Abstract

The Web Services Policy 1.5 - Framework provides a general purpose model and corresponding syntax to describe the policies of entities in a Web services-based system.

Web Services Policy Framework defines a base set of constructs that can be used and extended by other Web services specifications to describe a broad range of service requirements and capabilities.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This is a Last Call Working Draft of the Web Services Policy 1.5 - Framework specification. The Last Call period ends 12 January 2007. This Working Draft was produced by the members of the Web Services Policy Working Group which is part of the W3C Web Services Activity. The Working Group expects to
advance this Working Draft to Recommendation Status.

A list of changes in this version of the document [p.34] and a diff-marked version against the previous version of this document are available. Major changes in this version of the document encompass the addition of a new feature described in section [4.4 Ignorable Policy Assertions][p.24] and the definition of an XML Media Type in section [A. The application/wspolicy+xml Media Type][p.29]. The Working Group solicits feedback on a proposal to incorporate the versioning guidance from the [Web Services Policy 1.5 - Primer] document into in a future version of this document.

The Working Group is tracking all comments via [Bugzilla] and highly prefers to receive comments via this system. If access to Bugzilla is not feasible, you may send your comments to the mailing list public-ws-policy-comments@w3.org mailing list (public archive). Each Bugzilla entry and email message should contain only one comment. All comments on this specification should be made following the Description for Issues of the Working Group.

This document was produced by a group operating under the [5 February 2004 W3C Patent Policy]. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

---

**Table of Contents**

1. [Introduction][p.3]
   1.1 [Example][p.4]

2. [Notations and Terminology][p.4]
   2.1 [Notational Conventions][p.4]
   2.2 [Extensibility][p.5]
   2.3 [XML Namespaces][p.5]
   2.4 [Terminology][p.6]

3. [Policy Model][p.8]
   3.1 [Policy Assertion][p.8]
   3.2 [Policy Alternative][p.9]
   3.3 [Policy][p.9]
   3.4 [Policies of Entities in a Web Services Based System][p.10]

4. [Policy Expression][p.10]
   4.1 [Normal Form Policy Expression][p.11]
   4.2 [Policy Identification][p.12]
   4.3 [Compact Policy Expression][p.13]
     4.3.1 [Optional Policy Assertions][p.14]
     4.3.2 [Policy Assertion Nesting][p.15]
     4.3.3 [Policy Operators][p.17]
1. Introduction

Web Services Policy 1.5 - Framework defines a framework and a model for expressing policies that refer to domain-specific capabilities, requirements, and general characteristics of entities in a Web services-based system.

A policy [p.9] is a collection of policy alternatives, where a policy alternative [p.9] is a collection of policy assertions. A policy assertion [p.8] represents an individual requirement, capability, or other property of a behavior. A policy expression [p.10] is an XML Infoset representation of a policy, either in a normal form or in an equivalent compact form. Some policy assertions specify traditional requirements and capabilities that will ultimately manifest on the wire (e.g., authentication scheme, transport protocol selection). Other policy assertions have no wire manifestation yet are critical to proper service selection and usage (e.g., privacy policy, QoS characteristics). Web Services Policy 1.5 - Framework provides a single policy language to allow both kinds of assertions to be expressed and evaluated in a consistent manner.

Web Services Policy 1.5 - Framework does not specify policy discovery or policy attachments. A policy attachment [p.10] is a mechanism for associating policy with one or more policy scopes, where a policy scope [p.12] is a collection of policy subjects to which a policy may apply. A policy subject [p.8] is an entity (e.g., an endpoint, message, resource, interaction) with which a policy can be associated. Other specifications are free to define technology-specific mechanisms for associating policy with various entities and resources. Web Services Policy 1.5 - Attachment [Web Services Policy Attachment [p.31]] defines such mechanisms, especially for associating policy with arbitrary XML elements. [XML 1.0 [p.31]]
. WSDL artifacts [WSDL 1.1][p.32], [WSDL 2.0 Core Language][p.33], and UDDI elements [UDDI API 2.0][p.32], [UDDI Data Structure 2.0][p.32], [UDDI 3.0][p.32].

1.1 Example


Example 1-1. Use of Web Services Policy with security policy assertions.

(01)  <wsp:Policy
        xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
        xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
(02)    <wsp:ExactlyOne>
(03)      <wsp:All>
(04)       <sp:SignedParts/>
(05)       <sp:Body/>
(06)       </sp:SignedParts/>
(07)      </wsp:All>
(08)    </wsp:ExactlyOne>
(09)    <wsp:All>
(10)       <sp:EncryptedParts/>
(11)       <sp:Body/>
(12)       </sp:EncryptedParts/>
(13)    </wsp:All>
(14)  </wsp:Policy>

Lines (03-06) represent one policy alternative for signing a message body.

Lines (08-11) represent a second policy alternative for encrypting a message body.

Lines (02-13) illustrate the ExactlyOne policy operator. Policy operators group policy assertions into policy alternatives. A valid interpretation of the policy above would be that an invocation of a Web service will either sign or encrypt the message body.

2. Notations and Terminology

This section specifies the notations, namespaces, and terminology used in this specification.

2.1 Notational Conventions

This specification uses the following syntax within normative outlines:

- The syntax appears as an XML instance, but values in *italics* indicate data types instead of literal values.
- Characters are appended to elements and attributes to indicate cardinality:
The character "|" is used to indicate a choice between alternatives.

- The characters "(" and ")" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.

- This document relies on the XML Information Set [XML Information Set][p.31]. Information item properties are indicated by the style [infoset property].

- XML namespace prefixes (see Table 2-1[p.6]) are used to indicate the namespace of the element or attribute being defined.

- The ellipses characters "…" are used to indicate a point of extensibility that allows other Element or Attribute Information Items.

Elements and Attributes defined by this specification are referred to in the text of this document using XPath 1.0 [XPATH 1.0] expressions. Extensibility points are referred to using an extended version of this syntax:

- An element extensibility point is referred to using {any} in place of the element name. This indicates that any element name can be used, from any namespace.

- An attribute extensibility point is referred to using @{any} in place of the attribute name. This indicates that any attribute name can be used, from any namespace.

Normative text within this specification takes precedence over normative outlines, which in turn take precedence over the XML Schema [XML Schema Structures][p.31] descriptions.

### 2.2 Extensibility

Within normative outlines, ellipses (i.e., "…") indicate a point of extensibility that allows other Element or Attribute Information Items. Information Items MAY be added at the indicated extension points but MUST NOT contradict the semantics of the element information item indicated by the [parent] or [owner] property of the extension. If an Attribute Information Item is not recognized, it SHOULD be ignored; if an Element Information Item is not recognized, it MUST be treated as an assertion.

### 2.3 XML Namespaces

This specification uses a number of namespace prefixes throughout; they are listed in Table 2-1[p.6]. Note that the choice of any namespace prefix is arbitrary and not semantically significant (see XML [Namespaces][p.31]).
Table 2-1. Prefixes and Namespaces used in this specification

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsp</td>
<td><a href="http://www.w3.org/2006/07/ws-policy">http://www.w3.org/2006/07/ws-policy</a></td>
<td>This specification</td>
</tr>
<tr>
<td>wsu</td>
<td><a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd</a></td>
<td>[WS-Security 2004] [p.31]</td>
</tr>
<tr>
<td>xs</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
<td>[XML Schema Structures] [p.31], [XML Schema Datatypes] [p.32]</td>
</tr>
</tbody>
</table>

All information items defined by this specification are identified by the XML namespace URI [XML Namespaces] [p.31] http://www.w3.org/2006/07/ws-policy. A normative XML Schema [XML Schema Structures] [p.31], XML Schema Datatypes [p.32] document can be obtained by dereferencing the XML namespace URI.

