Web Services Policy 1.5 - Framework

W3C Working Draft 27 September 2006

This version:
http://www.w3.org/TR/2006/WD-ws-policy-20060927
Latest version:
http://www.w3.org/TR/ws-policy
Previous versions:
http://www.w3.org/TR/2006/WD-ws-policy-20060731
Editors:
Asir S Vedamuthu, Microsoft Corporation
David Orchard, BEA Systems, Inc.
Maryann Hondo, IBM Corporation
Toufic Boubez, Layer 7 Technologies
Prasad Yendluri, webMethods, Inc.

This document is also available in these non-normative formats: PDF, PostScript, XML, and plain text.

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 an updated Public Working Draft of the Web Services Policy 1.5 - Framework specification for review by W3C members and other interested parties. It has been produced by the Web Services Policy Working Group, which is part of the W3C Web Services Activity. A list of changes in this version of the document [p.27] and a diff-marked version against the previous version of this document are available.

Discussion of this document takes place on the public public-ws-policy@w3.org mailing list (public archive) and within Bugzilla. 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

Appendices

---

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.

[Definition: A **policy** is a collection of policy alternatives [p.3] ,] where [Definition: a **policy alternative** is a collection of policy assertions [p.3] .] [Definition: A **policy assertion** represents an individual requirement, capability, or other property of a behavior.] [Definition: A **policy expression** is an XML Infoset representation of a policy [p.3] , 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 attachment [p.3] . [Definition: A **policy attachment** is a mechanism for associating policy [p.3] with one or more policy scopes [p.3] .] [Definition: A **policy scope** is a collection of policy subjects [p.3] to which a policy may apply.] [Definition: A **policy subject** is an entity (e.g., an endpoint, message, resource, interaction) with which a policy [p.3] 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.26]*] defines such mechanisms, especially for associating policy with arbitrary XML elements [*XML 1.0 [p.25]*], WSDL artifacts [*WSDL 1.1 [p.26]* , *WSDL 2.0 Core Language [p.26]*], and UDDI elements [*UDDI API 2.0 [p.26]* , *UDDI Data Structure 2.0 [p.26]* , *UDDI 3.0 [p.26]*].

1.1 Example

The following example illustrates a security policy expression [p.3] using assertions defined in WS-SecurityPolicy [*WS-SecurityPolicy [p.26]*]:

