"
                  maxOccurs="unbounded" />
      </choice>
   </sequence>
</complexType>
```

The following describes the elements contained in the schema above.

`/ws-xacml:`*`XACMLAssertion`*`/ws-xacml:Requirements/ws-xacml:Vocabulary`

This element SHALL contain the identifier of a policy vocabulary: a set of policy variables or an XML document schema that contains policy-related information. There MAY be more than one such element, each specifying a different vocabulary. The policy vocabulary of the `Requirements` SHALL be the union of all the policy variables and schemas specified in the `Vocabulary` elements of the `Requirements`. Each policy variable referenced from the `Policy`, `PolicySet`, or `Apply` elements in the `Requirements` section SHALL be a member of one of these policy vocabularies. There MAY be policy vocabularies identified in `Vocabulary` elements for which there are no references from any of the `Policy`, `PolicySet`, or `Apply` elements.

A `Requirements` element that contains only `Vocabulary` elements SHALL be a notification that an XACML policy using those policy vocabularies MAY be applied at the time of the Web Service interaction associated with the XACMLAssertion, even though the policy itself is not stated in the XACMLAssertion. This MAY be used as a mechanism for notifying consumers of the XACMLAssertion that certain policy vocabularies are used by the publisher of the XACMLAssertion.

If a vocabulary variable from the policy vocabulary in a Vocabulary element is not referenced from the `Policy`, `PolicySet`, or `Apply` elements in the `Requirements`, then the XACMLAssertion SHALL be interpreted as placing no requirements on that variable's values: all values for that variable SHALL be acceptable according to the requirements of the XACMLAssertion.

If two XACMLAssertions with different names that apply to the same policy target contain the same `Requirements` policy vocabulary identifiers, care MUST be taken to ensure that the requirements on those policy vocabularies are consistent.

`/ws-xacml:`*`XACMLAssertion`*`/ws-xacml:Requirements/xacml:Policy`

This element is an XACML `Policy` instance representing requirements on the policy vocabulary with respect to the policy target.

`/ws-xacml:`*`XACMLAssertion`*`/ws-xacml:Requirements/xacml:PolicySet`

This element is an XACML `PolicySet` instance representing requirements on the policy vocabulary with respect to the policy target.

`/ws-xacml:`*`XACMLAssertion`*`/ws-xacml:Requirements/xacml:Apply`

This element is an XACML `Apply` element representing a requirement on a variable from the policy vocabulary with respect to the policy target. There MAY be any number of such XACML `Apply` elements. Each `Apply` element SHALL represent a constraint on some policy variable, expressed as an XACML `AttributeDesignator` or `AttributeSelector`, that MUST be satisfied in order to meet the requirements of the entity publishing this XACMLAssertion. The constraints SHALL be logically `AND`'d together; that is, all constraints MUST be satisfied in order for the `Requirements` element to be satisfied.

A `Requirements` element using `Apply` elements SHALL be semantically equivalent to an `xacml:Policy` instance of the following form, and MAY be evaluated using standard XACML policy semantics when converted into this form:

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

3.3 ws-xacml:Capabilities

The `Capabilities` element in an XACMLAssertion SHALL specify policy-related capabilities of the entity publishing this XACMLAssertion with respect to the policy target, in conjunction with the stated requirements; the capabilities SHALL be stated in terms of declared policy vocabularies. The capabilities SHALL 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">
   <sequence>
      <element name="Vocabulary" type="anyURI" minOccurs="1"
maxOccurs="unbounded"/>
      <choice>
         <element ref="xacml-context:Request" minOccurs="0"
   maxOccurs="unbounded"/>
         <element ref="xacml-context:ResourceContent" minOccurs="0"
   maxOccurs="unbounded"/>
         <element ref="xacml:Apply" minOccurs="0"
   maxOccurs="unbounded" />
      </choice>
   </sequence>
</complexType>
```

The following describes the elements contained in the schema above.

`/ws-xacml:`*`XACMLAssertion`*`/ws-xacml:Requirements/ws-xacml:Vocabulary`

This element SHALL contain the identifier of a policy vocabulary: a set of policy variables or a document schema containing policy-related information. There MAY be more than one such element, each specifying a different vocabulary. The policy vocabulary of the `Capabilities` SHALL be the union of all the policy variables and schema instances specified in its `Vocabulary` elements. Each

policy variable referenced from the `Request` or `Apply` elements in the `Capabilities` section SHALL be a member of one of these policy vocabularies. There MAY be vocabularies identified for which there are no references from any of the `Request` or `Apply` elements.