It is the intent of the W3C Web Services Policy Working Group that the Web Services Policy 1.5 - Framework and Web Services Policy 1.5 - Attachment XML namespace URI will not change arbitrarily with each subsequent revision of the corresponding XML Schema documents but rather change only when a subsequent revision, published as a WD, CR or PR draft results in non-backwardly compatible changes from a previously published WD, CR or PR draft of the specification.

Under this policy, the following are examples of backwards compatible changes that would not result in assignment of a new XML namespace URI:

- Addition of new global element, attribute, complexType and simpleType definitions.
- Addition of new elements or attributes in locations covered by a previously specified wildcard.
- Modifications to the pattern facet of a type definition for which the value-space of the previous definition remains valid or for which the value-space of the vast majority of instances would remain valid.
- Modifications to the cardinality of elements (i.e. modifications to minOccurs or maxOccurs attribute value of an element declaration) for which the value-space of possible instance documents conformant to the previous revision of the schema would still be valid with regards to the revised cardinality rule.

2.4 Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [IETF RFC 2119] [p.31].

We introduce the following terms that are used throughout this document:
An **ignorable policy assertion** is an assertion that may be ignored for policy intersection (as defined in [2.4 Terminology](#).)

A **nested policy expression** is a policy expression that is an Element Information Item in the [children] property of a policy assertion.

A **policy** is a potentially empty collection of policy alternatives.

A **policy alternative** is a potentially empty collection of policy assertions.

A **policy alternative vocabulary** is the set of all policy assertion types within the policy alternative.

A **policy assertion** represents an individual requirement, capability, or other property of a behavior.

A **policy assertion parameter** qualifies the behavior indicated by a policy assertion.

A **policy assertion type** represents a class of policy assertions and implies a schema for the assertion and assertion-specific semantics.

A **policy attachment** is a mechanism for associating policy with one or more policy scopes.

A **policy expression** is an XML Infoset representation of a policy, either in a normal form or in an equivalent compact form.
A **policy scope** is a collection of **policy subjects** to which a policy may apply.

A **policy subject** is an entity (e.g., an endpoint, message, resource, operation) with which a policy can be associated.

A **policy vocabulary** is the set of all **policy assertion types** used in a policy.

### 3. Policy Model

This section defines an abstract model for policies and for operations upon policies.

The descriptions below use XML Infoset terminology for convenience of description. However, this abstract model itself is independent of how it is represented as an XML Infoset.

#### 3.1 Policy Assertion

[Definition: A **policy assertion** represents an individual requirement, capability, or other property of a behavior.] A **policy assertion** identifies a behavior that is a requirement or capability of a **policy subject**. [Definition: A **policy subject** is an entity (e.g., an endpoint, message, resource, operation) with which a policy can be associated.] Assertions indicate domain-specific (e.g., security, transactions) semantics and are expected to be defined in separate, domain-specific specifications.

An assertion MAY indicate that it is an ignorable policy assertion (see [4.4 Ignorable Policy Assertions](p.24)). [Definition: An **ignorable policy assertion** is an assertion that may be ignored for policy intersection (as defined in [4.5 Policy Intersection](p.24)).] By default, an assertion is not ignorable for policy intersection.

Assertions are typed by the authors that define them. [Definition: A **policy assertion type** represents a class of **policy assertions** and implies a schema for the assertion and assertion-specific semantics.] The **policy assertion type** is identified only by the XML Infoset [namespace name] and [local name] properties (that is, the qualified name or QName) of the root Element Information Item representing the assertion. Assertions of a given type MUST be consistently interpreted independent of their **policy subjects**.

Authors MAY define that an assertion contains a **policy expression** (as defined in [4. Policy Expression](p.10)) as one of its [children]. Nested policy expression(s) are used by authors to further qualify one or more specific aspects of the original assertion. For example, security policy authors may define an assertion describing a set of security algorithms to qualify the specific behavior of a security binding assertion.

The XML Infoset of a **policy assertion** MAY contain a non-empty [attributes] property and/or a non-empty [children] property. Such properties are [policy assertion parameters] and MAY be used to parameterize the behavior indicated by the assertion. **Policy assertion parameters** qualify the behavior indicated by a **policy assertion**. For example, an assertion identifying support for a
specific reliable messaging mechanism might include an attribute information item to indicate how long an endpoint will wait before sending an acknowledgement.

Authors should be cognizant of the processing requirements when defining complex assertions containing policy assertion parameters [p.8] or nested policy expressions [p.15]. Specifically, authors are encouraged to consider when the identity of the root Element Information Item alone is enough to convey the requirement or capability.

3.2 Policy Alternative

[Definition: A policy alternative is a potentially empty collection of policy assertions [p.8].] An alternative with zero assertions indicates no behaviors. An alternative with one or more assertions indicates behaviors implied by those, and only those assertions. [Definition: A policy vocabulary is the set of all policy assertion types [p.8] used in a policy.] [Definition: A policy alternative vocabulary is the set of all policy assertion types [p.8] within the policy alternative [p.9].] When an assertion whose type is part of the policy’s vocabulary is not included in a policy alternative, the policy alternative without the assertion type indicates that the assertion will not be applied in the context of the attached policy subject. See the example in Section 4.3.1 Optional Policy Assertions [p.14]

 Assertions within an alternative are not ordered, and thus aspects such as the order in which behaviors (indicated by assertions) are applied to a subject [p.8] are beyond the scope of this specification. However, authors can write assertions that control the order in which behaviours are applied.

A policy alternative MAY contain multiple assertions of the same type. Mechanisms for determining the aggregate behavior indicated by the assertions (and their Post-Schema-Validation Infoset (PSVI) (See XML Schema Part 1 XML Schema Structures [p.31]) content, if any) are specific to the assertion type and are outside the scope of this document.

3.3 Policy

[Definition: A policy is a potentially empty collection of policy alternatives [p.9].] A policy with zero alternatives contains no choices; a policy with one or more alternatives indicates choice in requirements or capabilities within the policy.

Alternatives are not ordered, and thus aspects such as preferences between alternatives in a given context are beyond the scope of this specification.

Alternatives within a policy may differ significantly in terms of the behaviors they indicate. Conversely, alternatives within a policy may be very similar. In either case, the value or suitability of an alternative is generally a function of the semantics of assertions within the alternative and is therefore beyond the scope of this specification.
3.4 Policies of Entities in a Web Services Based System

Applied to a Web services based system, policy is used to convey conditions on an interaction between entities (requester application, provider service, Web infrastructure component, etc). An interaction involves one or more message exchanges between two entities. It is the responsibility of authors to define the interaction scope of an assertion including any constraints on the policy subjects to which the assertion may be attached and a clear specification of the message(s) within that interaction scope to which the assertion applies.

Any entity in a Web services based system may expose a policy to convey conditions under which it functions. Satisfying assertions in the policy usually results in behavior that reflects these conditions. For example, if two entities - requester and provider - expose their policies, a requester might use the policy of the provider to decide whether or not to use the service. A requester MAY choose any alternative since each is a valid configuration for interaction with the service, but a requester MUST choose only a single alternative for an interaction with a service since each represents an alternative configuration.

A policy assertion is supported by an entity in the Web services based system if and only if the entity satisfies the requirement (or accommodates the capability) corresponding to the assertion. A policy alternative is supported by an entity if and only if the entity supports all the assertions in the alternative. And, a policy is supported by an entity if and only if the entity supports at least one of the alternatives in the policy. Note that although policy alternatives are meant to be mutually exclusive, it cannot be decided in general whether or not more than one alternative can be supported at the same time.

