Web Services Policy 1.5 - Framework

W3C Recommendation 04 September 2007

This version:
    http://www.w3.org/TR/2007/REC-ws-policy-20070904
Latest version:
    http://www.w3.org/TR/ws-policy
Previous version:
    http://www.w3.org/TR/2007/PR-ws-policy-20070706/

Editors:
    Asir S Vedamuthu, Microsoft Corporation
    David Orchard, BEA Systems, Inc.
    Frederick Hirsch, Nokia
    Maryann Hondo, IBM Corporation
    Prasad Yendluri, webMethods (A subsidiary of Software AG)
    Toufic Boubez, Layer 7 Technologies
    Ümit Yalçinalp, SAP AG.

Please refer to the **errata** for this document, which may include some normative corrections.

See also **translations**.

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 the W3C Recommendation of the Web Services Policy 1.5 - Framework specification. It has been produced by the Web Services Policy Working Group, which is part of the W3C Web Services Activity.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

The Working Group released a test suite along with an implementation report. A diff-marked version against the previous version of this document is available.

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.

Table of Contents

---

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.10] is a collection of policy alternatives. A policy alternative [p.9] is a collection of policy assertions. A policy assertion [p.8] represents a requirement, capability, or other property of a behavior. A policy expression [p.11] is an XML Infoset representation of its policy, either in a normal form or in its equivalent compact form. Some policy assertions specify traditional requirements and capabilities that will manifest themselves in the messages exchanged(e.g., authentication scheme, transport protocol selection). Other policy assertions have no wire manifestation in the messages exchanged, yet are relevant to 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 cover discovery of policy, policy scopes and subjects, or their respective attachment mechanisms. A policy attachment [p.11] is a mechanism for associating policy with one or more policy scopes. A policy scope [p.13] is a collection of policy subjects to which a policy applies. A policy subject [p.8] is an entity (e.g., an endpoint, message, resource, interaction) with which a policy can be associated. Web Services Policy 1.5 - Attachment [*Web Services Policy Attachment [p.38]*] defines such policy attachment mechanisms, especially for associating policy with arbitrary XML elements [*XML 1.0 [p.38]*], WSDL artifacts [*WSDL 1.1 [p.40]* , *WSDL 2.0 Core Language [p.40]*], and UDDI elements [*UDDI API 2.0 [p.39]* , *UDDI Data Structure 2.0 [p.40]* , *UDDI 3.0 [p.40]*]. Other specifications are free to define either extensions to the mechanisms defined in Web Services Policy 1.5 - Attachment [*Web Services Policy Attachment [p.38]*], or additional mechanisms not covered by Web Services Policy 1.5 - Attachment [*Web Services Policy Attachment [p.38]*], for purposes of associating policy with policy scopes and subjects.

1.1 Example

Example 1-1 [p.4] illustrates a security policy expression [p.11] using assertions defined in WS-SecurityPolicy [*WS-SecurityPolicy [p.40]*]:

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

```
(01) <wsp:Policy
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/ws-policy" >
(02)   <wsp:ExactlyOne>
(03)     <wsp:All>
(04)        <sp:SignedParts>
(05)           <sp:Body/>
(06)        </sp:SignedParts>
(07)     </wsp:All>
(08)     <wsp:All>
(09)        <sp:EncryptedParts>
(10)           <sp:Body/>
(11)        </sp:EncryptedParts>
(12)     </wsp:All>
(13)   </wsp:ExactlyOne>
(14) </wsp:Policy>
```

Lines (03-07) represent one policy alternative for signing a message body.

Lines (08-12) 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:
  - "?" (0 or 1)
  - "*" (0 or more)
  - "+" (1 or more)
- The character "|" is used to indicate an exclusive 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.38]*]. 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, unless specified otherwise such as in Section **4.3.3 Policy Operators** [p.18] .
- 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.39]*] descriptions.

2.2 Extensibility

Within normative outlines, in this specification, 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. In this context, 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 a policy assertion, unless specified otherwise such as in Section **4.3.4 Policy References** [p.26] .

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.39]*]).

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

Prefix	Namespace	Specification
`sp`	`http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702`	[*WS-SecurityPolicy [p.40]*]
`wsam`	`http://www.w3.org/2007/05/addressing/metadata`	[*WS-Addressing Metadata [p.40]*]
`wsp`	`http://www.w3.org/ns/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.38]*]
`xs`	`http://www.w3.org/2001/XMLSchema`	[*XML Schema Structures [p.39]*]

All information items defined by this specification are identified by the XML namespace URI [*XML Namespaces [p.39]*] `http://www.w3.org/ns/ws-policy`. A normative XML Schema [*XML Schema Structures [p.39] , XML Schema Datatypes [p.39]*] document can be obtained indirectly by dereferencing the namespace document at the WS-Policy 1.5 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 as the specifications transition through Candidate Recommendation, Proposed Recommendation and Recommendation status. However, should the specifications revert to Working Draft status, and 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, the namespace URI will be changed accordingly.

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.38]*].