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)     <sp:Basic256Rsa15 />
(04)     <sp:TripleDesRsa15 />
(05)   </wsp:ExactlyOne>
(06) </wsp:Policy>
```

Lines (01-06) represent a policy for the algorithm suite required for performing cryptographic operations with symmetric or asymmetric key-based security tokens.

Lines (02-05) 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 uses one of the algorithm suite assertions (Lines 03-04) specified.

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:
  - "?" (0 or 1)
  - "*" (0 or more)
  - "+" (1 or more)
- 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.25]*]. Information items properties are indicated by the style **[infoset property]**.
- XML namespace prefixes (see Table 2-1 [p.5]) 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 other than the http://www.w3.org/2006/07/ws-policy 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.25]*] 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 SHOULD 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.5] . Note that the choice of any namespace prefix is arbitrary and not semantically significant (see [*XML Namespaces [p.25]*]).

Table 2-1. Prefixes and Namespaces used in this specification

Prefix	Namespace	Specification
`sp`	`http://schemas.xmlsoap.org/ws/2005/07/securitypolicy`	[*WS-SecurityPolicy [p.26]*]
`wsp`	`http://www.w3.org/2006/07/ws-policy`	This specification
`wsu`	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd`	[*WS-Security 2004 [p.25]*]
`xs`	`http://www.w3.org/2001/XMLSchema`	[*XML Schema Structures [p.25]*]

All information items defined by this specification are identified by the XML namespace URI [*XML Namespaces [p.25]*] `http://www.w3.org/2006/07/ws-policy`. A normative XML Schema [*XML Schema Structures [p.25]* , *XML Schema Datatypes [p.25]*] 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 preponderance of instance would remain valid.
- Modifications to the cardinality of elements 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.25]*].

We introduce the following terms that are used throughout this document:

policy [p.3]

> A **policy** is a collection of policy alternatives [p.3] ,

policy alternative [p.3]

> a **policy alternative** is a collection of policy assertions [p.3] .

policy assertion [p.3]

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

policy assertion parameter [p.8]

> A **policy assertion parameter** qualifies the behavior indicated by a policy assertion [p.3] .

policy assertion type [p.7]

> A **policy assertion type** represents a class of policy assertions [p.3] and implies a schema for the assertion and assertion-specific semantics.

policy attachment [p.3]

A **policy attachment** is a mechanism for associating policy [p.3] with one or more policy scopes [p.3] .

policy expression [p.3]

A **policy expression** is an XML Infoset representation of a policy [p.3] , either in a normal form or in an equivalent compact form.

policy scope [p.3]

A **policy scope** is a collection of policy subjects [p.3] to which a policy may apply.

policy subject [p.3]

A **policy subject** is an entity (e.g., an endpoint, message, resource, interaction) with which a policy [p.3] can be associated.

policy vocabulary [p.8]

A **policy vocabulary** is the set of all policy assertion types [p.7] 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

A policy assertion [p.3] identifies a behavior that is a requirement (or capability) of a policy subject [p.3] . Assertions indicate domain-specific (e.g., security, transactions) semantics and are expected to be defined in separate, domain-specific specifications.

Assertions are strongly typed by the domain authors that define them. The policy assertion type [p.7] 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. [Definition: A **policy assertion type** represents a class of policy assertions [p.3] and implies a schema for the assertion and assertion-specific semantics.] Assertions of a given type MUST be consistently interpreted independent of their policy subjects [p.3] .

Domain authors MAY define that an assertion contains a policy expression [p.3] as one of its **[children]**. Policy expression nesting is used by domain authors to further qualify one or more specific aspects of the original assertion. For example, security policy domain 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 [p.3] MAY contain a non-empty **[attributes]** property and/or a non-empty **[children]** property. Such content MAY be used to parameterize the behavior indicated by the assertion. [Definition: A **policy assertion parameter** qualifies the behavior indicated by a policy assertion [p.3] .] 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.

Domain authors should be cognizant of the processing requirements when defining complex assertions containing additional assertion content or nested policy expressions. Specifically, domain 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

A policy alternative [p.3] is a logical construct which represents a potentially empty collection of policy assertions [p.3] . An alternative with zero assertions indicates no behaviors. An alternative with one or more assertions indicates behaviors implied by those, and only those assertions.

The vocabulary of a policy alternative is the set of all policy assertion types [p.7] within the alternative. [Definition: A **policy vocabulary** is the set of all policy assertion types [p.7] used in a policy.] An assertion whose type is part of the policy's vocabulary but is not included in an alternative is explicitly prohibited by the alternative.

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.3] are beyond the scope of this specification.

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) content, if any) are specific to the assertion type and are outside the scope of this document.

3.3 Policy

At the abstract level a policy [p.3] is a potentially empty collection of policy alternatives [p.3] . 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 Web services

Applied in the Web services model, policy [p.3] is used to convey conditions on an interaction between a Web service requestor and a Web service provider. Satisfying assertions in the policy usually results in behavior that reflects these conditions. Typically, the provider of a Web service exposes a policy to convey conditions under which it provides the service. A requester might use this policy 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 [p.3] is supported by a requester if and only if the requester satisfies the requirement (or accommodates the capability) corresponding to the assertion. A policy alternative [p.3] is supported by a requester if and only if the requester supports all the assertions in the alternative. And, a policy [p.3] is supported by a requester if and only if the requester 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 a requester may be able to support a policy even if the requester does not understand the type [p.7] of each assertion in the vocabulary of the policy [p.8] ; the requester only has to understand the type of each assertion in the vocabulary of a policy alternative [p.3] the requester 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 requesters to continue to use old alternatives in a backward-compatible manner.

4. Policy Expression

To convey policy in an interoperable form, a policy expression [p.3] is an XML Infoset representation of a policy. The normal form policy expression is the most straightforward Infoset; equivalent, alternative Infosets allow compactly expressing a policy through a number of constructs.

4.1 Normal Form Policy Expression

To facilitate interoperability, this specification defines a normal form for policy expressions [p.3] 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:

```
<wsp:Policy … >
  <wsp:ExactlyOne>
    ( <wsp:All> ( <Assertion …> … </Assertion> )* </wsp:All> )*
  </wsp:ExactlyOne>
</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 [p.3] in the normal form of a policy expression contains a nested policy expression, the nested policy expression MUST contain at most one policy alternative.