Note that an entity may be able to support a policy even if the entity does not understand the type of each assertion in the vocabulary of the policy; the entity only has to understand the type of each assertion in the vocabulary of a policy alternative the entity supports. This characteristic is crucial to versioning and incremental deployment of new assertions because this allows a provider’s policy to include new assertions in new alternatives while allowing entities to continue to use old alternatives in a backward-compatible manner.

4. Policy Expression

This section describes how to convey policy in an interoperable form, using the XML Infoset representation of a policy. [Definition: A policy expression is an XML Infoset representation of a policy, either in a normal form or in an equivalent compact form.] Other subsections below describe several important aspects related to policy expression, namely (i) Normal form of a policy expression (ii) Compact form of a policy expression (iii) Identification of policy expressions and (iv) Policy intersection.

The normal form of a policy expression is the most straightforward Infoset representation; equivalent, alternative Infosets allow compactly expressing a policy through a number of constructs.

This specification does not define processing for arbitrary Policy Element Information Items in any context other than as an Element Information Item in the [children] property of an Element Information Item that is in the [children] property of an element Information Item defined in section 4.1 below.
4.1 Normal Form Policy Expression

To facilitate interoperability, this specification defines a normal form for policy expressions that is a straightforward XML Infoset representation of a policy, enumerating each of its alternatives that in turn enumerate each of their assertions. The schema outline for the normal form of a policy expression is as follows:

(01) <wsp:Policy ... >
(02)   <wsp:ExactlyOne>
(03)     ( <wsp:All> ( <Assertion ...> ... </Assertion> )* </wsp:All> )*  
(04)   </wsp:ExactlyOne>
(05) </wsp:Policy>

The following describes the Element Information Items defined in the schema outline above:

/wsp:Policy  
A policy expression.

/wsp:Policy/wsp:ExactlyOne  
A collection of policy alternatives. If there are no Element Information Items in the [children] property, there are no admissible policy alternatives, i.e., no behavior is admissible.

/wsp:Policy/wsp:ExactlyOne/wsp:All  
A policy alternative; a collection of policy assertions. If there are no Element Information Items in the [children] property, this is an admissible policy alternative that is empty, i.e., no behavior is specified.

/wsp:Policy/wsp:ExactlyOne/wsp:All/*  
XML Infoset representation of a policy assertion.

/wsp:Policy/@{any}  
Additional attributes MAY be specified but MUST NOT contradict the semantics of the [owner element]; if an attribute is not recognized, it SHOULD be ignored.

If an assertion in the normal form of a policy expression contains a nested policy expression, the nested policy expression MUST contain at most one policy alternative (see 4.3.2 Policy Assertion Nesting).

To simplify processing and improve interoperability, the normal form of a policy expression SHOULD be used where practical.

For example, the following is the normal form of a policy expression.
4.2 Policy Identification

A policy expression MAY be associated with an IRI. The schema outline for attributes to associate an IRI is as follows:

```
<wsp:Policy ( Name="xs:anyURI" )?
  ( wsu:Id="xs:ID" | xml:id="xs:ID" )?
  ... >
</wsp:Policy>
```

The following describes the Attribute Information Items listed and defined in the schema outline above:

/wsp:Policy/@Name

The identity of the policy expression as an absolute IRI. If omitted, there is no implied value. This IRI MAY be used to refer to a policy from other XML documents using a policy attachment mechanism such as those defined in WS-PolicyAttachment. [Definition: A policy attachment is a mechanism for associating policy [p.9] with one or more policy scopes [p.12].] [Definition: A policy scope is a collection of policy subjects to which a policy may apply.]

/wsp:Policy/@wsu:Id | @xml:id

The identity of the policy expression as an ID within the enclosing XML document. If omitted, there is no implied value. The constraints of the XML 1.0 ID type MUST be met. To refer to this policy expression, an IRI-reference MAY be formed using this value per Section 4.2 of WS-Security when @wsu:Id is used.
The use of xml:id attribute in conjunction with Canonical XML 1.0 is inappropriate as described in Appendix C of xml:id Version 1.0 [XML ID/p.31] and thus this combination must be avoided (see [C14N 1.0 Note/p.32]). For example, a policy expression identified using xml:id attribute should not be signed using XML Digital Signature when Canonical XML 1.0 is being used as the canonicalization method.

The following example illustrates how to associate a policy expression with the absolute IRI "http://www.example.com/policies/P1":

(01) <wsp:Policy
    Name="http://www.example.com/policies/P1"
    xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
(02)   <!-- Details omitted for readability -->
(03) </wsp:Policy>

The following example illustrates how to associate a policy expression with the IRI-reference "#P1":

(01) <wsp:Policy
    wsu:Id="P1"
    xmlns:wsp="http://www.w3.org/2006/07/ws-policy"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" >
(02)   <!-- Details omitted for readability -->
(03) </wsp:Policy>

4.3 Compact Policy Expression

To express a policy in a more compact form while still using the XML Infoset, this specification defines three constructs: an attribute to decorate an assertion, semantics for recursively nested policy operators, and a policy reference/inclusion mechanism. Each is described in the subsections below.

To interpret a compact expression in an interoperable form, a compact expression may be converted to the corresponding normal form expression by the following procedure:

1. Start with the document element property D of the Document Information Item of the policy expression. The namespace name of D is always "http://www.w3.org/2006/07/ws-policy". In the base case, the local name property of D is "Policy"; in the recursive case, the local name property of D is "Policy", "ExactlyOne", or "All".

2. Expand Element Information Items in the children property of D that are policy references per Section 4.3.5 Policy Inclusion.[p.23]

3. Convert each Element Information Item C in the children property of D into normal form.

   1. If the namespace name property of C is "http://www.w3.org/2006/07/ws-policy" and the local name property of C is "Policy", "ExactlyOne", or "All", C is an expression of a policy operator; normalize C by recursively applying this procedure.
4.3 Compact Policy Expression

2. Otherwise the Element Information Item C is an assertion; normalize C per Sections [4.3.1 Optional Policy Assertions][p.14] and [4.3.2 Policy Assertion Nesting][p.15].

4. Apply the policy operator indicated by D to the normalized Element Information Items in its [children] property and construct a normal form per Section [4.3.3 Policy Operators][p.17].

Note that an implementation may use a more efficient procedure and is not required to explicitly convert a compact expression into the normal form as long as the processing results are indistinguishable from doing so.

4.3.1 Optional Policy Assertions

To indicate that a policy assertion [p.8] is optional, this specification defines an attribute that is a compact authoring style for expressing a pair of [alternatives][p.9], one with and one without that assertion. The schema outline for this attribute is as follows:

```
(01) <Assertion ( wsp:Optional="xs:boolean" )? ...> ... </Assertion>
```

The following describes the Attribute Information Item defined in the schema outline above:

```
/Assertion/@wsp:Optional
```

If the actual value (See XML Schema Part 1 [XML Schema Structures][p.31]) is true, the expression of the assertion is semantically equivalent to the following:

```
(01) <wsp:ExactlyOne>
(02)   <wsp:All> <Assertion ...> ... </Assertion> </wsp:All>
(03)   <wsp:All />
(04) </wsp:ExactlyOne>
```

If the actual value (See XML Schema Part 1 [XML Schema Structures][p.31]) is false, the expression of the assertion is semantically equivalent to the following:

```
(01) <wsp:ExactlyOne>
(02)   <wsp:All> <Assertion ...> ... </Assertion> </wsp:All>
(03) </wsp:ExactlyOne>
```

Omitting this attribute is semantically equivalent to including it with a value of false. Policy expressions should not include this attribute with a value of false, but policy parsers must accept this attribute with a value of false.