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

collection [p.9]

> The items in a **collection** in this specification are unordered and may contain duplicates.

ignorable policy assertion [p.8]

> An **ignorable policy assertion** is an assertion that may be ignored for purposes of determining the compatibility of alternatives in policy intersection in a lax mode (as defined in 4.5 Policy Intersection).

nested policy expression [p.15]

> A **nested policy expression** is a policy expression [p.11] that is an Element Information Item in the **[children]** property of a policy assertion [p.8] .

policy [p.10]

> A **policy** is a potentially empty collection of policy alternatives [p.9] .

policy alternative [p.9]

> A **policy alternative** is a potentially empty collection [p.9] of policy assertions [p.8] .

policy assertion [p.8]

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

policy assertion parameter [p.9]

A **policy assertion parameter** qualifies the behavior indicated by a policy assertion [p.8] .

policy assertion type [p.9]

A **policy assertion type** represents a class of policy assertions [p.8] and implies a schema for the assertion and assertion-specific semantics.

policy attachment [p.13]

A **policy attachment** is a mechanism for associating policy [p.10] with one or more policy scopes [p.13] .

policy expression [p.11]

A **policy expression** is an XML Infoset representation of a policy [p.10] , either in a normal form or in an equivalent compact form.

policy scope [p.13]

A **policy scope** is a collection of policy subjects [p.8] to which a policy may apply.

policy subject [p.8]

A **policy subject** is an entity (e.g., an endpoint, message, resource, operation) with which a policy [p.10] can be associated.

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 a requirement, a capability, or other property of a behavior.] A policy assertion [p.8] identifies a behavior that is a requirement or capability of a policy subject [p.8] . [Definition: A **policy subject** is an entity (e.g., an endpoint, message, resource, operation) with which a policy [p.10] 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.29]). [Definition: An **ignorable policy assertion** is an assertion that may be ignored for purposes of determining the compatibility of alternatives in policy intersection in a lax mode (as defined in 4.5 Policy Intersection).] 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 [p.8] and implies a schema for the assertion and assertion-specific semantics.] The policy assertion type [p.9] 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 [p.8] .

Authors MAY define that an assertion contains a policy expression [p.11] (as defined in **4. Policy Expression** [p.11]) as one of its **[children]**. Nested policy expression(s) [p.15] are used by authors to further qualify one or more specific aspects of the parent policy assertion. The qualification may indicate a relationship or context between the parent policy assertion and a nested policy expression. For example within a security domain, security policy authors may define an assertion describing a set of security algorithms to qualify the specific behavior of a security binding assertion. A parent policy assertion of one domain may also serve as a container for the nested policy expression from another domain.

The XML Infoset of a policy assertion [p.8] MAY contain a non-empty **[attributes]** property and/or a non-empty **[children]** property. Such properties, excluding the Attribute and Element Information Items from the WS-Policy language XML namespace name are policy assertion parameters [p.9] and 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.8] .] 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.9] 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 [p.9] of policy assertions [p.8] .] [Definition: The items in a **collection** in this specification are unordered and may contain duplicates.] An alternative with zero assertions indicates no behaviors. An alternative with one or more assertions indicates behaviors implied by those, and only those assertions.

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 behaviors 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.39]*]) content, if any) are specific to the assertion type and are outside the scope of this document. If policy assertion authors did not specify the semantics of repetition of policy assertions [p.8] of a type [p.9] that allows neither parameters [p.9] nor nested policy expressions [p.15] within a policy alternative [p.9] , then repetition is simply redundancy, and multiple assertions [p.8] of the assertion type [p.9] within a policy alternative [p.9] have the same meaning as a single assertion [p.8] of the type [p.9] within the policy alternative [p.9] .

Note: Depending on the semantics of the domain specific policy assertions regardless if they are qualified by nested policy expressions, a combination of the policy assertions can be required to specify a particular behavior. For example, a combination of two or three assertions from the WS-SecurityPolicy [*WS-SecurityPolicy [p.40]*] specification is used to indicate message-level security for protecting messages - that is, the `sp:AsymmetricBinding` assertion is used to indicate message-level security, the `sp:SignedParts` assertion is used to indicate the parts of a message to be protected and the `sp:EncryptedParts` assertion is used to indicate the parts of a message that require confidentiality.

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 [p.10] 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 assertion [p.8] authors to define the interaction scope of an assertion including any constraints on the policy subjects [p.8] 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 [p.8] 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 [p.9] is supported by an entity if and only if the entity supports all the assertions in the alternative. And, a policy [p.10] 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 policy; the entity only has to understand the type of each assertion in a policy alternative that 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 [p.10] in an interoperable form, using the XML Infoset representation of a policy. [Definition: A **policy expression** is an XML Infoset representation of a policy [p.10] , either in a normal form or in an equivalent compact form.]

The normal form (see Section **4.1 Normal Form Policy Expression** [p.11]) of a policy expression is the most straightforward XML Infoset representation of the policy data model. Equivalent, alternative representations allow policy authors to compactly express a policy (see Section **4.3 Compact Policy Expression** [p.14]). Policy authors might be more interested in the compact form (see Section **4.3 Compact Policy Expression** [p.14]), where the outlines and definitions describe what is valid with regards to the policy language XML Schema.

While the policy language XML Schema is a representation of the compact form, the normal form is more restrictive as outlined in Section **4.1 Normal Form Policy Expression** [p.11] .