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 the policy expression example introduced earlier.

```
(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:Basic256Rsa15 />
(05)     </wsp:All>
(06)     <wsp:All>
(07)       <sp:TripleDesRsa15 />
(08)     </wsp:All>
(09)   </wsp:ExactlyOne>
(10) </wsp:Policy>
```

Lines (03-05) and Lines (06-08) express the two alternatives in the policy. If the first alternative is selected, only the Basic 256 RSA 15 algorithm suite [*WS-SecurityPolicy [p.26]*] is supported; conversely, if the second alternative is selected, only the 3DES RSA 15 algorithm suite is supported.

4.2 Policy Identification

A policy expression [p.3] MAY be associated with an IRI [*IETF RFC 3987 [p.25]*]. 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 [*IETF RFC 3987 [p.25]*]. 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 [p.3] mechanism such as those defined in WS-PolicyAttachment [*Web Services Policy Attachment [p.26]*].

`/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. To refer to this policy expression, an IRI-reference MAY be formed using this value per Section 4.2 of WS-Security [*WS-Security 2004 [p.25]*] 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.25]*] and thus this combination must be avoided (see [*C14N 1.0 Note [p.26]*]). 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 [p.3] , semantics for recursively nested policy operators, and a policy reference/inclusion mechanism. Each is described in the subsections below.

To interpret a compact policy 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 **4.3.4 Policy Inclusion** [p.20] .

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.

    2. Otherwise the Element Information Item C is an assertion; normalize C per Sections **4.3.1 Optional Policy Assertions** [p.12] and **4.3.2 Policy Assertion Nesting** [p.13] .

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.16] .

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.3] is optional, this specification defines an attribute that is a syntactic shortcut for expressing policy alternatives with and without the assertion. The schema outline for this attribute is as follows:

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

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

`/Assertion/@wsp:Optional`

If true, the expression of the assertion is semantically equivalent to the following:

```
<wsp:ExactlyOne>
  <wsp:All> <Assertion …> … </Assertion> </wsp:All>
  <wsp:All />
</wsp:ExactlyOne>
```

If false, the expression of the assertion is semantically equivalent to the following:

```
<wsp:ExactlyOne>
  <wsp:All> <Assertion …> … </Assertion> </wsp:All>
</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.3] MAY contain a nested policy expression [p.3] . The schema outline for a nested policy expression is:

```
<Assertion …>
  …
  ( <wsp:Policy …> … </wsp:Policy> )?
  …
</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.16] , this is equivalent to a nested policy expression with a single alternative that has zero assertions. If this is not done then two assertions of the same type will not be compatible and intersection may fail (see Section **4.4 Policy Intersection** [p.22]).

/Assertion/*/wsp:Policy

> This specification does not define processing for arbitrary `wsp:Policy` Element Information Items in the descendants of an assertion, 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 compact nested policy expression:

```
<wsp:Policy
   xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
   xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
  <sp:TransportBinding>
    <wsp:Policy>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <wsp:ExactlyOne>
            <sp:Basic256Rsa15 />
            <sp:TripleDesRsa15 />
          </wsp:ExactlyOne>
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:TransportToken>
        <wsp:Policy>
          <sp:HttpsToken RequireClientCertificate="false" />
        </wsp:Policy>
      </sp:TransportToken>
```

```
           <!-- Details omitted for readability -->
(17)      </wsp:Policy>
(18)    </sp:TransportBinding>
(19) </wsp:Policy>
```

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 this policy is equivalent to the following:

```
<wsp:Policy
   xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
   xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic256Rsa15 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken RequireClientCertificate="false" />
            </wsp:Policy>
          </sp:TransportToken>
          <!-- Details omitted for readability -->
        </wsp:Policy>
      </sp:TransportBinding>
    </wsp:All>
    <wsp:All>
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:TripleDesRsa15 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken RequireClientCertificate="false" />
            </wsp:Policy>
          </sp:TransportToken>
          <!-- Details omitted for readability -->
        </wsp:Policy>
      </sp:TransportBinding>
    </wsp:All>
  </wsp:ExactlyOne>
</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

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

`wsp:Policy` is equivalent to `wsp:All` .

Empty

- `<wsp:All />` expresses a policy with zero policy assertions. Note that since `wsp:Policy` is equivalent to `wsp:All` , `<wsp:Policy />` is therefore equivalent to `<wsp:All />`, i.e., a policy alternative with zero assertions.
- `<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 **3.2 Policy Alternative** [p.8] and **3.3 Policy** [p.8] , respectively), `wsp:All` and `wsp:ExactlyOne` are commutative. For example,