`/ws-xacml:`*`XACMLAssertion`*`/ws-xacml:Capabilities/xacml-context:Request`

This element is an XACML `Request Context` instance. An instance of this element is semantically equivalent to a `Capabilities` element where, for each `Attribute` 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>
   <AttributeDesignator AttributeId="AttributeId of Attribute[i]"
DataType="DataType of Attribute[i]"/>
</Apply>
```

`/ws-xacml:`*`XACMLAssertion`*`/ws-xacml:Capabilities/xacml-context:ResourceContent`

This element is an XML document representing an instance of a policy vocabulary. For example, this could be an actual instance of a P3P policy. There MAY be any number of such elements.

`/ws-xacml:`*`XACMLAssertion`*`/ws-xacml:Capabilities/xacml:Apply`

This element is an XACML `Apply` element instance. There may be any number of such elements. Each instance SHALL represent a set of values for a particular policy variable that the entity publishing this XACMLAssertion is able to provide in conjunction with the `Requirements` specified in the same XACMLAssertion. The constraints SHALL be logically `OR`'d together; that is, any subset of these constraints MAY be used to satisfied the `Requirements` specified in another XACMLAssertion.

3.4 ws-xacml:Apply element format

Each `Apply` element in a `Capabilities` or `Requirements` SHALL represent 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 SHALL conform to the following requirements.

Exactly one policy vocabulary variable SHALL be referenced in each `Apply` element. That is, there SHALL 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` functions specified in Section 8.

The `FunctionId` attribute in an `Apply` element SHALL 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.

TBD: we could add a new "value preference" XML attribute to <Apply> elements that would allow the entity taking the intersection to choose a particular set of values that are most preferred by one of the parties. This attribute would indicate, where multiple numeric or date/dateTime values are acceptable (for example: classification <= 5), whether the policy issuer prefers the lowest mutually acceptable value or the highest. For discrete sets of acceptable values, the elements in the set can be treated as ordered for purposes of preference. WS-PolicyConstraints defines such a value preference attribute.

3.5 Obligations

An obligation in an XACMLAssertion SHALL be represented by an XACML `Attribute`, specified as either an `AttributeDesignator` or an `AttributeSelector`. Imposing an obligation SHALL be expressed by including an `Apply` element equating the obligation `Attribute` with the value associated with the obligation in the *XACMLAssertion*`/Requirements` element, either as part of a `Policy` or `PolicySet`, or as a separate *`XACMLAssertion`*`/Requirements/Apply` element. An entity able and willing to fulfill an obligation SHALL include an instance of the obligation `Attribute` having the appropriate value in the *`XACMLAssertion`*`/Capabilities/Request` element, or SHALL include an `Apply` element equating the obligation `Attribute` to the appropriate value in its *`XACMLAssertion`*`/Capabilities` element.

3.6 XPath expressions in XACMLAssertion instances

In an XACMLAssertion, when the format of a Requirements or Capabilities element is a sequence of Apply elements, the syntax of XPath expressions used in any AttributeSelectors is restricted. Each XPath expression SHALL be absolute, not relative. Query operators SHALL NOT be used. Ordered element selectors SHALL NOT be used.

> *Non-normative:* These restrictions are necessary to support the matching and intersection of constraints specified in Appendix A. Without these restrictions, it is not possible to tell whether two XPath expressions refer to the same nodeset in all instances of a schema, and without knowing that information, it is not possible to detect constraints on the same sets of nodes.
>
> *TBD:* These restrictions might not be sufficient; more study is needed. Contributions on this topic from theorists and XPath experts are invited.

3.7 Evaluation of an XACMLAssertion instance

An XACMLAssertion SHALL evaluate to “True”, indicating that the Assertion is satisfied, if and only if all Requirements in the Assertion evaluate to “True” against an actual instance of (that is, set of values for a subset of the policy variables in) the policy vocabulary related to the policy target. Otherwise, an XACMLAssertion SHALL evaluate to “False”, indicating that the Assertion is not satisfied.

3.8 Matching two XACMLAssertion instances

At the generic policy engine level, any XACMLAssertion in one policy SHALL match any XACMLAssertion of the same name in another policy. For example, any `XACMLPrivacyAssertion` in one `wsp:Policy` alternative SHALL match any `XACMLPrivacyAssertion` in another `wsp:Policy` alternative. At the XACML domain-specific level, an XACMLAssertion in one policy SHALL match an XACMLAssertion of the same name in another policy only if all of the `Requirements` of each entity are satisfied by the `Capabilities` of the other entity. An empty XACMLAssertion, or an XACMLAssertion with a missing `Requirements` or `Capabilities` element SHALL always match another XACMLAssertion: it SHALL represent an acknowledgment or requirement that an XACML policy MAY be applied to the target of the Web Services policy when a Web Service interaction occurs.

Table 1 in this Section describes the rules for matching the `Requirements` of one XACMLAssertion against the `Capabilities` of another XACMLAssertion. The vocabularies of the two XACMLAssertions SHALL match if and only if, for each `Vocabulary` element in the `Requirements` of the 1st XACMLAssertion, there is a matching `Vocabulary` element in the `Capabilities` of the 2nd XACMLAssertion.

1st XACMLAssertion Requirements Format	2nd XACMLAssertion Capabilities Format	Match algorithm
Missing `Requirements` element	Any	Always “”`True`”. The publisher of the 1st XACMLAssertion is not willing to provide `Requirements`, but is indicating that an XACML policy MAY be applied at the time of the Web Services interaction.
Any	Missing `Capabilities` element.	Always “”`True`”. The publisher of the 2nd XACMLAssertion is not willing to provide `Capabilities`, but is acknowledging that an XACML policy MAY be applied at the time of the Web Services interaction.
Empty `Requirements` element	Any	Always “`True`”.
`xacml:Policy` or `xacml:PolicySet`	`xacml-context:Request` or `xacml-context:ResourceContent`	“`True`” if the vocabularies match AND if there is at least one `Request` or `ResourceContent` element in the `Capabilities` that when evaluated against the `Policy` or `PolicySet` according to the XACML Standard returns “`Permit`”; otherwise “`False`”.
`xacml:Policy` or `xacml:PolicySet`	`xacml:Apply`	Not defined.
`xacml:Apply`	`xacml-context:Request` or `xacml-context:ResourceContent`	Consider the `Requirements` to be the semantically equivalent XACML `Policy` as described in Section 3.2. “`True`” if the vocabularies match AND if there is at least one `Request` or `ResourceContent` element in the `Capabilities` that when evaluated against the `Policy` according to the XACML Standard returns “`Permit`”; otherwise “`False`”.
`xacml:Apply`	`xacml:Apply`	“`True`” if the vocabularies match AND if, for each `Apply` element in the `Requirements`, there is at least one `Apply` element in the `Capabilities` for which the intersection defined in Appendix A is non-empty; otherwise “`False`”.

Table 1: Match algorithms for matching two XACMLAssertions

Intersection of two Apply elements

An `Apply` element from the `Requirements` element in one XACMLAssertion is **coincident** with an `Apply` element from the `Capabilities` element in another XACMLAssertion if the two XACMLAssertions have the same name and if the two `Apply` elements reference the same policy variable.

The intersection of two **coincident** Apply elements is either the empty set, 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 2 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.

Non-normative Apply element intersection result example

The following Apply element indicates that any value greater than "5" for the Attribute "*some-namespace:*A" is acceptable:

```
<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">
    <AttributeDesignator AttributeId="some-namespace:A"
DataType="http://www.w3.org/2001/XMLSchema#integer"/>
  </Apply>
  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"/>5</AttributeVa
lue>
</Apply>
```

The following Apply element indicates that any value greater than "8" for the Attribute "*some-namespace:*A" is acceptable:

```
<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">
    <AttributeDesignator AttributeId="some-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 intersection of these two Apply elements, which is equivalent to the 2nd input element in this case, is an Apply function indicating that any value greater than "8" is acceptable. This exactly describes the set of integer values that are acceptable to both input Apply elements. The resulting Apply element is:

```
<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">
    <AttributeDesignator AttributeId="some-namespace:A"
DataType="http://www.w3.org/2001/XMLSchema#integer"/>
  </Apply>
  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"/>8</AttributeVa
lue>
</Apply>
```

A client using this intersected result MAY select any value greater than "8" for actual use in the Web Services interaction. For example, if the Web Services protocol calls for the client to indicate a specific mutually acceptable value for use in a specific interaction, the client could specify "9" by using the 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">
    <AttributeDesignator AttributeId="some-namespace:A"
DataType="http://www.w3.org/2001/XMLSchema#integer"/>
  </Apply>
  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"/>8</AttributeVa
lue>
</Apply>
```

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

Intersection of two XACMLAssertion instances

The intersection of two XACMLAssertion instances of the same name SHALL represent a non-binding agreement concerning the policy to be applied between the two parties publishing the XACMLAssertions. The intersection is defined only for the cases listed in Table 1 in this Section. The two XACMLAssertions match and are compatible only if, for each XACMLAssertion, the `Requirements` match the `Capabilities` in the other XACMLAssertion according to the rules defined previously in this Section.

If the two XACMLAssertions match, then the result of the intersection is two new XACMLAssertions, 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 computed according to the rules defined previously in this Section. The `Requirements` section for a given party contains the original `Requirements` of that given party since every requirement must be satisfied in order for the two XACMLAssertions to match.

The intersection of two XACMLAssertions MAY represent multiple policy alternatives, one for each combination of variable values accepted by the various constraints in the XACMLAssertion. For example, if a resulting XACMLAssertion contains only a `Requirements` element with an `Apply` function that allows the value of the P3P `PURPOSE` element to contain any combination of values in the set {“`current`”, “`admin`”}, then that XACMLAssertion represents three policy alternatives: one where the value is {“current”}, one where the value is {“admin”}, and one where the value is {“`current`”, “`admin`”}. A service MAY propose a specific alternative to the other party by conveying a new XACMLAssertion in which each `Apply` element in the `Requirements` and `Capabilities` sections is replaced by an equality function between the `AttributeDesignator` or `AttributeSelector` from the original `Apply` element and one of the literal values that satisfies the original `Apply` element. This Profile does not define how such a specific alternative is selected nor how such a specific alternative is conveyed to the other party.

4 XACMLAuthzAssertion

An XACMLAuthzAssertion is a named element derived from the XACMLAssertionAbstractType for use in expressing authorization and access control policy requirements and capabilities.

For security reasons, many entities are unwilling to publish their complete authorization and access control policies. Therefore, an XACMLAuthzAssertion MAY not be a complete specification of an entity's authorization and access control policy. A service MAY choose to publish all or a subset of the XACML authorization or access control policy it will apply with respect to particular policy targets. Likewise, a client MAY choose to make its authorization and access control policy applying to one or more policy targets available for comparison with potential services. Clients and services MAY also use instances of this assertion internally to reflect more complete policies with respect to the policy target. A published instance of an XACMLAuthzAssertion SHALL be a subset of any more complete authorization policy used internal to the client or service; the instance MAY omit any requirements or capabilities that the publisher is not able and willing to make public.

```
<element name="XACMLAuthzAssertion"
xsi:type="XACMLAssertionAbstractType"/>
```

The following describes the elements defined in the schema above and additional semantics associated with the elements defined by the XACMLAssertionAbstractType when used in an XACMLAuthzAssertion.

/ws-xacml:XACMLAuthzAssertion

This element identifies an XACMLAuthzAssertion. This element is of type XACMLAssertionAbstractType and SHALL conform to the generic semantics of that type. It SHALL contain only vocabularies, capabilities, and requirements related to authorization or access control.

/ws-xacml:XACMLAuthzAssertion/ws-xacml:Requirements/ws-xacml:Vocabulary

/ws-xacml:XACMLAuthzAssertion/ws-xacml:Requirements/ws-xacml:Vocabulary

This element specifies a particular policy vocabulary to which the XACMLAuthzAssertion Requirements or Capabilities applies. Various authorization and access control policy vocabularies MAY be used with an XACMLAuthzAssertion.

5 XACMLPrivacyAssertion

An XACMLPrivacyAssertion is a named element derived from the XACMLAssertionAbstractType for use in expressing privacy policy requirements and capabilities. An XACMLPrivacyAssertion MAY be used to express a client's privacy requirements (privacy preferences and privacy obligations) on a service, or a service's privacy requirements (privacy or confidentiality policies and obligations) on a client, or the client's or services capabilities for complying with such requirements.