4.1 Normal Form Policy Expression

To facilitate interoperability, this specification defines a normal form for policy expressions [p.11] that is a straightforward XML Infoset representation of a policy, enumerating each of its alternatives [p.9] that in turn enumerate each of their assertions [p.8] . 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.8] in the normal form of a policy expression contains a nested policy expression [p.15] , the nested policy expression MUST contain at most one policy alternative (see **4.3.2 Policy Assertion Nesting** [p.15]).

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.

```
(01) <wsp:Policy
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/ws-policy" >
(02)   <wsp:ExactlyOne>
(03)     <wsp:All>
(04)        <sp:SignedParts>
(05)           <sp:Body/>
(06)        </sp:SignedParts>
(07)     </wsp:All>
(08)     <wsp:All>
(09)        <sp:EncryptedParts>
(10)           <sp:Body/>
(11)        </sp:EncryptedParts>
(12)     </wsp:All>
(13)   </wsp:ExactlyOne>
(14) </wsp:Policy>
```

Lines (03-07) and Lines (08-12) express the two alternatives in the policy. If the first alternative is selected, the message body needs to be signed [*WS-SecurityPolicy [p.40]*] is supported; conversely, if the second alternative is selected, the message body needs to be encrypted.

4.2 Policy Identification

A policy expression [p.11] MAY be associated with an IRI [*IETF RFC 3987 [p.38]*]. 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.38]*]. 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.13] mechanism such as those defined in WS-PolicyAttachment [*Web Services Policy Attachment [p.38]*]. [Definition: A **policy attachment** is a mechanism for associating policy [p.10] with one or more policy scopes [p.13] .] [Definition: A **policy scope** is a collection of policy subjects [p.8] 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 [*XML 1.0 [p.38]*] 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 [*WS-Security 2004 [p.38]*] when @wsu:Id is used.

Note:

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.38]*] and thus this combination must be avoided (see [*C14N 1.0 Note [p.39]*]). 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.

Note:

Canonical XML 1.1 [*C14N11 [p.40]*] is intended to address the issues that occur with Canonical XML 1.0 with regards to `xml:id`. The W3C XML Security Specifications Maintenance WG has been chartered to address how to integrate Canonical XML 1.1 with XML Security, including XML Signature [*SecSpecMaintWG [p.39]*] (See http://www.w3.org/2007/xmlsec/.)

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

```
<wsp:Policy
      Name="http://www.example.com/policies/P1"
      xmlns:wsp="http://www.w3.org/ns/ws-policy" >
  <!-- Details omitted for readability -->
</wsp:Policy>
```

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

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

4.3 Compact Policy Expression

To express a policy [p.10] in a more compact form while still using the XML Infoset, this specification defines three constructs: an attribute to decorate an assertion [p.8] , semantics for recursively nested policy operators, and a policy reference/inclusion mechanism. Each sub section below describes a construct and its equivalent normal form. To interpret a compact expression in an interoperable form, a policy expression in the compact form can be converted (see Section **4.3.6 Normalization** [p.28]) to the normal form (see Section **4.1 Normal Form Policy Expression** [p.11]).

A policy expression [p.11] consists of a `wsp:Policy` wrapper element and zero or more child and descendent elements.

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:

```
<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.39]*]) is 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 the actual value (See XML Schema Part 1 [*XML Schema Structures [p.39]*]) is 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://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/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://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/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.11] . [Definition: A **nested policy expression** is a policy expression [p.11] 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.

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.18] , this is equivalent to a nested policy expression with a single alternative that has zero assertions.

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:

```
<wsp:Policy>
    <Lorem>
        <Ipsum>
            <wsp:Policy>
            …
            </wsp:Policy>
        </Ipsum>
    </Lorem>
</wsp:Policy>
```

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:

```
(01) <wsp:Policy
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/ws-policy" >
(02)   <sp:TransportBinding>
(03)     <wsp:Policy>
(04)       <sp:AlgorithmSuite>
(05)         <wsp:Policy>
(06)           <wsp:ExactlyOne>
(07)             <sp:Basic256Rsa15 />
(08)             <sp:TripleDesRsa15 />
(09)           </wsp:ExactlyOne>
(10)         </wsp:Policy>
(11)       </sp:AlgorithmSuite>
(12)       <sp:TransportToken>
(13)         <wsp:Policy>
               <sp:HttpsToken>
                 <wsp:Policy/>
               </sp:HttpsToken>
(15)         </wsp:Policy>
(16)       </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 example above is equivalent to the following:

```
(01) <wsp:Policy
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/ws-policy" >
(02)   <wsp:ExactlyOne>
(03)     <wsp:All>
(04)       <sp:TransportBinding>
(05)         <wsp:Policy>
(06)           <sp:AlgorithmSuite>
(07)             <wsp:Policy>
(08)               <sp:Basic256Rsa15 />
(09)             </wsp:Policy>
(10)           </sp:AlgorithmSuite>
(11)           <sp:TransportToken>
(12)             <wsp:Policy>
                   <sp:HttpsToken>
                     <wsp:Policy/>
                   </sp:HttpsToken>
(14)             </wsp:Policy>
(15)           </sp:TransportToken>
               <!-- Details omitted for readability -->
(16)         </wsp:Policy>
(17)       </sp:TransportBinding>
(18)     </wsp:All>
(19)     <wsp:All>
(20)       <sp:TransportBinding>
(21)         <wsp:Policy>
(22)           <sp:AlgorithmSuite>
(23)             <wsp:Policy>
(24)               <sp:TripleDesRsa15 />
(25)             </wsp:Policy>
(26)           </sp:AlgorithmSuite>
(27)           <sp:TransportToken>
(28)             <wsp:Policy>
                   <sp:HttpsToken>
                     <wsp:Policy/>
                   </sp:HttpsToken>
(30)             </wsp:Policy>
(31)           </sp:TransportToken>
               <!-- Details omitted for readability -->
(32)         </wsp:Policy>
(33)       </sp:TransportBinding>
(34)     </wsp:All>
(35)   </wsp:ExactlyOne>
(36) </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 [p.10] are used to convey a set of capabilities, requirements, and general characteristics of entities (see **1. Introduction** [p.3]). These are generally expressible as a set of policy alternatives [p.9] . Policy operators (`wsp:Policy`, `wsp:All` and `wsp:ExactlyOne` elements) are used to group policy assertions [p.8] into policy alternatives [p.9] . 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 schema outline for the `wsp:Policy` element in the compact form is as follows:

```
<wsp:Policy … >
  ( <wsp:Policy …>…</wsp:Policy> |
    <wsp:ExactlyOne>…</wsp:ExactlyOne> |
    <wsp:All>…</wsp:All> |
    <wsp:PolicyReference … >…</wsp:PolicyReference> |
    …
  )*
</wsp:Policy>
```

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

`/wsp:Policy`

This element is the `wsp:Policy` operator.

`/wsp:Policy/wsp:Policy`

This element is a nested `wsp:Policy` operator.

`/wsp:Policy/wsp:ExactlyOne`

This element is a nested `wsp:ExactlyOne` operator.

`/wsp:Policy/wsp:All`

This element is a nested `wsp:All` operator.

`/wsp:Policy/wsp:PolicyReference`

This element references a policy expression to be included per Section **4.3.5 Policy Inclusion** [p.27] .

`/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.

`/wsp:Policy/{any}`

Additional elements MAY be specified. Such elements MUST NOT use the Web Services Policy language XML namespace name and MUST NOT contradict the semantics of the **[parent element]**.

The schema outline for the `wsp:ExactlyOne` element in the compact form is as follows:

```
<wsp:ExactlyOne>
  ( <wsp:Policy … >…</wsp:Policy> |
    <wsp:ExactlyOne>…</wsp:ExactlyOne> |
    <wsp:All>…</wsp:All> |
    <wsp:PolicyReference … >…</wsp:PolicyReference> |
    …
  )*
</wsp:ExactlyOne>
```

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

`/wsp:ExactlyOne`

This element is the `wsp:ExactlyOne` operator.

`/wsp:ExactlyOne/wsp:Policy`

This element is a nested `wsp:Policy` operator.

`/wsp:ExactlyOne/wsp:ExactlyOne`

This element is a nested `wsp:ExactlyOne` operator.

`/wsp:ExactlyOne/wsp:All`

This element is a nested `wsp:All operator`.

`/wsp:ExactlyOne/wsp:PolicyReference`

This element references a policy expression to be included per Section **4.3.5 Policy Inclusion** [p.27] .

`/wsp:ExactlyOne/{any}`

Additional elements MAY be specified. Such elements MUST NOT use the Web Services Policy language XML namespace name and MUST NOT contradict the semantics of the **[parent element]**.

The schema outline for the `wsp:All` element in the compact form is as follows:

```
<wsp:All>
  ( <wsp:Policy … >…</wsp:Policy> |
    <wsp:ExactlyOne>…</wsp:ExactlyOne> |
    <wsp:All>…</wsp:All> |
    <wsp:PolicyReference … >…</wsp:PolicyReference> |
    …
  )*
</wsp:All>
```

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

`/wsp:All`

This element is the `wsp:All` operator.

`/wsp:All/wsp:Policy`

This element is a nested `wsp:Policy` operator.

`/wsp:All/wsp:ExactlyOne`

This element is a nested `wsp:ExactlyOne` operator.

`/wsp:All/wsp:All`

This element is a nested `wsp:All` operator.

`/wsp:All/wsp:PolicyReference`

This element references a policy expression to be included per Section **4.3.5 Policy Inclusion** [p.27] .

`/wsp:All/{any}`

Additional elements MAY be specified. Such elements MUST NOT use the Web Services Policy language XML namespace name and MUST NOT contradict the semantics of the **[parent element]**.

Note:

The `wsp:All` and `wsp:ExactlyOne` elements do not allow attribute extensibility because such attributes cannot be preserved through normalization.

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

Equivalence

Use of `wsp:Policy` as an operator within a policy expression is equivalent to `wsp:All`.

A collection of assertions in an `wsp:All` operator is equivalent to a policy alternative [p.9] . For instance,

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

is equivalent to:

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

Empty

- `<wsp:All />` expresses a policy alternative 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.9] and **3.3 Policy** [p.10] , 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` is distributive over `wsp:ExactlyOne`. That is, a `wsp:All` element containing only `wsp:ExactlyOne` child elements is equivalent to a `wsp:ExactlyOne` element containing, for each possible combination of one child element from each of the `wsp:ExactlyOne` element over which being distributed, a `wsp:All` element containing that combination. 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 by repeatedly distributing wsp:All over wsp:ExactlyOne,