```
<wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
```

is equivalent to:

```
<wsp:All> <!-- assertion 2 --> <!-- assertion 1 --> </wsp:All>
```

and:

```
<wsp:ExactlyOne>
  <!-- assertion 1 --> <!-- assertion 2 -->
</wsp:ExactlyOne>
```

is equivalent to:

```
<wsp:ExactlyOne>
  <!-- assertion 2 --> <!-- assertion 1 -->
</wsp:ExactlyOne>
```

Associative

`wsp:All` and `wsp:ExactlyOne` are associative. For example,

```
<wsp:All>
  <!-- assertion 1 -->
  <wsp:All> <!-- assertion 2 --> </wsp:All>
</wsp:All>
```

is equivalent to:

```
<wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
```

and:

```
<wsp:ExactlyOne>
  <!-- assertion 1 -->
  <wsp:ExactlyOne> <!-- assertion 2 --> </wsp:ExactlyOne>
</wsp:ExactlyOne>
```

is equivalent to:

```
<wsp:ExactlyOne>
  <!-- assertion 1 --> <!-- assertion 2 -->
</wsp:ExactlyOne>
```

Idempotent

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

```
<wsp:All>
  <wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
</wsp:All>
```

is equivalent to:

```
<wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
```

and:

```
<wsp:ExactlyOne>
  <wsp:ExactlyOne>
    <!-- assertion 1 --> <!-- assertion 2 -->
  </wsp:ExactlyOne>
</wsp:ExactlyOne>
```

is equivalent to:

```
<wsp:ExactlyOne>
  <!-- assertion 1 --> <!-- assertion 2 -->
</wsp:ExactlyOne>
```

Distributive

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

```
<wsp:All>
  <wsp:ExactlyOne>
    <!-- assertion 1 -->
    <!-- assertion 2 -->
  </wsp:ExactlyOne>
</wsp:All>
```

is equivalent to:

```
<wsp:ExactlyOne>
  <wsp:All>
    <!-- assertion 1 -->
  </wsp:All>
  <wsp:All>
    <!-- assertion 2 -->
  </wsp:All>
</wsp:ExactlyOne>
```

Similarly,

```
<wsp:All>
  <wsp:ExactlyOne>
    <!-- assertion 1 -->
    <!-- assertion 2 -->
  </wsp:ExactlyOne>
  <wsp:ExactlyOne>
    <!-- assertion 3 -->
    <!-- assertion 4 -->
  </wsp:ExactlyOne>
</wsp:All>
```

is equivalent to:

```
<wsp:ExactlyOne>
  <wsp:All><!-- assertion 1 --><!-- assertion 3 --></wsp:All>
  <wsp:All><!-- assertion 1 --><!-- assertion 4 --></wsp:All>
  <wsp:All><!-- assertion 2 --><!-- assertion 3 --></wsp:All>
  <wsp:All><!-- assertion 2 --><!-- assertion 4 --></wsp:All>
</wsp:ExactlyOne>
```

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

```
<wsp:All>
  <wsp:ExactlyOne>
    <!-- assertion 1 -->
    <!-- assertion 2 -->
  </wsp:ExactlyOne>
  <wsp:ExactlyOne />
</wsp:All>
```

is equivalent to:

```
<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.12] to `@wsp:Optional` in Line (02), and distributing `wsp:All` over `wsp:ExactlyOne` per Section **4.3.3 Policy Operators** [p.16] 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)       <sp:WssUsernameToken11 />
(14)     </wsp:All>
(15)   </wsp:ExactlyOne>
(16) </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)       <sp:WssUsernameToken10 />
(13)     </wsp:All>
(14)     <wsp:All>
(15)       <sp:WssUsernameToken11 />
(16)     </wsp:All>
(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 Inclusion

In order to share assertions [p.3] across policy expressions [p.3] , 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.)

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

```
<wsp:Policy>
  …
  <wsp:PolicyReference
      URI="xs:anyURI"
    ( Digest="xs:base64Binary" ( DigestAlgorithm="xs:anyURI" )? )?
      … />
  …
</wsp:Policy>
```

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

`/wsp:Policy/…/wsp:PolicyReference`

This element references a policy expression that is being included.

`/wsp:Policy/…/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** [p.11]) 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:Policy/…/wsp:PolicyReference/@Digest`

This optional attribute 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:Policy/…/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.