```
<element name="XACMLPrivacyAssertion"
xsi:type="XACMLAssertionAbstractType"/>
```

The following describes the elements contained in the schema above.

/ws-xacml:XACMLPrivacyAssertion

> This element identifies an XACMLPrivacyAssertion. It is derived from the XACMLAssertionAbstractType and conforms to the generic semantics defined for instances of that type. An XACMLPrivacyAssertion SHALL contain only privacy-related policy vocabularies, capabilities, and requirements, although the vocabularies MAY include general vocabulary variables necessary to indicate the resources, information, or behaviors to which the privacy requirements or capabilities apply.

5.1 P3P Privacy Policy Vocabulary

One of the policy vocabularies that MAY be used to specify an XACMLPrivacyAssertion is a P3P 1.0 [P3P] full schema. A client MAY use P3P 1.0 as the policy vocabulary for its Capabilities to describe personal identifying information that the client is able and willing to provide to the service. A service MAY use P3P 1.0 as the policy vocabulary for its Requirements to describe personal identifying information that the client is required to provide to the service. A client MAY use P3P 1.0 as the policy vocabulary for its Requirements to express the client's privacy preferences regarding use of the client's personal identifying information by the service. A service MAY use P3P 1.0 as the policy vocabulary for its Capabilities to express the service's willingness and ability to use the client's personal identifying information only in accordance with certain client requirements.

Constraints that apply to P3P 1.0 full schema instances SHALL be expressed using XACML AttributeSelectors where the RequestContextPath has a prefix value of "//P3P10/POLICIES/". The remainder of the RequestContextPath SHALL be interpreted as applying to the P3P 1.0 POLICY instances associated with the Web Services policy target.

5.2 Client Privacy Policy Constraints

Client information release and obligation capabilities to satisfy a service

In this Profile, a client MAY describe privacy-related information that the client is able and willing to provide (“preferences”), and all privacy-related obligations that the client is able and willing to fulfill with respect to a policy target by using XACML constraints on the contents of a policy vocabulary in the Capabilities element of the client's XACMLPrivacyAssertion that applies to that target. These capabilities are offered in combination with any requirements stated in the Requirements element of the same XACMLPrivacyAssertion.

Client requirements and obligations on the service

A client MAY describe privacy-related requirements and obligations it has on the service by using XACML constraints on the contents of a policy vocabulary in the Requirements element of the client's

XACMLPrivacyAssertion. A client MAY include other constraints in the Requirements element of its XACMLPrivacyAssertion to limit the applicability of the privacy policy constraints in the Capabilities element of its XACMLPrivacyAssertion. The capabilities are offered in combination with any requirements stated in the Requirements element of the same XACMLPrivacyAssertion.

5.3 Service Privacy Policy Constraints

Service requirements and obligations on the client

In this Profile, a service MAY describe privacy-related information that the service requires the client to provide, and obligations that the service requires the client to fulfill by using XACML constraints on the contents of a policy vocabulary in the Requirements element of the service's XACMLPrivacyAssertion.

Service requirement and obligation capabilities to satisfy a client

A service MAY describe privacy-related requirements and obligations that the service is able and willing to fulfill for the client by using XACML constraints on the contents of a policy vocabulary in the Capabilities element of the service's XACMLPrivacyAssertion. A service MAY include other constraints in the Requirements element of the same XACMLPrivacyAssertion to limit the applicability of the privacy policy constraints in the Capabilities element of its XACMLPrivacyAssertion.

6 Non-normative Examples of XACMLAssertions

[This Section is non-normative]

XACMLPrivacyAssertion privacy policy constraints describe what acceptable instances of the service's privacy policy can look like. The following examples show XACMLPrivacyAssertion instances that describe acceptable instances of a service's privacy policy expressed as a P3P 1.0 full schema instance.

Example 1: Client Privacy Preferences and Confidentiality Capabilities

The following XACMLPrivacyAssertion is an example of expressing a client's privacy requirements as well as the client's willingness and ability to conform to data retention requirements on data provided by the service. These are expressed using constraints on the acceptable values in the service's P3P 1.0 full policy schema instance and on one of the new Attributes defined in this Profile.

In this example, the client requires that the service collect and retain only the data found in standard HTTP access logs (that is, only “nonident”ified data), only for purposes of of the “current” transaction, “admin”istration, or transaction “tailoring” to the client and with only the service and its agents allowed to receive such data. The client also requires the service to be able and willing to correct any harm done. The client is able and willing to delete data obtained from the service on or after 30 days.

```
<XACMLPrivacyAssertion>

  <Requirements>
    <Vocabulary>urn:oasis:names:tc:xacml:3.0:vocabulary:p3p10full</
Vocabulary>
    <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:must-
be-present">
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">//P3P10/POLICIES
/POLICY/ACCESS/nonident</AttributeValue>
    </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:xpath-
expression-subset">
      <AttributeSelector
DataType="http://www.w3.org/2001/XMLSchema#string
RequestContextPath="//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/*"/>
      <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:xpath-expression-
bag">
        <AttributeValue
DataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
expression">//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/current</Attr
ibuteValue>
        <AttributeValue
DataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
expression">//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/admin</Attrib
uteValue>
        <AttributeValue
DataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
expression">//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/tailoring</At
tributeValue>
      </Apply>
    </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:xpath-
expression-subset">
      <AttributeSelector
DataType="http://www.w3.org/2001/XMLSchema#string"
```

```
RequestContextPath="//P3P10/POLICIES/POLICY/STATEMENT/RECIPIENT/*"
/>
      <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:xpath-expression-
bag">