```
<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 />
</wsp:All>
```

is equivalent to:

```
<wsp:ExactlyOne />
```

and:

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

is equivalent to:

```
(01) <wsp:ExactlyOne />
```

For example, given the following compact policy expression:

```
(01) <wsp:Policy
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/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.18] for the assertions in Lines (04-05) yields:

```
(01) <wsp:Policy
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/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://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/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).

Consider another example, given the following compact policy expression:

```
(01) <wsp:Policy
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/ws-policy" >
(02)   <sp:RequireDerivedKeys />
(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="false"` in Line (02), and distributing wsp:All over wsp:ExactlyOne per Section **4.3.3 Policy Operators** [p.18] for the assertions in Lines (04-05) yields:

```
(01) <wsp:Policy
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/ws-policy" >
(02)   <wsp:ExactlyOne>
(03)     <wsp:All>
(04)       <sp:RequireDerivedKeys />
(05)     </wsp:All>
(06)   </wsp:ExactlyOne>
(07)   <wsp:ExactlyOne>
(08)     <wsp:All>
(09)       <sp:WssUsernameToken10 />
(10)     </wsp:All>
(11)     <wsp:All>
(12)       <sp:WssUsernameToken11 />
(13)     </wsp:All>
(14)   </wsp:ExactlyOne>
(15) </wsp:Policy>
```

Note that the assertion listed in Line (02) in the first listing expands into an alternative in Lines (03-05) 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://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/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:ExactlyOne>
(12) </wsp:Policy>
```

Note that the first alternative listed in Lines (03-05) in the second listing is combined with the two alternatives listed in Lines (08-13) in the second listing to create two alternatives in the normalized policy, Lines (03-06) and (07-10).

4.3.4 Policy References

The `wsp:PolicyReference` element is used to reference policy expressions [p.11] . 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** [p.27]).

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

```
<wsp:PolicyReference
    URI="xs:anyURI"
  ( Digest="xs:base64Binary" ( DigestAlgorithm="xs:anyURI" )? )?
    … >
   …
</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** [p.12]) 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:base64Binary` 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.

URI	Description
`http://www.w3.org/ns/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.40]*].

`/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.