URI	Description
`http://www.w3.org/2006/07/ws-policy/Sha1Exc` (implied)	The digest is a SHA1 hash over the octet stream resulting from using the Exclusive XML canonicalization defined for XML Signature [*XML-Signature [p.27]*].

`/wsp:Policy/…/wsp:PolicyReference/@{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.

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.

```
(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)   <sp:IncludeTimestamp />
(03)   <wsp:PolicyReference URI="#Protection" />
(04)   <sp:OnlySignEntireHeadersAndBody />
(05) </wsp:Policy>
```

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 Policy Intersection

Policy intersection is useful when two or more parties express policy [p.3] and want to limit the policy alternatives [p.3] 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, associative function that takes two policies and returns a policy.

Because the set of behaviors indicated by a policy alternative [p.3] depends on the domain-specific semantics of the collected assertions, determining whether two policy alternatives are compatible generally involves domain-specific processing. As a first approximation, an algorithm is defined herein that approximates compatibility in a domain-independent manner; specifically, for two policy alternatives to be compatible, they must at least have the same vocabulary (see Section **3.2 Policy Alternative** [p.8]).

- Two policy assertions [p.3] are compatible if they have the same type [p.7] and
- If either assertion contains a nested policy expression [p.3] , 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 [p.8] are not part of the compatibility determination defined herein but may be part of other, domain-specific compatibility processing.

- Two policy alternatives [p.3] are compatible if each assertion in one is compatible with an assertion in the other, and vice-versa. If two alternatives are compatible, their intersection is an alternative containing all of the assertions in both alternatives.
- Two policies [p.3] 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:

```
(01) <wsp:Policy
        xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
        xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
       <!-- Policy P1 -->
(02)   <wsp:ExactlyOne>
```

```
(03)     <wsp:All> <!-- Alternative A1 -->
(04)       <sp:SignedElements>
(05)         <sp:XPath>/S:Envelope/S:Body</sp:XPath>
(06)       </sp:SignedElements>
(07)       <sp:EncryptedElements>
(08)         <sp:XPath>/S:Envelope/S:Body</sp:XPath>
(09)       </sp:EncryptedElements>
(10)     </wsp:All>
(11)     <wsp:All> <!-- Alternative A2 -->
(12)       <sp:SignedParts>
(13)         <sp:Body />
(14)         <sp:Header
                Namespace="http://www.w3.org/2005/08/addressing" />
(15)       </sp:SignedParts>
(16)       <sp:EncryptedParts>
(17)         <sp:Body />
(18)       </sp:EncryptedParts>
(19)     </wsp:All>
(20)   </wsp:ExactlyOne>
(21) </wsp: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.

```
(01) <wsp:Policy
        xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
        xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
       <!-- Policy P2 -->
(02)   <wsp:ExactlyOne>
(03)     <wsp:All> <!-- Alternative A3 -->
(04)       <sp:SignedParts />
(05)       <sp:EncryptedParts>
(06)         <sp:Body />
(07)       </sp:EncryptedParts>
(08)     </wsp:All>
(09)     <wsp:All> <!-- Alternative A4 -->
(10)       <sp:SignedElements>
(11)         <sp:XPath>/S:Envelope/S:Body</sp:XPath>
(12)       </sp:SignedElements>
(13)     </wsp:All>
(14)   </wsp:ExactlyOne>
(15) </wsp:Policy>
```

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.

```
<wsp:Policy
   xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
   xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
  <!-- Intersection of P1 and P2 -->
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SignedParts >
        <sp:Body />
        <sp:Header
           Namespace="http://www.w3.org/2005/08/addressing" />
      </sp:SignedParts>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
      <sp:SignedParts />
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

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 `sp:SignedParts` assertion. To leverage intersection, assertion authors are encouraged to factor assertions such that two assertions of the same assertion type are always (or at least typically) compatible.

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 1.1 [p.26]* , *SOAP 1.2 Messaging Framework [p.26]*] using WS-Security [*WS-Security 2004 [p.25]*] or embedded within other objects using object-specific security mechanisms.

6. Conformance

A. References

A.1 Normative References

[IETF RFC 2119]
Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, Author. Internet Engineering Task Force, June 1999. Available at http://www.ietf.org/rfc/rfc2119.txt.