For example, the following compact policy expression:

```
(01) <wsp:Policy
    xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
    xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
(02)   <sp:IncludeTimestamp wsp:Optional="true" />
(03) </wsp:Policy>
```
is equivalent to the following normal form policy expression:

(01) <wsp:Policy
   xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
   xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
(02)   <wsp:ExactlyOne>
(03)     <wsp:All>
(04)       <sp:IncludeTimestamp />
(05)     </wsp:All>
(06)     <wsp:All />
(07)   </wsp:ExactlyOne>
(08) </wsp:Policy>

The @wsp:Optional attribute in Line (02) of the first policy expression indicates that the assertion in Line (02) is to be included in a policy alternative whilst excluded from another; it is included in Lines (03-05) and excluded in Line (06). Note that @wsp:Optional does not appear in the normal form of a policy expression.

4.3.2 Policy Assertion Nesting

Any [policy assertion][p.8] MAY contain a[policy expression][p.10]. [Definition: A nested policy expression is a[policy expression][p.10] that is an Element Information Item in the [children] property of a[policy assertion][p.8].] The schema outline for a[nested policy expression][p.15] is:

(01) <Assertion ...>
(02) ...
(03) ( <wsp:Policy ...> ... </wsp:Policy> )?
(04) ...
(05) </Assertion>

The following describes additional processing constraints on the outline listed above:

/Assertion/wsp:Policy

This indicates that the assertion contains a nested policy expression. If there is no wsp:Policy Element Information Item in the [children] property, the assertion has no nested policy expression.

Note: if the schema outline for an assertion type requires a nested policy expression but the assertion does not further qualify one or more aspects of the behavior indicated by the assertion type (i.e., no assertions are needed in the nested policy expression), the assertion MUST include an empty <wsp:Policy/> Element Information Item in its [children] property; as explained in Section [4.3.3 Policy Operators][p.17], this is equivalent to a nested policy expression with a single alternative that has zero assertions. The reason for requiring at least an empty <wsp:Policy/> Element above is to ensure that two assertions of the same type will always be compatible and an intersection would not fail (see Section [4.5 Policy Intersection][p.24]).

Note: This specification does not define processing for arbitrary wsp:Policy Element Information Items in the descendants of an assertion parameter, e.g., in the [children] property of one of the [children] as in <Lorem><Ipsum><wsp:Policy> ... </wsp:Policy></Ipsum></Lorem>.
Policy assertions containing a nested policy expression are normalized recursively. The nesting of a policy expression (and a wsp:Policy child) is retained in the normal form, but in the normal form, each nested policy expression contains at most one policy alternative. If an assertion A contains a nested policy expression E, and if E contains more than one policy alternative, A is duplicated such that there are as many instances of A as choices in E, and the nested policy expression of a duplicate A contains a single choice. This process is applied recursively to the assertions within those choices and to their nested policy expression, if any. Intuitively, if a compact policy is thought of as a tree whose branches have branches etc, in the normal form, a policy is a stump with straight vines.

For example, consider the following policy expression with nested policy expressions in a compact form:

```
<sp:TransportBinding>
  <sp:AlgorithmSuite>
    <sp:ExactlyOne>
      <sp:Basic256Rsa15 />
      <sp:TripleDesRsa15 />
    </wsp:ExactlyOne>
  </wsp:Policy>
  <sp:TransportToken>
    <sp:HttpsToken RequireClientCertificate="false" />
  </wsp:Policy>
</sp:TransportBinding>
```

Lines (02-18) in this policy expression contain a single transport binding security policy assertion; within its nested policy expression (Lines 03-17), is an algorithm suite assertion (Lines 04-11) whose nested policy expression (Lines 05-10) contains two policy alternatives (Lines 07-08). Generally, a nested policy expression implies recursive processing; in the example above, the behavior indicated by the transport binding assertion requires the behavior indicated by one of the assertions within the algorithm suite assertion.

The normalized form of the example above is equivalent to the following:

```
<wsp:ExactlyOne>
  <wsp:All>
    <sp:TransportBinding>
      <sp:AlgorithmSuite>
        <sp:Basic256Rsa15 />
      </wsp:Policy>
    </sp:TransportToken>
  </wsp:Policy>
</wsp:Policy>
```
In the listing above, the transport binding and its nested policy expression have been duplicated once for each of the nested alternatives in Lines (07-08) of the compact policy. The first alternative (Lines 03-18) contains a single nested algorithm suite alternative (Line 08) as does the second alternative (Lines 19-34 and 24).

4.3.3 Policy Operators

Policies are used to convey a set of capabilities, requirements, and general characteristics of entities (see 1. Introduction). These are generally expressible as a set of policy alternatives. Policy operators (wsp:Policy, wsp:All and wsp:ExactlyOne) are used to group policy assertions into policy alternatives. In some instances, complex policies expressed in normal form can get relatively large and hard to manage. To compactly express complex policies, policy operators MAY be recursively nested; that is, one or more instances of wsp:Policy, wsp:All, and/or wsp:ExactlyOne MAY be nested within wsp:Policy, wsp:All, and/or wsp:ExactlyOne.

The following rules are used to transform a compact policy expression into a normal form policy expression:

Equivalence
Use of \textit{wsp:Policy} as an operator within a policy expression is equivalent to \textit{wsp:All}.

Empty

- \texttt{<wsp:All />} expresses a policy with zero policy assertions. Note that since \textit{wsp:Policy} is equivalent to \textit{wsp:All}, \texttt{<wsp:Policy />} is therefore equivalent to \texttt{<wsp:All />}, i.e., a policy alternative with zero assertions.
- \texttt{<wsp:ExactlyOne />} expresses a policy with zero policy alternatives.

Commutative

In line with the previous statements that policy assertions within a policy alternative and policy alternatives within a policy are not ordered (see \textbf{3.2 Policy Alternative} [p.9] and \textbf{3.3 Policy} [p.9], respectively), \textit{wsp:All} and \textit{wsp:ExactlyOne} are commutative. For example,

\begin{verbatim}
(01) <wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
\end{verbatim}

is equivalent to:

\begin{verbatim}
(01) <wsp:All> <!-- assertion 2 --> <!-- assertion 1 --> </wsp:All>
\end{verbatim}

and:

\begin{verbatim}
(01) <wsp:ExactlyOne>
(02)  <!-- assertion 1 --> <!-- assertion 2 -->
(03) </wsp:ExactlyOne>
\end{verbatim}

is equivalent to:

\begin{verbatim}
(01) <wsp:ExactlyOne>
(02)  <!-- assertion 2 --> <!-- assertion 1 -->
(03) </wsp:ExactlyOne>
\end{verbatim}

Associative

\textit{wsp:All} and \textit{wsp:ExactlyOne} are associative. For example,

\begin{verbatim}
(01) <wsp:All>
(02)  <!-- assertion 1 -->
(03) <wsp:All> <!-- assertion 2 --> </wsp:All>
(04) </wsp:All>
\end{verbatim}

is equivalent to:

\begin{verbatim}
(01) <wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
\end{verbatim}

and:

\begin{verbatim}
(01) <wsp:ExactlyOne>
(02)  <!-- assertion 1 -->
(03) <wsp:ExactlyOne> <!-- assertion 2 --> </wsp:ExactlyOne>
(04) </wsp:ExactlyOne>
\end{verbatim}
is equivalent to:

(01) <wsp:ExactlyOne>
(02) <!-- assertion 1 --> <!-- assertion 2 -->
(03) </wsp:ExactlyOne>

Idempotent

wsp:All and wsp:ExactlyOne are idempotent. For example,

(01) <wsp:All>
(02) <wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
(03) </wsp:All>

is equivalent to:

(01) <wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>

and:

(01) <wsp:ExactlyOne>
(02) <wsp:ExactlyOne>
(03) <!-- assertion 1 --> <!-- assertion 2 -->
(04) </wsp:ExactlyOne>
(05) </wsp:ExactlyOne>

is equivalent to:

(01) <wsp:ExactlyOne>
(02) <!-- assertion 1 --> <!-- assertion 2 -->
(03) </wsp:ExactlyOne>

Distributive

wsp:All distributes over wsp:ExactlyOne. For example,

(01) <wsp:All>
(02) <wsp:ExactlyOne>
(03) <!-- assertion 1 -->
(04) <!-- assertion 2 -->
(05) </wsp:ExactlyOne>
(06) </wsp:All>

is equivalent to:

(01) <wsp:ExactlyOne>
(02) <wsp:All>
(03) <!-- assertion 1 -->
(04) </wsp:All>
(05) <wsp:All>
(06) <!-- assertion 2 -->
(07) </wsp:All>
(08) </wsp:ExactlyOne>
Similarly by repeatedly distributing wsp:All over wsp:ExactlyOne,

(01) <wsp:All>
(02)   <wsp:ExactlyOne>
(03)     <!-- assertion 1 -->
(04)     <!-- assertion 2 -->
(05)   </wsp:ExactlyOne>
(06)   <wsp:ExactlyOne>
(07)     <!-- assertion 3 -->
(08)     <!-- assertion 4 -->
(09)   </wsp:ExactlyOne>
(10) </wsp:All>

is equivalent to:

(01) <wsp:ExactlyOne>
(02)   <wsp:All>
(03)     <!-- assertion 1 --><!-- assertion 3 --></wsp:All>
(04)   <wsp:All>
(05)     <!-- assertion 1 --><!-- assertion 4 --></wsp:All>
(06)   <wsp:ExactlyOne>
(07) </wsp:All>

Distributing wsp:All over an empty wsp:ExactlyOne is equivalent to no alternatives. For example,

(01) <wsp:All>
(02)   <wsp:ExactlyOne>
(03)     <!-- assertion 1 -->
(04)     <!-- assertion 2 -->
(05)   </wsp:ExactlyOne>
(06)   <wsp:ExactlyOne />
(07) </wsp:All>

is equivalent to:

(01) <wsp:ExactlyOne />

For example, given the following compact policy expression:

(01) <wsp:Policy
   xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
   xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
(02)   <sp:RequireDerivedKeys wsp:Optional="true" />
(03)   <wsp:ExactlyOne>
(04)     <sp:WssUsernameToken10 />
(05)     <sp:WssUsernameToken11 />
(06)   </wsp:ExactlyOne>
(07) </wsp:Policy>

Applying Section 4.3.1 Optional Policy Assertions [p.14] to @wsp:Optional in Line (02), and distributing wsp:All over wsp:ExactlyOne per Section 4.3.3 Policy Operators [p.17] for the assertions in Lines (04-05) yields:
(01) <wsp:Policy
    xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
    xmlns:wsp="http://www.w3.org/2006/07/ws-policy">
(02)     <wsp:ExactlyOne>
(03)         <wsp:All> <!-- @wsp:Optional alternative with assertion -->
(04)             <sp:RequireDerivedKeys />
(05)         </wsp:All>
(06)     <wsp:All /> <!-- @wsp:Optional alternative without -->
(07)     </wsp:ExactlyOne>
(08)     <wsp:ExactlyOne>
(09)         <wsp:All>
(10)             <sp:WssUsernameToken10 />
(11)         </wsp:All>
(12)     </wsp:All>
(13)     <wsp:All>
(14)         <sp:WssUsernameToken11 />
(15)     </wsp:All>
(16)     </wsp:ExactlyOne>
(17) </wsp:Policy>

Note that the assertion listed in Line (02) in the first listing expands into the two alternatives in Lines (03-06) in the second listing.

Finally, noting that wsp:Policy is equivalent to wsp:All, and distributing wsp:All over wsp:ExactlyOne yields the following normal form policy expression:

(01) <wsp:Policy
    xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
    xmlns:wsp="http://www.w3.org/2006/07/ws-policy">
(02)     <wsp:ExactlyOne>
(03)         <wsp:All>
(04)             <sp:RequireDerivedKeys />
(05)             <sp:WssUsernameToken10 />
(06)         </wsp:All>
(07)     <wsp:All>
(08)             <sp:RequireDerivedKeys />
(09)             <sp:WssUsernameToken11 />
(10)         </wsp:All>
(11)     </wsp:All>
(12)     <wsp:All>
(13)             <sp:WssUsernameToken10 />
(14)         </wsp:All>
(15)     <wsp:All>
(16)             <sp:WssUsernameToken11 />
(17)     </wsp:ExactlyOne>
(18) </wsp:Policy>

Note that the two alternatives listed in Lines (03-06) in the second listing are combined with the two alternatives listed in Lines (09-14) in the second listing to create four alternatives in the normalized policy, Lines (03-06), (07-10), (11-13), and (14-16).
4.3.4 Policy References

The `<wsp:PolicyReference>` element is used to reference policy expressions. The semantics of the `<wsp:PolicyReference>` element are determined by the context in which it is used (for an example, see 4.3.5 Policy Inclusion).

The schema outline for the `<wsp:PolicyReference>` element is as follows:

```
(01) <wsp:PolicyReference
(02)     URI="xs:anyURI"
(03)     ( Digest="xs:base64Binary" ( DigestAlgorithm="xs:anyURI" )? )? 
(04)     ... >
(05)     ...
(06) </wsp:PolicyReference>
```

The following describes the Attribute and Element Information Items defined in the schema outline above:

/wsp:PolicyReference

This element references a policy expression that is being referenced.

/wsp:PolicyReference/@URI

This attribute references a policy expression by an IRI. For a policy expression within the same XML Document, the reference SHOULD be an IRI-reference to a policy expression identified by an ID. For an external policy expression, there is no requirement that the IRI be resolvable; retrieval mechanisms are beyond the scope of this specification. After retrieval, there is no requirement to check that the retrieved policy expression is associated (Section 4.2 Policy Identification) with this IRI. The IRI included in the retrieved policy expression, if any, MAY be different than the IRI used to retrieve the policy expression.

/wsp:PolicyReference/@Digest

This attribute is of type `xs:boolean` and specifies the digest of the referenced policy expression. This is used to ensure the included policy is the expected policy. If omitted, there is no implied value.

/wsp:PolicyReference/@DigestAlgorithm

This optional URI attribute specifies the digest algorithms being used. This specification predefines the default algorithm below, although additional algorithms can be expressed.

<table>
<thead>
<tr>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/2006/07/ws-policy/Sha1Exc">http://www.w3.org/2006/07/ws-policy/Sha1Exc</a> (implied)</td>
<td>The digest is a SHA1 hash over the octet stream resulting from using the Exclusive XML canonicalization defined for XML Signature [XML-Signature][p.33].</td>
</tr>
</tbody>
</table>
Additional attributes MAY be specified but MUST NOT contradict the semantics of the [owner element]; if an attribute is not recognized, it SHOULD be ignored.

Additional elements MAY be specified but MUST NOT contradict the semantics of the [parent element]; if an element is not recognized, it SHOULD be ignored.

4.3.5 Policy Inclusion

In order to share assertions [p.8] across policy expressions [p.10], the wsp:PolicyReference element MAY be present anywhere a policy assertion is allowed inside a policy expression. This element is used to include the content of one policy expression in another policy expression.