`/wsp:PolicyReference/{any}`

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.11] , 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.

```
<wsp:Policy
   xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
   xmlns:wsp="http://www.w3.org/ns/ws-policy"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
   wsu:Id="Protection" >
  <sp:EncryptSignature wsp:Optional="true" />
  <sp:ProtectTokens wsp:Optional="true" />
</wsp:Policy>
```

```
<wsp:Policy
   xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
   xmlns:wsp="http://www.w3.org/ns/ws-policy" >
  <wsp:PolicyReference URI="#Protection" />
  <sp:OnlySignEntireHeadersAndBody />
</wsp:Policy>
```

```
<wsp:Policy
   xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
   xmlns:wsp="http://www.w3.org/ns/ws-policy" >
  <sp:IncludeTimestamp />
  <wsp:PolicyReference URI="#Protection" />
  <sp:OnlySignEntireHeadersAndBody />
</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.

4.3.6 Normalization

To interpret a compact expression [p.11] in an interoperable form, a compact expression may be converted to the corresponding normal form expression by the following procedure:

1. Start with the Element Information Item E (as defined in the XML Information Set [*XML Information Set [p.38]*]) of the policy expression. The **[namespace name]** of E is always `"http://www.w3.org/ns/ws-policy"`. In the base case, the **[local name]** property of E is `"Policy"`; in the recursive case, the **[local name]** property of E is `"Policy"`, `"ExactlyOne"`, or `"All"`.

2. Expand Element Information Items (as defined in the XML Information Set [*XML Information Set [p.38]*]) in the **[children]** property of E that are policy references per Section **4.3.5 Policy Inclusion** [p.27] .

3. Convert each Element Information Item C in the **[children]** property of E into normal form.

    1. If the **[namespace name]** property of C is `"http://www.w3.org/ns/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.14] and **4.3.2 Policy Assertion Nesting** [p.15] .

4. Apply the policy operator indicated by E to the normalized Element Information Items in its **[children]** property and construct a normal form per Section **4.3.3 Policy Operators** [p.18] and **4.1 Normal Form Policy Expression** [p.11] .

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.

4.4 Ignorable Policy Assertions

The `wsp:Ignorable` attribute indicates if a policy assertion is an ignorable policy assertion [p.8] . The behavior implied by an ignorable assertion is expected to be a behavior that need not be engaged for successful interoperation with the entity that includes such ignorable assertions in its policy.

The schema outline for the `wsp:Ignorable` attribute is as follows:

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

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

`/Assertion/@wsp:Ignorable`

This attribute is of type `xs:boolean`. If the actual value (See XML Schema Part 1 [*XML Schema Structures [p.39]*]) is true, the assertion is an ignorable policy assertion [p.8] . If the actual value is false, the assertion is not an ignorable policy assertion [p.8] . Omitting this attribute is semantically equivalent to including it with a value of false.

4.5 Policy Intersection

Policy intersection is OPTIONAL but, a useful tool when two or more parties express policy [p.10] and want to limit the policy alternatives [p.9] 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. Policy Intersection is a commutative operation performed on two policies that yields a policy that contains a collection of the compatible policy alternatives. (Note: while policy intersection at times is analogous with set intersection, it does not imply formal set intersection semantics). 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.

As a first approximation, an intersection algorithm is defined below that approximates compatibility of policy assertions [p.8] in a domain-independent manner. Mechanisms for determining assertion parameter [p.9] compatibility are not part of this domain-independent policy intersection. Determining whether two policy assertions [p.8] of the same type are compatible may involve domain-specific processing for purposes of determining assertion parameter [p.9] compatibility. Domain-independent policy intersection may be extended to include domain-specific processing. If a domain-specific intersection processing algorithm is required this will be known from the QName of the specific assertion type [p.9] involved in the policy alternative [p.9] . However, regardless of whether an assertion's QName indicates domain-specific processing, an implementation of the domain-independent intersection need not apply the domain-specific processing.

The domain-independent policy intersection algorithm is:

- Two policy assertions [p.8] are compatible if they have the same type [p.9] and
- If either assertion contains a nested policy expression [p.11] , 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.9] are not part of the domain-independent compatibility determination defined herein but this domain-independent policy intersection may be extended to include domain-specific processing for purposes of determining Assertion parameter [p.9] compatibility.

- If the mode is strict, two policy alternatives [p.9] A and B are compatible:
  - 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 [p.9] A and B are compatible:

  - if each assertion in A that is not an ignorable policy assertion [p.8] is compatible with an assertion in B, and
  - if each assertion in B that is not an ignorable policy assertion [p.8] is compatible with an assertion in A.

  If two alternatives are compatible, their intersection is an alternative containing all of the occurrences of all of the assertions from both alternatives (i.e., the bag union of the two), regardless of whether or not they are marked with the `wsp:Ignorable='true'` attribute.

- Two policies [p.10] 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.
- The result of policy intersection can be zero or more alternatives [p.9] . Each alternative [p.9] may contain more than one assertion [p.8] of the same type [p.9] which may come from different input policies [p.10] . See Section **3.2 Policy Alternative** [p.9] for mechanisms for determining the aggregate behavior indicated by multiple assertions [p.8] of the same policy assertion type [p.9] . If policy assertion authors did not specify the semantics of multiple assertions [p.8] of the same assertion type [p.9] within a policy alternative [p.9] and the type [p.9] and its descendant assertion types [p.9] (within a nested policy expression [p.15] outline, if any) do not allow any parameters [p.9] , then multiple assertions [p.8] of the type [p.9] within a policy alternative [p.9] in the intersection result have the same meaning as a single assertion [p.8] of the type [p.9] within the policy alternative [p.9] .

An entity applies all the behaviors implied by a policy alternative when that policy alternative is chosen from the intersection result (see **3.4 Policies of Entities in a Web Services Based System** [p.10]). If an entity includes a policy assertion type A in its policy, and this policy assertion type A does not occur in an intersected result, then that entity SHOULD not apply the behavior implied by assertion type A. If a policy assertion type Z is not included in the input policies being intersected then the intersection result is silent about the behavior implied by the assertion type Z.

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

```
(01) <wsp:Policy
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/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://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://www.w3.org/ns/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 -->
```

```
      <sp:SignedElements>
        <sp:XPath>/S:Envelope/S:Body</sp:XPath>
      </sp:SignedElements>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Because there is only one alternative (A2) in policy P1 with the same assertion 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://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
   xmlns:wsp="http://www.w3.org/ns/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 two assertions [p.8] of the type `sp:SignedParts` and two assertions [p.8] of the type [p.9] `sp:EncryptedParts`, one from each of the input Policies [p.10] . In general, whether two assertions [p.8] of the same type [p.9] are compatible or repetition is redundancy depends on the domain-specific semantics of the assertion type [p.9] . As mentioned above, if the assertions [p.8] have no parameters [p.9] and the assertions [p.8] in nested policiy expressions [p.15] have no parameters [p.9] , then multiple assertions [p.8] of the type [p.9] within a policy alternative [p.9] in the intersection result have the same meaning as a single assertion [p.8] of the type [p.9] within the policy alternative [p.9] .

Based on the semantics of multiple assertions [p.8] of the EncryptedParts assertion type [p.9] , as specified in the WS-SecurityPolicy [*WS-SecurityPolicy [p.40]*] specification, one of the `sp:EncryptedParts` assertion [p.8] in the above example is redundant.

Whether the two `sp:SignedParts` assertions [p.8] are compatible or one of them is redundant depends on the semantics defined for this assertion type [p.9] .