[IETF RFC 3987]
Internationalized Resource Identifiers (IRIs) , M. Duerst and M. Suignard, Authors. Internet Engineering Task Force, January 2005. Available at http://www.ietf.org/rfc/rfc3987.txt.

[WS-Security 2004]
Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), A. Nadalin, C. Kaler, P. Hallam-Baker, and R. Monzillo, Editors. Organization for the Advancement of Structured Information Standards, March 2004. Available at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

[XML 1.0]
Extensible Markup Language (XML) 1.0 (Fourth Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, Editors. World Wide Web Consortium, 10 February 1998, revised 16 August 2006. This version of the XML 1.0 Recommendation is http://www.w3.org/TR/2006/REC-xml-20060816/. The latest version of XML 1.0 is available at http://www.w3.org/TR/REC-xml/.

[XML ID]
xml:id Version 1.0, J. Marsh, D. Veillard and N. Walsh, Editors. World Wide Web Consortium, 9 September 2005. This version of the xml:id Version 1.0 Recommendation is http://www.w3.org/TR/2005/REC-xml-id-20050909/. The latest version of xml:id Version 1.0 is available at http://www.w3.org/TR/xml-id/.

[XML Information Set]
XML Information Set (Second Edition), J. Cowan and R. Tobin, Editors. World Wide Web Consortium, 24 October 2001, revised 4 February 2004. This version of the XML Information Set Recommendation is http://www.w3.org/TR/2004/REC-xml-infoset-20040204. The latest version of XML Information Set is available at http://www.w3.org/TR/xml-infoset.

[XML Namespaces]
Namespaces in XML 1.0, T. Bray, D. Hollander, A. Layman, and R. Tobin, Editors. World Wide Web Consortium, 14 January 1999, revised 16 August 2006. This version of the XML Information Set Recommendation is http://www.w3.org/TR/2006/REC-xml-names-20060816/. The latest version of Namespaces in XML is available at http://www.w3.org/TR/REC-xml-names.

[XML Schema Structures]
XML Schema Part 1: Structures Second Edition, H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, Editors. World Wide Web Consortium, 2 May 2001, revised 28 October 2004. This version of the XML Schema Part 1 Recommendation is http://www.w3.org/TR/2004/REC-xmlschema-1-20041028. The latest version of XML Schema Part 1 is available at http://www.w3.org/TR/xmlschema-1.

[XML Schema Datatypes]
XML Schema Part 2: Datatypes Second Edition, P. Byron and A. Malhotra, Editors. World Wide Web Consortium, 2 May 2001, revised 28 October 2004. This version of the XML Schema Part 2 Recommendation is http://www.w3.org/TR/2004/REC-xmlschema-2-20041028. The latest version of XML Schema Part 2 is available at http://www.w3.org/TR/xmlschema-2.

A.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]
Simple Object Access Protocol (SOAP) 1.1, D. Box, et al, Editors. World Wide Web Consortium, 8 May 2000. Available at http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[SOAP 1.2 Messaging Framework]
SOAP Version 1.2 Part 1: Messaging Framework, M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. Frystyk Nielsen, Editors. World Wide Web Consortium, 24 June 2003. This version of the SOAP Version 1.2 Part 1: Messaging Framework Recommendation is http://www.w3.org/TR/2003/REC-soap12-part1-20030624/. The latest version of SOAP Version 1.2 Part 1: Messaging Framework is available at http://www.w3.org/TR/soap12-part1/.

[UDDI API 2.0]
UDDI Version 2.04 API, T. Bellwood, Editor. Organization for the Advancement of Structured Information Standards, 19 July 2002. This version of UDDI Version 2.0 API is http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm. The latest version of the UDDI 2.0 API is available at http://uddi.org/pubs/ProgrammersAPI_v2.htm.

[UDDI Data Structure 2.0]
UDDI Version 2.03 Data Structure Reference, C. von Riegen, Editor. Organization for the Advancement of Structured Information Standards, 19 July 2002. This version of UDDI Version 2.0 Data Structures is http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm. The latest version of the UDDI 2.0 Data Structures is available at http://uddi.org/pubs/DataStructure_v2.htm.

[UDDI 3.0]
UDDI Version 3.0.1, L. Clément, et al, Editors. Organization for the Advancement of Structured Information Standards, 14 October 2003. This version of the UDDI Version 3.0 is http://uddi.org/pubs/uddi-v3.0.1-20031014.htm. The latest version of the UDDI 3.0 specification is available at http://uddi.org/pubs/uddi_v3.htm.