When a wsp:PolicyReference element references a wsp:Policy element, then the semantics of inclusion are simply to replace the wsp:PolicyReference element with a wsp:All element whose [children] property is the same as the [children] property of the referenced wsp:Policy element. That is, the contents of the referenced policy conceptually replace the wsp:PolicyReference element and are wrapped in a wsp:All operator. Using the wsp:PolicyReference element, a policy expression MUST NOT reference itself either directly or indirectly. (Note: References that have a @Digest attribute SHOULD be validated before being included.)

In the example below two policies include and extend a common policy. In the first example there is a single policy document containing two policy assertions. The expression is given an identifier but not a fully qualified location. The second and third expressions reference the first expression by URI indicating the referenced expression is within the document.

```xml
(01) <wsp:Policy
    xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
    xmlns:wsp="http://www.w3.org/2006/07/ws-policy"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    wsu:Id="Protection" >
(02)   <sp:EncryptSignature wsp:Optional="true" />
(03)   <sp:ProtectTokens wsp:Optional="true" />
(04) </wsp:Policy>

(01) <wsp:Policy
    xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
    xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
(02)   <wsp:PolicyReference URI="#Protection" />
(03)   <sp:OnlySignEntireHeadersAndBody />
(04) </wsp:Policy>

(01) <wsp:Policy
    xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
    xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
(02)   <IncludeTimestamp />
(03)   <wsp:PolicyReference URI="#Protection" />
(04)   <sp:OnlySignEntireHeadersAndBody />
(05) </wsp:Policy>
```

23
There are times when it is desirable to "re-use" a portion of a policy expression. Generally, this can be accomplished by placing the common assertions in a separate policy expression and referencing it.

### 4.4 Ignorable Policy Assertions

The `wsp:Ignorable` attribute indicates if a policy assertion is an ignorable policy assertion. The schema outline for this attribute is as follows:

```xml
(01) <Assertion ( wsp:Ignorable="xs:boolean" )? ... > ... </Assertion>
```

The following describes the Attribute Information Item defined in the schema outline above:

```xml
/Assertion/@wsp:Ignorable
```

This attribute is of type `xs:boolean`. If the actual value (See XML Schema Part 1 [XML Schema Structures][p.31]) is true, the assertion is an ignorable policy assertion. If the actual value is false, the assertion is not an ignorable policy assertion. Omitting this attribute is semantically equivalent to including it with a value of false.

### 4.5 Policy Intersection

Policy intersection is useful when two or more parties express policy and want to limit the policy alternatives to those that are mutually compatible. For example, when a requester and a provider express requirements on a message exchange, intersection identifies compatible policy alternatives (if any) included in both requester and provider policies. Intersection is a commutative function that takes two policies and returns a policy. There are two modes for intersection: strict and lax. How the mode is selected or indicated for the policy intersection is outside the scope of this specification.

Because the set of behaviors indicated by a policy alternative depends on the domain-specific semantics of the collected assertions, determining whether two policy alternatives are compatible generally involves domain-specific processing. If a domain-specific intersection processing algorithm is required this will be known from the QNames of the specific assertion types involved in the policy alternatives. As a first approximation, an algorithm is defined herein that approximates compatibility in a domain-independent manner:

- Two policy assertions are compatible if they have the same type and
- If either assertion contains a nested policy expression, the two assertions are compatible if they both have a nested policy expression and the alternative in the nested policy expression of one is compatible with the alternative in the nested policy expression of the other.

Assertion parameters are not part of the compatibility determination defined herein but may be part of other, domain-specific compatibility processing.

- If the mode is strict, two policy alternatives A and B are compatible:
4.5 Policy Intersection

- if each assertion in A is compatible with an assertion in B, and
- if each assertion in B is compatible with an assertion in A.

If the mode is lax, two policy alternatives A and B are compatible:

- if each assertion in A that is not an ignorable policy assertion is compatible with an assertion in B, and
- if each assertion in B that is not an ignorable policy assertion is compatible with an assertion in A.

If two alternatives are compatible, their intersection is an alternative containing all of the assertions in both alternatives.

- Two policies are compatible if an alternative in one is compatible with an alternative in the other. If two policies are compatible, their intersection is the set of the intersections between all pairs of compatible alternatives, choosing one alternative from each policy. If two policies are not compatible, their intersection has no policy alternatives.

As an example of intersection, consider two input policies in normal form:

```
             xmlns:wsp="http://www.w3.org/2006/07/ws-policy">
  <!-- Policy P1 -->
  <wsp:ExactlyOne>
    <wsp:All> <!-- Alternative A1 -->
      <sp:SignedElements>
        <sp:XPath>/S:Envelope/S:Body</sp:XPath>
      </sp:SignedElements>
      <sp:EncryptedElements>
        <sp:XPath>/S:Envelope/S:Body</sp:XPath>
      </sp:EncryptedElements>
    </wsp:All>
    <wsp:All> <!-- Alternative A2 -->
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
      <sp:Header
        Namespace="http://www.w3.org/2005/08/addressing" />
    </wsp:All>
  </wsp:ExactlyOne>
</wsop:Policy>
```

The listing above contains two policy alternatives. The first alternative, (Lines 03-10) contains two policy assertions. One indicates which elements should be signed (Lines 04-06); its type is sp:SignedElements (Line 04), and its parameters include an XPath expression for the content to be signed (Line 05). The other assertion (Lines 07-09) has a similar structure: type (Line 07) and parameters (Line 08).
The second alternative (Lines 11-19) also contains two assertions, each with type (Line 12 and Line 16) and parameters (Lines 13-14 and Line 17).

As this example illustrates, compatibility between two policy assertions is based on assertion type and delegates parameter processing to domain-specific processing.

Because there is only one alternative (A2) in policy P1 with the same vocabulary — the assertions have the same type — as another alternative (A3) in policy P2, the intersection is a policy with a single alternative that contains all of the assertions in A2 and in A3.

Note that there are > 1 assertions of the type sp:SignedParts; when the behavior associated with sp:SignedParts is invoked, the contents of both assertions are used to indicate the correct behavior. Whether these two assertions are compatible depends on the domain-specific semantics of the
5. Security Considerations

It is RECOMMENDED that policies and assertions be signed to prevent tampering.

Policies SHOULD NOT be accepted unless they are signed and have an associated security token to specify the signer has the right to "speak for" the scope containing the policy. That is, a relying party shouldn’t rely on a policy unless the policy is signed and presented with sufficient credentials to pass the relying parties’ acceptance criteria.

It should be noted that the mechanisms described in this document could be secured as part of a SOAP message, SOAP 1.2 Messaging Framework using WS-Security or embedded within other objects using object-specific security mechanisms.

This section describes the security considerations that service providers, requestors, policy authors, policy assertion authors, and policy implementers need to consider when exposing, consuming and designing policy expressions, authoring policy assertions or implementing policy.

5.1 Information Disclosure Threats

A policy is used to represent the capabilities and requirements of a Web Service. Policies may include sensitive information. Malicious consumers may acquire sensitive information, fingerprint the service and infer service vulnerabilities. These threats can be mitigated by requiring authentication for sensitive information, by omitting sensitive information from the policy or by securing access to the policy. For securing access to policy metadata, policy providers can use mechanisms from other Web Services specifications such as WS-Security and WS-MetadataExchange.

5.2 Spoofing and Tampering Threats

If a policy expression is unsigned it could be easily tampered with or replaced. To prevent tampering or spoofing of policy, requestors should discard a policy unless it is signed by the provider and presented with sufficient credentials. Requestors should also check that the signer is actually authorized to express policies for the given policy subject.