As another example of intersection of WS-Addressing assertions that utilize the framework intersection algorithm, consider two input policies:

```
(01) <wsp:Policy
      xmlns:wsp="http://www.w3.org/ns/ws-policy"
      xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" >
(02)   <wsp:ExactlyOne>
(03)     <wsp:All>        <!-- Alternative A5 -->
(04)       <wsam:Addressing>
(05)         <wsp:Policy/>
(06)       </wsam:Addressing>
(07)     </wsp:All>
(08)   </wsp:ExactlyOne>
(09) </wsp:Policy>
```

Lines (04)-(06) in the above policy expression contain an addressing policy assertion with the empty `<wsp:Policy/>` in line (05). The empty `<wsp:Policy/>` is a nested policy expression with an alternative that has zero assertions. In the example above, the addressing assertion indicates the use of addressing without any restriction.

```
(01) <wsp:Policy
      xmlns:wsp="http://www.w3.org/ns/ws-policy"
      xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" >
(02)   <wsp:ExactlyOne>
(03)     <wsp:All>       <!-- Alternative A6 -->
(04)       <wsam:Addressing>
(05)         <wsp:Policy>
(06)           <wsam:AnonymousResponses/>
(07)         </wsp:Policy>
(08)       </wsam:Addressing>
(09)     </wsp:All>
(10)   </wsp:ExactlyOne>
(11) </wsp:Policy>
```

Lines (04)-(08) in the above policy expression contain an addressing policy assertion with a nested policy expression in lines (05)-(06). The nested policy expression indicates that the provider requires request messages to use response endpoint EPRs that contain the anonymous URI. The nested policy expression contains an alternative that has one assertion, `wsam:AnonymousResponses`.

The two assertions in alternatives A5 and A6 have the same assertion type and have nested policy expressions. The nested policy expression within the addressing assertion in the alternative A5 contains an alternative that has zero assertions. The nested policy expression within the addressing assertion in the alternative A6 contains an alternative that has one assertion. The nested policy expressions within these two assertions are incompatible because the alternative in one is incompatible with the alternative in the other.

Therefore, the two assertions are incompatible and hence the two alternatives are incompatible.

4.6 Use of IRIs in Policy Expressions

Policy expressions use IRIs for some identifiers. This document does not define a base URI but relies on the mechanisms defined in XML Base [*XML BASE [p.38]*] and RFCs 3023 [*IETF RFC 3023 [p.39]*], 3986 [*IETF RFC 3986 [p.38]*] and 3987 [*IETF RFC 3987 [p.38]*] for establishing a base URI against which relative IRIs can be made absolute.

5. Security Considerations

It is RECOMMENDED that policies [p.10] and assertions [p.8] be integrity protected to permit the detection of tampering. This can be done using a technology such as XML DSig [*XML-Signature [p.40]*], SSL/TLS [*IETF RFC 2246 [p.39]*], or WS-Security 2004 [*WS-Security 2004 [p.38]*].

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 [p.13] 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.39] , SOAP 1.2 Messaging Framework [p.39]*] using WS-Security [*WS-Security 2004 [p.38]*] 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 [p.11] , 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 [*WS-Security 2004 [p.38]*] and WS-MetadataExchange [*WS-MetadataExchange [p.40]*] .