        <AttributeValue
DataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
expression">//P3P10/POLICIES/POLICY/STATEMENT/RECIPIENT/ours</Attri
buteValue>
      </Apply>
    </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:must-
be-present">
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">//P3P10/POLICIES
/POLICY/DISPUTES/correct</AttributeValue>
    </Apply>
  </Requirements>

  <Capabilities>
    <Vocabulary>
urn:oasis:names:tc:xacml:3.0:vocabulary:xacml</vocabulary>
    <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-
than-or-equal">
      <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-one-and-
only">
        <AttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:3.0:action:max-data-
retention-days"
DataType="http://www.w3.org/2001/XMLSchema#integer"/>
      </Apply>
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">30</AttributeVa
lue>
    </Apply>
  </Capabilities>

</XACMLPrivacyAssertion>
```

Example 2: Service Privacy Policy

The following `XACMLPrivacyAssertion` is an example of a service's privacy policy. This represents the privacy-related information that the service requires the client to provide, the privacy-related obligations that the service requires the client to fulfill, and the privacy-related requirements and obligations that the service is able and willing to meet for the client. These are all expressed using constraints on the acceptable values of a service's P3P 1.0 full policy schema instance.

In this example, the service is willing and able to collect only non-identified data for purposes of the “current” transaction or for “tailoring”, for use only by the service and its agents, with the client data retained only so long as required to satisfy the stated purpose. The service is able and willing to accept an obligation to correct any harm done. The service requires that the client retain data provided by the service for no more than 30 days.

```
<XACMLPrivacyAssertion>

  <Requirements>
    <Vocabulary>
```

```
urn:oasis:names:tc:xacml:3.0:vocabulary:xacml</vocabulary>
    <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-
than-or-equal">
      <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-or-
equal">
        <AttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:3.0:action:max-data-
retention-days"
DataType="http://www.w3.org/2001/XMLSchema#integer"/>
      </Apply>
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">30</AttributeVa
lue>
    </Apply>
  </Requirements>

  <Capabilities>
    <Vocabulary>urn:oasis:names:tc:xacml:3.0:vocabulary:p3p10full</
Vocabulary>
    <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:must-
be-present">
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">//P3P10/POLICIES
/POLICY/ACCESS/nonident</AttributeValue>
    </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:xpath-
expression-subset">
      <AttributeSelector
DataType="http://www.w3.org/2001/XMLSchema#string
RequestContextPath="//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/*"/>
      <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:xpath-expression-
bag">
        <AttributeValue
DataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
expression">//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/current</Attr
ibuteValue>
        <AttributeValue
DataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
expression">//P3P10/POLICIES/POLICY/STATEMENT/PURPOSE/tailoring</At
tributeValue>
      </Apply>
    </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:xpath-
expression-subset">
      <AttributeSelector
DataType="http://www.w3.org/2001/XMLSchema#string"
RequestContextPath="//P3P10/POLICIES/POLICY/STATEMENT/RECIPIENT/*"
/>
      <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:xpath-expression-
bag">
        <AttributeValue
DataType="urn:oasis:names:tc:xacml:2.0:data-type:xpath-
expression">//P3P10/POLICIES/POLICY/STATEMENT/RECIPIENT/ours</Attri
buteValue>
      </Apply>
    </Apply>
```

```
      <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:must-
   be-present">
        <AttributeValue
   DataType="http://www.w3.org/2001/XMLSchema#string">//P3P10/POLICIES
   /POLICY/DISPUTES/correct</AttributeValue>
      </Apply>
    </Capabilities>