[Web Services Policy Attachment]
Web Services Policy 1.5 - Attachment, A. S. Vedamuthu, D. Orchard, M. Hondo, T. Boubez and P. Yendluri, Editors. World Wide Web Consortium, 27, September 2006. This version of the specification of the Web Services Policy 1.5 - Attachment specification is http://www.w3.org/TR/2006/WD-ws-policy-attach-20060927. The latest version of Web Services Policy 1.5 - Attachment is available at http://www.w3.org/TR/ws-policy-attach.

[WS-SecurityPolicy]
WS-SecurityPolicy v1.0, A. Nadalin, M. Gudgin, A. Barbir, and H. Granqvist, Editors. Organization for the Advancement of Structured Information Standards, 8 December 2005. Available at http://www.oasis-open.org/committees/download.php/15979/oasis-wssx-ws-securitypolicy-1.0.pdf.

[WSDL 1.1]
Web Services Description Language (WSDL) 1.1, E. Christensen, et al, Authors. World Wide Web Consortium, March 2001. Available at http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[WSDL 2.0 Core Language]
Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, R. Chinnici, J. J. Moreau, A. Ryman, S. Weerawarana, Editors. World Wide Web Consortium, 27 March 2006. This version of the WSDL 2.0 specification is http://www.w3.org/TR/2006/CR-wsdl20-20060327. The

latest version of WSDL 2.0 is available at http://www.w3.org/TR/wsdl20.

[XML-Signature]
XML-Signature Syntax and Processing, D. Eastlake, J. Reagle, and D. Solo, Editors. The Internet Society & World Wide Web Consortium, 12 February 2002. This version of the XML-Signature Syntax and Processing Recommendation is http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/. The latest version of XML-Signature Syntax and Processing is available at http://www.w3.org/TR/xmldsig-core/.

B. 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), 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.), Bijan Parsia (University of Manchester), Fabian Ritzmann (Sun Microsystems, Inc.), Daniel Roth (Microsoft Corporation), 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 Yalçinalp (SAP AG), Prasad Yendluri (webMethods, Inc.).

The people who have contributed to discussions on public-ws-policy@w3.org are also gratefully acknowledged.

C. Changes in this Version of the Document (Non-Normative)

A list of substantive changes since the Working Draft dated 31 July 2006 is below:

- Added support for the `xml:id` attribute.
- Added an empty conformance section.
- Replaced URI with IRI.

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

Date	Author	Description

20060712	ASV	Updated the list of editors. Completed action items 12, 16 and 20 from the Austin F2F.
20060718	DBO	Completed action items: RFC2606 for domain names 09 (note: PLH had already done but it didn't show up in the change log)
20060726	ASV	Incorporated the XML namespace URI versioning policy adopted by the WG.
20060803	PY	Completed Issue: 3551 Misc updates throughout.
20060808	PY	Completed action item: 20 to highlight infoset terms uniformly.
20060808	DBO	Completed action items: 15 as early as possible in the doc, use the definition that are defined in the doc.
20060808	ASV	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.
20060809	ASV	Implemented the resolution for issue 3563.
20060811	DBO	Completed action items: 15 remove use if emph/ital terms. Framework: removed emph on conceptually replace and support; attachment: make merge a termdef
20060813	ASV	Added a new Section **C. Changes in this Version of the Document** [p.27] (that provides a list of substantive chanages since the previous publication).
20060818	ASV	Implemented the resolution for issue 3560.
20060822	TIB	Completed action item: resolution for issue 3565.
20060824	PY	Completed action item: resolution for issue 3552.
20060827	TIB	Completed action item: resolution for adding Conformance section.
20060828	DBO	Completed action item: Partial resolution for issue 3590. for adding document attribute extensbility of wsp:Policy/@{any} and wsp:Policy/.../wsp:PolicyReference/@{any}
20060829	ASV	Implemented the resolution for issue 3561: replaced URI with IRI.
20060830	DBO	Completed action item: resolution for issue 3604. Removing Goals section, resulted in moving Policy expression definition to 2nd para of intro.
200609006	DBO	Completed partial resolution for issue 3590. for adding document attribute extensbility of wsp:Policy/@{any} and wsp:Policy/.../wsp:PolicyReference/@{any}, specifically making attribute extensibility for any namespace.
20060906	TIB	Completed action item: resolution for issue 3607. Better describe policy language capabilities in the Introduction.