5.3 Downgrade Threats

A policy may offer several alternatives that vary from weak to strong set of requirements. An adversary may interfere and remove all the alternatives except the weakest one (say no security requirements). Or, an adversary may interfere and discard this policy and insert a weaker policy previously issued by the same provider. Policy authors or providers can mitigate these threats by sun-setting older or weaker policy alternatives. Requestors can mitigate these threats by discarding policies unless they are signed by the provider.
5.4 Repudiation Threats

Malicious providers may include policy assertions in its policy whose behavior cannot be verified by examining the wire message from the provider to requestor. In general, requestors have no guarantee that a provider will behave as described in the provider’s policy expression. The provider may not and perform a malicious activity. For example, say the policy assertion is privacy notice information and the provider violates the semantics by disclosing private information. Requestors can mitigate this threat by discarding policy alternatives which include assertions whose behavior cannot be verified by examining the wire message from the provider to requestor. Assertion authors can mitigate this threat by not designing assertions whose behavior cannot be verified using wire messages.

5.5 Denial of Service Threats

Malicious providers may provide a policy expression with a large number of alternatives, a large number of assertions in alternatives, deeply nested policy expressions or chains of PolicyReference elements that expand exponentially (see the chained sample below; this is similar to the well-known DTD entity expansion attack). Policy implementers need to anticipate these rogue providers and use a configurable bound with defaults on number of policy alternatives, number of assertions in an alternative, depth of nested policy expressions, etc.

Example 5-1. Chained Policy Reference Elements

```
(01) <Policy wsu:Id="p1">
(02)   <PolicyReference URI="#p2"/>
(03)   <PolicyReference URI="#p2"/>
(04) </Policy>
(05)
(06) <Policy wsu:Id="p2" >
(07)   <PolicyReference URI="#p3"/>
(08)   <PolicyReference URI="#p3"/>
(09) </Policy>
(10)
(11) <Policy wsu:Id="p3" >
(12)   <PolicyReference URI="#p4"/>
(13)   <PolicyReference URI="#p4"/>
(14) </Policy>
(15)
(16) <!-- Policy/@wsu:Id p4 through p99 -->
(17)
(18) <Policy wsu:Id="p100" >
(19)   <PolicyReference URI="#p101"/>
(20)   <PolicyReference URI="#p101"/>
(21) </Policy>
(22)
(23) <Policy wsu:Id="p101" >
(24)   <mtom:OptimizedMimeSerialization />
(25) </Policy>
```

Malicious providers may provide a policy expression that includes multiple PolicyReference elements that use a large number of different internet addresses. These may require the consumers to establish a large number of TCP connections. Policy implementers need to anticipate such rogue providers and use a
configurable bound with defaults on number of PolicyReference elements per policy expression.

5.6 General XML Considerations

Implementers of Web Services policy language should be careful to protect their software against general XML threats like deeply nested XML or XML that contains malicious content.

6. Conformance

An element information item whose namespace name is "http://www.w3.org/2006/07/ws-policy" and whose local part is Policy or PolicyReference conforms to this specification if it is valid according to the XML Schema [XML Schema Structures [p.31]] for that element as defined by this specification [http://www.w3.org/2006/07/ws-policy.xsd] and additionally adheres to all the constraints contained in this specification. Such a conformant element information item constitutes a policy expression [p.10].

A. The application/wspolicy+xml Media Type

This appendix defines the "application/wspolicy+xml" media type which can be used to describe Web Services Policy documents serialized as XML. Either wsp:Policy or wsp:PolicyAttachment could be the root element of such a document.

A.1 Registration

MIME media type name:

application

MIME subtype name:

wspolicy+xml

Required parameters:

none

Optional parameters:

charset

This parameter has identical semantics to the charset parameter of the "application/xml" media type as specified in [RFC 3023 [p.32]].

Encoding considerations:

Identical to those of "application/xml" as described in [RFC 3023 [p.32]], section 3.2, as applied to the Web Services Policy document Infoset.
Security considerations:


Interoperability considerations:

There are no known interoperability issues.

Published specifications:

This document and Web Services Policy Attachment [p.31].

Applications which use this media type:

This new media type is being registered to allow for deployment of Web Services Policy and references to Web Services Policy on the World Wide Web.

Additional information:

File extension:

wspolicy

Fragment identifiers:

A syntax identical to that of "application/xml" as described in RFC 3023 [p.32].

Base URI:

As specified in RFC 3023 [p.32], section 6.

Macintosh File Type code:

WSPOLICY

Person and email address to contact for further information:

World Wide Web Consortium <web-human@w3.org>

Intended usage:

COMMON

Author/Change controller:

The Web Services Policy 1.5 specification set is a work product of the World Wide Web Consortium’s Web Service Policy Working Group. The W3C has change control over these specifications.
B. References

B.1 Normative References

[Web Services Policy Attachment]


[IETF RFC 2119]


[IETF RFC 3987]


[WS-Security 2004]


/XML 1.0]


/XML ID]


/XML Information Set]


/XML Namespaces]


/XML Schema Structures]


[XML Schema Datatypes]

[RFC 3023]
IETF "RFC 3023: XML Media Types", M. Murata, S. St. Laurent, D. Kohn, July 1998. (See [RFC 3023].)

B.2 Other References

[C14N 1.0 Note]
Known Issues with Canonical XML 1.0 (C14N/1.0) J. Kahan and K. Lanz, Editors. World Wide Web Consortium, 17 August 2006. Available at http://www.w3.org/2006/04/c14n-note/c14n-note.html.

[SOAP 1.1]

[SOAP 1.2 Messaging Framework]

[UDDI API 2.0]

[UDDI Data Structure 2.0]

[UDDI 3.0]

[WS-SecurityPolicy]

[WSDL 1.1]
Web Services Description Language (WSDL) 1.1 E. Christensen, et al, Authors. World Wide Web...
C. Acknowledgements (Non-Normative)

This document is the work of the W3C Web Services Policy Working Group.

Members of the Working Group are (at the time of writing, and by alphabetical order): Dimitar Angelov (SAP AG), Abbie Barbir (Nortel Networks), Charlton Barreto (Adobe Systems Inc.), Sergey Beryozkin (IONA Technologies, Inc.), Vladislav Bezrukov (SAP AG), Toufic Boubez (Layer 7 Technologies), Paul Cotton (Microsoft Corporation), Jeffrey Crump (Sonic Software), Glen Daniels (Sonic Software), Jacques Durand (Fujitsu Limited), Ruchith Fernando (WSO2), Christopher Ferris (IBM Corporation), William Henry (IONA Technologies, Inc.), Frederick Hirsch (Nokia), Maryann Hondo (IBM Corporation), Tom Jordahl (Adobe Systems Inc.), Philippe Le Hégaret (W3C/MIT), Jong Lee (BEA Systems, Inc.), Mark Little (JBoss Inc.), Ashok Malhotra (Oracle Corporation), Monica Martin (Sun Microsystems, Inc.), Jeff Mischkinsky (Oracle Corporation), Dale Moberg (Cyclone Commerce, Inc.), Anthony Nadalin (IBM Corporation), David Orchard (BEA Systems, Inc.), Fabian Ritzmann (Sun Microsystems, Inc.), Daniel Roth (Microsoft Corporation), Tom Rutt (Fujitsu Limited), Sanka Samaranayake (WSO2), Felix Sasaki (W3C/Keio), Skip Snow (Citigroup), Yakov Sverdlov (Computer Associates), Mark Temple-Raston (Citigroup), Asir Vedamuthu (Microsoft Corporation), Sanjiva Weerawarana (WSO2), Ümit Yalçınalp (SAP AG), Prasad Yendluri (webMethods, Inc.).

Previous members of the Working Group were: Bijan Parsia (University of Manchester), Seumas Soltysik (IONA Technologies, Inc.)