   </XACMLPrivacyAssertion>
```

This service's policy matches the client preferences in the previous example. Every client requirement is satisfied by a service capability, and every service requirement is satisfied by a client capability. Note that this service collects data for only a subset of the purposes allowed by the client, and the maximum retention days for the service's data is a range that has a non-empty intersection with the range of values supported by the client.

7 Conveying XACML Attributes in a SOAP Message

At the time a service is invoked, the 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 service MAY use XACML to make such an authorization decision.

When a 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.

7.1 xacml-samlp:XACMLAuthzDecisionQuery

The first way in which `Attributes` may be provided to a 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 service. The `xacml-samlp:XACMLAuthzDecisionQuery` SHOULD be signed by an entity that the service trusts to authenticate the enclosed XACML `Attributes`.

The service MAY use the Attributes in such an `xacml-samlp:XACMLAuthzDecisionQuery` as part of evaluating XACML policies related to the Web Service interaction. The service SHOULD verify that the query is signed by an entity that the 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 service.

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

7.2 saml:Attribute

A second way in which `Attributes` may be provided to a service is in the form of `saml:Attribute` elements in a s`aml: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* [XACML-SAML] by an XACML Context Handler for use by a PDP associated with the service in evaluating XACML policies related to the Web Service interaction.

8 New XACML Functions

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

8.1 urn:oasis:names:tc:xacml:3.0:function:must-be-present

This function SHALL take one or more XACML `AttributeDescriptors` or `AttributeValues` with data type "`http://www.w3.org/2001/XMLSchema#string`" as its arguments and SHALL return a Boolean result. The `AttributeValue` strings SHALL be interpreted as XPath expressions. The result SHALL be "`True`" if all the bags containing the values of the specified `AttributeDescriptors` and all the nodesets representing the XPath expressions contain at least one value; otherwise, the result SHALL be "`False`".

Where multiple arguments are present, all of the specified Attributes MUST be present.

This function is equivalent to applying the "`bag-size`" function to each `AttributeDescriptor` and `AttributeValue` and comparing the results using the "`urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal`" function to the integer value "1". If all such applications of the "`bag-size`" function are equal to "1", then the function SHALL return "`True`"; otherwise, the function SHALL return "`False`".

8.2 urn:oasis:names:tc:xacml:3.0:function:must-not-be-present

This function SHALL take one or more XACML `AttributeDescriptors` or `AttributeValues` with data type "`http://www.w3.org/2001/XMLSchema#string`" as its arguments and SHALL return a Boolean result. The `AttributeValue` strings SHALL be interpreted as an XPath expressions. The result SHALL be "`true`" if all the bags containing the values of the specified `AttributeDescriptor` elements and all the nodesets representing the XPath expressions contain no values (that is, they are empty bags or sets); otherwise, the result SHALL be "`false`".

When multiple arguments are present, all of the specified Attributes MUST be absent.

This function is equivalent to applying the "`bag-size`" function to each of the `AttributeDescriptors` and XPath expressions and comparing the results using the "`function:integer-equal`" function to the integer value "0". If all such applications of the "`-bagsize`" function are equal to "0", then the function SHALL return "True"; otherwise, the function SHALL return "False".

8.3 urn:oasis:names:tc:xacml:3.0:function:limit-scope:all

This function is used to place multiple constraints on descendants of all nodes selected by a specified XPath expression; it serves as a universal qualifier. In order to make intersection tractable, this function SHALL only be used in a `Requirements` element. This function MAY be nested arbitrarily deeply.

This function SHALL take two or more parameters, where the first parameter is an XACML `AttributeValue` with data type "`http://www.w3.org/2001/XMLSchema#string`", and the remaining parameters are XACML `Apply` elements. It SHALL return a result of data type "`http://www.w3.org/2001/XMLSchema#`Boolean". The `AttributeValue` used as the values of the first parameter SHALL be interpreted as an XPath expression. The XACML `Apply` element parameters SHALL specify policy constraints on elements that are descendants of the nodes selected by the XPath expression in the first parameter; that is, the XPath expression in the first parameter SHALL be the context node for the remaining parameters. The result SHALL be "`true`" if all the `Apply` element constraints are "true" when applied to every node selected by the XPath expression in the first parameter; otherwise, the result SHALL be "`false`".

The XPath expressions used in a scope-limiting function SHALL conform to the restrictions on XPath expressions specified in Section 3.6, with the exception that only the outer-most scope-limiting function's first parameter SHALL specify an absolute XPath expression. Each other XPath expression SHALL be relative to the context node specified by its outer scope-limiting function.

Non-normative example:

```
<Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:limit-
scope:all">
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">//header/security/k
ey-info</AttributeValue>
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-
is-in">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">96</AttributeValue
>
        <AttributeSelector RequestContextPath="key-length"
DataType=”http://www.w3.org/2001/XMLSchema#integer">
   </Apply>
   <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:string-is-
in">
       <AttributeSelector RequestContextPath="algorithm"
DataType="http://www.w3.org/2001/XMLSchema#string">
        <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">DES-
CBC</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">AES</AttributeValue
>
        </Apply>
    </Apply>
</Apply>
```

This function returns “`true`” only if every “`//header/security/key-info`” element has a “`key-length`” child with value `96` and an “`algorithm`” child with value “`DES-CBC`” or “`AES`”.

8.4 urn:oasis:names:tc:xacml:3.0:function:limit-scope:atLeastOne

This function is used to place multiple constraints on descendants of at least one of the nodes selected by a specified XPath expression; it serves as an existential qualifier. In order to make intersection tractable, this function SHALL only be used in a `Requirements` element. This function MAY be nested arbitrarily deeply.

This function SHALL take two or more parameters, where the first parameter is an XACML `AttributeValue` with data type “`http://www.w3.org/2001/XMLSchema#string`”, and the remaining parameters are XACML `Apply` elements. It SHALL return a result of data type “`http://www.w3.org/2001/XMLSchema#`Boolean”. The `AttributeValue` used as the values of the first parameter SHALL be interpreted as an XPath expression. The XACML `Apply` element parameters SHALL specify policy constraints on elements that are descendants of the nodes selected by the XPath expression in the first parameter; that is, the XPath expression in the first parameter SHALL be the context node for the remaining parameters. The result SHALL be “`true`” if all the `Apply` element constraints are “true” when applied to at least one of the nodes selected by the XPath expression in the first parameter; otherwise, the result SHALL be “`false`”.

The XPath expressions used in a scope-limiting function SHALL conform to the restrictions on XPath expressions specified in Section 3.6, with the exception that only the outer-most scope-limiting function's first parameter SHALL specify an absolute XPath expression. Each other XPath expression SHALL be relative to the context node specified by its outer scope-limiting function.

Non-normative example:

```
<Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:limit-
scope:atLeastOne">
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">//header/security/k
ey-info</AttributeValue>
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-
is-in">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">96</AttributeValue
>
        <AttributeSelector RequestContextPath="key-length"
DataType=”http://www.w3.org/2001/XMLSchema#integer">
   </Apply>
   <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:string-is-
in">
       <AttributeSelector RequestContextPath="algorithm"
DataType="http://www.w3.org/2001/XMLSchema#string">
        <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">DES-
CBC</AttributeValue>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">AES</AttributeValue
>
        </Apply>
    </Apply>
</Apply>
```

This function returns “`true`” only if at least one “`//header/security/key-info`” element has a “`key-length`” child with value `96` and an “`algorithm`” child with value “`DES-CBC`” or “`AES`”.

9 New XACML Attribute Identifiers

The following new Attribute Identifiers are defined by this Profile.

9.1 Data confidentiality and retention Attributes

This Profile defines the following new AttributeId values for use in XACML Attributes describing confidentiality and retention periods with respect to information other than personal identifying information. When used in an XACMLAssertion, constraints using such Attributes apply to all information provided to a receiving entity.

Maximum data retention days

`urn:oasis:names:tc:xacml:3.0:action:max-data-retention-days`

An `Attribute` using this `AttributeId` SHALL define the maximum number of days that data provided by one party SHALL be retained by the receiving party or by other parties to which the receiving party discloses the data. The `DataType` of this `Attribute` SHALL be "`http://www.w3.org/2001/XMLSchema#integer`".

Constraints on this Attribute in the `Capabilities` element of an XACMLAssertion or in an XACML `Request` apply to data received by the issuer of the XACMLAssertion or the `Request`. Constraints on this Attribute in the `Requirements` element of an XACMLAssertion or in an XACML `Policy` or `PolicySet` apply to data provided by the issuer of the XACMLAssertion, `Policy`, or `PolicySet`.

Data disclosure

`urn:oasis:names:tc:xacml:3.0:action:data-disclosure`

An Attribute using this `AttributeId` SHALL define whether and how data provided by one party SHALL or SHALL NOT be disclosed to another party. The `DataType` of this `Attribute` SHALL be "`http://www.w3.org/2001/XMLSchema#string`".

Constraints on this Attribute in the `Capabilities` element of an XACMLAssertion or in an XACML `Request` apply to data received by the issuer of the XACMLAssertion or the `Request`. Constraints on this Attribute in the `Requirements` element of an XACMLAssertion or in an XACML `Policy` or `PolicySet` apply to data provided by the issuer of the XACMLAssertion, `Policy`, or `PolicySet`.

The allowed values for this Attribute are the names of the child elements defined for the `RECIPIENT` element in the P3P 1.0 schema [P3P]: "`ours`", "`delivery`", "`same`", other-`recipient`", "`unrelated`", and "`public`". These values SHALL have the same interpretation as in P3P, except that they apply to all data provided by one entity to the receiving entity, and not just to personal identifying information. This Attribute MAY have two additional values:

– `none`: the receiving entity SHALL NOT disclose any received data to any other entity. This value SHALL NOT be used together with any other values for this Attribute.

– named: the receiving entity MAY disclose any received data to entities identified in an Attribute with `AttributeId` `urn:oasis:names:tc:xacml:3.0:subject:recipient`.

Named recipient

`urn:oasis:names:tc:xacml:3.0:subject:recipient`

An Attribute using this `AttributeId` SHALL specify the identity of an entity that is to receive data associated with the policy target. In order for data to be released to any entity identified in this Attribute, the `urn:oasis:names:tc:xacml:3.0:action:data-disclosure` Attribute SHALL also have a value of "`named`".

Constraints on this Attribute in the `Capabilities` element of an XACMLAssertion or in an XACML `Request` apply to data received by the issuer of the XACMLAssertion or the `Request`. Constraints on this Attribute in the `Requirements` element of an XACMLAssertion or in an XACML `Policy` or `PolicySet` apply to data provided by the issuer of the XACMLAssertion, `Policy`, or `PolicySet`.

10 Vocabulary Identifiers

The following identifiers are defined for use in the `Vocabulary` element of an XACMLAssertion `Requirements` or `Capabilities` element to identify all or part of the policy vocabulary used in that element. Additional Vocabulary identifiers MAY be defined and used for other policy vocabularies.

10.1 XACML Standard Attributes

```
urn:oasis:names:tc:xacml:3.0:vocabulary:xacml
```

A `Vocabulary` element having this value SHALL be used when the associated XACMLAssertion `Requirements` or `Capabilities` element contains references to any `AttributeId` values defined in the XACML 1.0, 2.0, and 3.0 OASIS Standard specifications.

10.2 P3P 1.0 Full Schema

```
urn:oasis:names:tc:xacml:3.0:vocabulary:p3p10full
```

A `Vocabulary` element having this value SHALL be used when the associated XACMLAssertion `Requirements` or `Capabilities` element contains a P3P 1.0 full schema instance or contains references to nodes in such an instance. This identifier is used only to place constraints on the `POLICY/ACCESS`, `POLICY/DISPUTES`, `POLICY/REMEDIES`, `POLICY/STATEMENT/NON-IDENTIFIABLE`, `POLICY/STATEMENT/PURPOSE`, `POLICY/STATEMENT/RECIPIENT`, `POLICY/STATEMENT/RETENTION`, `POLICY/STATEMENT/DATA-GROUP/DATA` elements and their children. The `POLICY/STATEMENT/DATA-GROUP/DATA` element and its children MAY be used to describe specific personal information that an entity requires from another entity or to describe personal information that an entity is willing and able to provide to another entity.

TBD: need to address the `POLICY/STATEMENT/NON-IDENTIFIABLE`, since that interacts with the other `POLICY/STATEMENT` elements.

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 policy variable, then they are said to be **coincident.** If two constraints in XACMLAssertions 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 2 below, then the two constraints are incompatible, and the XACMLAssertions 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 2.

Table 2 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. For readability, only the function-specific suffix for the function identifiers is shown, rather than the complete namespace. The namespace prefix for the FunctionId values shown is either "`urn:oasis:names:tc:xacml:1.0:function`", "`urn:oasis:names:tc:xacml:2.0:function`", or "`urn:oasis:names:tc:xacml:3.0:function`". None of these functions occur in differing forms in the various XACML Standard specifications, so the names are unambiguous.

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 Capabilities constraint" column, there is no single replacement `Apply` element: the constraints are compatible, but not combinable. In these cases, the original `Apply` elements SHALL NOT be modified by this step in the procedure.

The **referenced XPath expression** of the `FunctionId` in an `Apply` element is defined to be the XPath expression in the first child of the `Apply` element if the `FunctionId` is one of the two "`limit-scope`" functions; otherwise it is the XPath expression in the `RequestContextPath` attribute of the `AttributeSelector` that is one of the children of the `Apply` element. The **effective XPath expression** of the `FunctionId` in any given `Apply` element is recursively defined to be the effective XPath expression of any outer "`limit-scope`" function with the XPath expression referenced by the `FunctionId` in the given `Apply` element.

∩ means set intersection.

⊆ means "is a subset of".

	Requirements constraint	Capabilities constraint	Compatibility test	Replacement Capabilities 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	*type*-set-equals(a,b)	*type*-set-equals(a,c)	b == c	*type*-set-equals(a,b)	
19	*type*-set-equals(a,b)	*type*-subset(a,c)	b ⊆ c	*type*-set-equals(a,b)	
20	*type*-subset(a,b)	*type*-subset(a,c)	∩ (b,c) ≠ 0	*type*-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	limit-scope:atLeastOne(p,f1,f2, ...)			See below.	
30	limit-scope:all(p,f1,f2,...)			See below	

Table 2 - Intersection of coincident constraints

TBD: It would be very powerful to allow some form of regular-expression match in policy constraints. Computing intersections of restricted regular expressions is possible, but might be too expensive to justify inclusion.

Handling of limit-scope:atLeastOne during intersection

The result of taking the intersection of an `Apply` element having a `FunctionId` that is "`limit-scope:atLeastOne`" in a `Requirements` element with a given `Capabilities` element is computed as follows. There are two cases: 1) the `Capabilities` element contains a `Request` or `ResourceContent` element, or 2) the `Capabilities` element contains one or more `Apply` elements.

For case 1), let g[j] be the nodes in the `Capabilities` XML document that are selected by the effective XPath expression of the "`limit-scope:atLeastOne`" function. Let f[i] be the `Apply` elements in the parameters of the "`limit-scope:atLeastOne`" function. The intersection of the `Apply` element having the `FunctionId` that is "`limit-scope:atLeastOne`" with the `Capabilities` is the nodeset containing the matching nodes selected by each f[i] for each g[j] where all f[i] in g[j] have non-empty intersections with the `Capabilities` XML document according to Table 2 above. If the intersection of the `Apply` element having the `FunctionId` that is "`limit-scope:atLeastOne`" with the `Capabilities` is empty, then the `Requirements` and the `Capabilities` are not compatible.

For case 2), the intersection of the "`limit-scope:atLeastOne`" `Apply` function with the `Capabilities` includes the intersection of each Apply element f[i] that is a child of the "`limit-scope:atLeastOne`" `Apply` function with a coincident `Apply` element in the Capabilities according to Table 2 above. If for any f[i] there is no coincident constraint, or if the intersection with the coincident constraint is empty, then the `Requirements` and the `Capabilities` are not compatible.

Handling of limit-scope:all during intersection

The result of taking the intersection of an `Apply` element having a `FunctionId` that is "`limit-scope:all`" in a `Requirements` element with a given `Capabilities` element is computed as follows. There are two cases: 1) the `Capabilities` element contains a `Request` or `ResourceContent` element, or 2) the `Capabilities` element contains one or more `Apply` elements.

For case 1), let g[j] be the nodes in the `Capabilities` XML document that are selected by the effective XPath expression of the "`limit-scope:atLeastOne`" function. Let f[i] be the `Apply` elements in the parameters of the "`limit-scope:atLeastOne`" function. The intersection of the g[j] with the `Capabilities` is the nodeset containing the matching nodes selected by each f[i] in g[j] if all f[i] in g[j] have non-empty intersections with the `Capabilities` XML document according to Table 2 above. If the intersection of any g[j] with the `Capabilities` XML document is empty, then the `Requirements` and the `Capabilities` are not compatible. Otherwise, the intersection of the `Apply` element having the `FunctionId` that is "`limit-scope:atLeastOne`" with the `Capabilities` contains all the nodes in the intersections of all g[j] with the `Capabilities` XML document.

For case 2), the intersection of the "`limit-scope:all`" `Apply` function with the `Capabilities` includes the intersection of each Apply element f[i] that is a child of the "`limit-scope:all`" `Apply` function with a coincident `Apply` element in the Capabilities according to Table 2 above. If for any f[i] there is no coincident constraint, or if the intersection with the coincident constraint is empty, then the `Requirements` and the `Capabilities` are not compatible.

Appendix B. Acknowledgments

The semantics for intersection of XACMLAssertions 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

Rev	Date	By Whom	What
05	9 Oct 2006	Anne Anderson	Initial version, based on *WS-PolicyConstraints*, Working Draft 06, and the *XACML profile for Web-services*, Working Draft 04.
06	17 Nov 2006	Anne Anderson	Added privacy policies section. Clarified text elsewhere based on comments against previous revision.
07	5 Dec 2006	Anne Anderson	Added a schema file.
08	12 Dec 2006	Anne Anderson	In schema, made Requirements and Capabilities unbounded; added ResourceContent element option to Capabilities (allows inclusion of an actual P3P instance, for example). Defined limit-scope:atLeastOne (same as old limit-scope) and limit-scope:all functions, and restricted them to use in Requirements to make intersection tractable. Modified P3P profile to use P3P 1.0. Separated XACMLPrivacyAssertion from XACMLAuthzAssertion and created abstract type from which both are derived. Added more explanatory material, including some diagrams and a table of the matching algorithms. Defined two Vocabulary identifiers. Defined several confidentiality and data retention AttributeIds. Specified restrictions on XPath expressions used in AttributeSelectors and limit-scope functions.

Appendix D. Non-Normative Text