The people who have contributed to discussions on public-ws-policy@w3.org are also gratefully acknowledged.
D. Changes in this Version of the Document (Non-Normative)

A list of substantive changes since the Working Draft dated 2 November, 2006 is below:

- Defined an XML Media Type to describe Web Service Policy documents serialized as XML in Appendix [A. The application/wspolicy+xml Media Type][p.29].

E. Web Services Policy 1.5 - Framework Change Log (Non-Normative)

<table>
<thead>
<tr>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20060712</td>
<td>ASV</td>
<td>Updated the list of editors. Completed action items [12] [16] and [20] from the Austin F2F.</td>
</tr>
<tr>
<td>20060718</td>
<td>DBO</td>
<td>Completed action items: RFC2606 for domain names [09] (note: PLH had already done but it didn’t show up in the change log)</td>
</tr>
<tr>
<td>20060726</td>
<td>ASV</td>
<td>Incorporated the XML namespace URI versioning policy adopted by the WG.</td>
</tr>
<tr>
<td>20060803</td>
<td>PY</td>
<td>Completed Issue: [3551] Misc updates throughout.</td>
</tr>
<tr>
<td>20060808</td>
<td>PY</td>
<td>Completed action item: [20] to highlight infoset terms uniformy.</td>
</tr>
<tr>
<td>20060808</td>
<td>DBO</td>
<td>Completed action items: [15] as early as possible in the doc, use the definition that are defined in the doc.</td>
</tr>
<tr>
<td>20060808</td>
<td>ASV</td>
<td>Implemented the resolution for issue [3543] and the resolution for issue Modify wording in Abstract for Framework Restored Section [2.2 Extensibility][p.5] that was accidentally dropped. Completed action item [17] from the Austin F2F.</td>
</tr>
<tr>
<td>20060809</td>
<td>ASV</td>
<td>Implemented the resolution for issue [3563]</td>
</tr>
<tr>
<td>20060811</td>
<td>DBO</td>
<td>Completed action items: [15] remove use if emph/ital terms. Framework: removed emph on conceptually replace and support; attachment: make merge a termdef</td>
</tr>
<tr>
<td>20060813</td>
<td>ASV</td>
<td>Added a new Section [D. Changes in this Version of the Document][p.34] that provides a list of substantive changes since the previous publication.</td>
</tr>
<tr>
<td>20060818</td>
<td>ASV</td>
<td>Implemented the resolution for issue [3560]</td>
</tr>
<tr>
<td>20060822</td>
<td>TIB</td>
<td>Completed action item: resolution for issue [3565]</td>
</tr>
<tr>
<td>20060824</td>
<td>PY</td>
<td>Completed action item: resolution for issue [3552]</td>
</tr>
<tr>
<td>20060827</td>
<td>TIB</td>
<td>Completed action item: resolution for adding Conformance section.</td>
</tr>
<tr>
<td>Date</td>
<td>Author</td>
<td>Action Item Completion</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>------------------------</td>
</tr>
<tr>
<td>20060828</td>
<td>DBO</td>
<td>Completed action item: Partial resolution for issue 3590 for adding document attribute extensibility of wsp:Policy/@{any} and wsp:Policy/.../wsp:PolicyReference/@{any}</td>
</tr>
<tr>
<td>20060829</td>
<td>ASV</td>
<td>Implemented the resolution for issue 3561: replaced URI with IRI.</td>
</tr>
<tr>
<td>20060830</td>
<td>DBO</td>
<td>Completed action item: resolution for issue 3604: Removing Goals section, resulted in moving Policy expression definition to 2nd para of intro.</td>
</tr>
<tr>
<td>20060906</td>
<td>TIB</td>
<td>Completed action item: resolution for issue 3607: Better describe policy language capabilities in the Introduction.</td>
</tr>
<tr>
<td>20060912</td>
<td>DBO</td>
<td>Completed action item: 6</td>
</tr>
<tr>
<td>20060913</td>
<td>TIB</td>
<td>Completed action item: 8</td>
</tr>
<tr>
<td>20060913</td>
<td>TIB</td>
<td>Completed action item: 31</td>
</tr>
<tr>
<td>20060913</td>
<td>TIB</td>
<td>Completed action item: 11</td>
</tr>
<tr>
<td>20060918</td>
<td>PY</td>
<td>Completed action item: 16</td>
</tr>
<tr>
<td>20060918</td>
<td>PY</td>
<td>Completed action item: 17</td>
</tr>
<tr>
<td>20060918</td>
<td>PY</td>
<td>Completed action item: 23 for issue 3617: Namespace URI versioning Policy is not clear.</td>
</tr>
<tr>
<td>20060918</td>
<td>PY</td>
<td>Completed action item: 33 for issue 3672: Clarify the policy model for Web Services.</td>
</tr>
<tr>
<td>20060918</td>
<td>PY</td>
<td>Completed action item: 34 for issue 3703: Element within policy expression must be an assertion.</td>
</tr>
<tr>
<td>20060918</td>
<td>PY</td>
<td>Completed action item: 39 for issue 3710: Clarify that policy assertion parameters are opaque to framework processing.</td>
</tr>
<tr>
<td>20060918</td>
<td>PY</td>
<td>Completed action item: 40 for issue 3711: Add Cross-Product description to 4.3.3 in Framework.</td>
</tr>
<tr>
<td>20060921</td>
<td>PY</td>
<td>Completed action item: 29 for issue 3577: Semantics of successful intersection determined by domain-specific assertion content.</td>
</tr>
<tr>
<td>20060924</td>
<td>TIB</td>
<td>Implemented the editorial action 35 to include the Security Considerations section from the Primer document.</td>
</tr>
<tr>
<td>20060926</td>
<td>ASV</td>
<td>Implemented the action item: resolution for issue 3549</td>
</tr>
<tr>
<td>Date</td>
<td>Name</td>
<td>Action Item</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>20060927</td>
<td>MH</td>
<td>02</td>
</tr>
<tr>
<td>20060927</td>
<td>PY</td>
<td>46</td>
</tr>
<tr>
<td>20061002</td>
<td>DBO</td>
<td>7</td>
</tr>
<tr>
<td>20061002</td>
<td>DBO</td>
<td>3712</td>
</tr>
<tr>
<td>20061004</td>
<td>PY</td>
<td>10</td>
</tr>
<tr>
<td>20061007</td>
<td>TIB</td>
<td>47</td>
</tr>
<tr>
<td>20061007</td>
<td>TIB</td>
<td>19</td>
</tr>
<tr>
<td>20061008</td>
<td>MH</td>
<td>45</td>
</tr>
<tr>
<td>20061011</td>
<td>PY</td>
<td>15</td>
</tr>
<tr>
<td>20061012</td>
<td>DBO</td>
<td>15</td>
</tr>
<tr>
<td>20061019</td>
<td>PY</td>
<td>57</td>
</tr>
<tr>
<td>20061027</td>
<td>ASV</td>
<td>3705</td>
</tr>
<tr>
<td>20061030</td>
<td>DBO</td>
<td>3712</td>
</tr>
<tr>
<td>20061102</td>
<td>ASV</td>
<td>7</td>
</tr>
<tr>
<td>20061102</td>
<td>ASV</td>
<td>3721</td>
</tr>
<tr>
<td>20061102</td>
<td>ASV</td>
<td>3789</td>
</tr>
<tr>
<td>20061102</td>
<td>ASV</td>
<td>3721</td>
</tr>
<tr>
<td>20061110</td>
<td>ASV</td>
<td>3789</td>
</tr>
<tr>
<td>20061109</td>
<td>ASV</td>
<td>3721</td>
</tr>
<tr>
<td>Date</td>
<td>ASV</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>20061114</td>
<td>ASV</td>
<td>Fixed typos in Appendix §A. The application/wspolicy+xml Media Type. [p.29]</td>
</tr>
</tbody>
</table>