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

```
<wsp:Policy wsu:Id="p1">
  <wsp:PolicyReference URI="#p2"/ >
  <wsp:PolicyReference URI="#p2"/>
</wsp:Policy>

<wsp:Policy wsu:Id="p2" >
  <wsp:PolicyReference URI="#p3"/>
  <wsp:PolicyReference URI="#p3"/>
</wsp:Policy>

<wsp:Policy wsu:Id="p3" >
  <wsp:PolicyReference URI="#p4"/>
  <wsp:PolicyReference URI="#p4"/>
</wsp:Policy>

<!-- Policy/@wsu:Id p4 through p99 -->

<wsp:Policy wsu:Id="p100" >
  <wsp:PolicyReference URI="#p101"/>
  <wsp:PolicyReference URI="#p101"/>
</wsp:Policy>

<wsp:Policy wsu:Id="p101" >
  <mtom:OptimizedMimeSerialization />
</wsp: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/ns/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.39]*] for that element as defined by this specification (http://www.w3.org/2007/02/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.11] .

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 *IETF RFC 3023 [p.39]* .

Encoding considerations:

> Identical to those of "application/xml" as described in *IETF RFC 3023 [p.39]* , section 3.2, as applied to the Web Services Policy document Infoset.

Security considerations:

See section **5. Security Considerations** [p.34] in this document, and the Security Consideration section in *Web Services Policy Attachment [p.38]* .

Interoperability considerations:

There are no known interoperability issues.

Published specifications:

This document and *Web Services Policy Attachment [p.38]* .

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 *IETF RFC 3023 [p.39]* .

Base URI:

As specified in *IETF RFC 3023 [p.39]* , section 6. Also see section **4.6 Use of IRIs in Policy Expressions** [p.33] in this document and section 3.5 Use of IRIs in Policy Attachment in *Web Services Policy Attachment [p.38]* .

Macintosh File Type code:

TEXT

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]
Web Services Policy 1.5 - Attachment, A. S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez and Ü. Yalçinalp, Editors. World Wide Web Consortium, 04, September 2007. This version of the specification of the Web Services Policy 1.5 - Attachment specification is http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904. The latest version of Web Services Policy 1.5 - Attachment is available at http://www.w3.org/TR/ws-policy-attach.

[IETF RFC 2119]
Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, Author. Internet Engineering Task Force, March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt.

[IETF RFC 3986]
Uniform Resource Identifier (URI): Generic Syntax , T. Berners-Lee, R. Fielding and L. Masinter, Authors. Network Working Group, January 2005. Available at http://www.ietf.org/rfc/rfc3986.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 BASE]
XML Base, Jonathan Marsh, Editor. World Wide Web Consortium, 27 June 2001. This version of the XML Base Recommendation is http://www.w3.org/TR/2001/REC-xmlbase-20010627/. The latest version of XML Base is available at http://www.w3.org/TR/xmlbase/.

[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 Namespaces in XML 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.

[IETF RFC 3023]
IETF "RFC 3023: XML Media Types", M. Murata, S. St. Laurent, D. Kohn, July 1998. (See *http://www.ietf.org/rfc/rfc3023.txt*.)

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, 20 December 2006. Available at http://www.w3.org/TR/2006/NOTE-C14N-issues-20061220/.

[IETF RFC 2246]
IETF "RFC 2246: The TLS Protocol", T. Dierks, C. Allen, January 1999. (See *http://www.ietf.org/rfc/rfc2246.txt*.)

[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, revised 27 April 2007. This version of the SOAP Version 1.2 Part 1: Messaging Framework Recommendation is http://www.w3.org/TR/2007/REC-soap12-part1-20070427/. The latest version of SOAP Version 1.2 Part 1: Messaging Framework is available at http://www.w3.org/TR/soap12-part1/.

[SecSpecMaintWG]
XML Security Specifications Maintenance Working Group , See http://www.w3.org/2007/xmlsec.

[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 3.0.2, L. Clément, et al, Editors. Organization for the Advancement of Structured Information Standards, 19 October 2004. This version of the UDDI Version 3.0 is http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm. The latest version of the UDDI 3.0 specification is available at http://uddi.org/pubs/uddi_v3.htm.

[WS-Addressing Metadata]
Web Services Addressing 1.0 - Metadata, M. Gudgin, M. Hadley, T. Rogers and Ü. Yalçinalp, Editors. World Wide Web Consortium, 4 September 2007. This version of the Web Services Addressing 1.0 - Metadata is http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/. The latest version of Web Services Addressing 1.0 - Metadata is available at http://www.w3.org/TR/ws-addr-metadata.

[WS-SecurityPolicy]
WS-SecurityPolicy v1.2, A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist, Editors. Organization for the Advancement of Structured Information Standards, 1 July 2007. Available at http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf. Namespace document available at http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702.

[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, 26 June 2007. This version of the WSDL 2.0 specification is http://www.w3.org/TR/2007/REC-wsdl20-20070626/. The latest version of WSDL 2.0 is available at http://www.w3.org/TR/wsdl20/.

[WS-MetadataExchange]
Web Services Metadata Exchange (WS-MetadataExchange), K. Ballinger, et al, Authors. BEA Systems Inc., Computer Associates International, Inc., International Business Machines Corporation, Microsoft Corporation, Inc., SAP AG, Sun Microsystems, and webMethods, August 2006. Available at http://schemas.xmlsoap.org/ws/2004/09/mex/.

[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/.

[C14N11]
Canonical XML 1.1, J. Boyer and G. Marcy Authors. W3C Candidate Recommendation, 21 June 2007. This is a work in progress. This version is available at http://www.w3.org/TR/2007/CR-xml-c14n11-20070621. The latest version of Canonical XML 1.1 is available at http://www.w3.org/TR//xml-c14n11/.

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), Symon Chang (BEA Systems, Inc.), Paul Cotton (Microsoft Corporation), Glen Daniels (Progress Software), Doug Davis (IBM Corporation), Jacques Durand (Fujitsu Limited), Ruchith Fernando (WSO2), Christopher Ferris (IBM Corporation), William Henry (IONA Technologies, Inc.), Frederick Hirsch (Nokia), Maryann Hondo (IBM Corporation), Ondrej Hrebicek (Microsoft Corporation), Steve Jones (Layer 7 Technologies), Tom Jordahl (Adobe Systems Inc.), Paul Knight (Nortel Networks), Philippe Le Hégaret (W3C/MIT), Mark Little (JBoss Inc.), Mohammad Makarechian (Microsoft Corporation), Ashok Malhotra (Oracle Corporation), Jonathan Marsh (WSO2), Monica Martin (Sun Microsystems, Inc.), Arnaud Meyniel (Axway Software), Jeff Mischkinsky (Oracle Corporation), Dale Moberg (Axway Software), Anthony Nadalin (IBM Corporation), David Orchard (BEA Systems, Inc.), Sanjay Patil (SAP AG), Manjula Peiris (WSO2), Fabian Ritzmann (Sun Microsystems, Inc.), Daniel Roth (Microsoft Corporation), Tom Rutt (Fujitsu Limited), Sanka Samaranayake (WSO2), Felix Sasaki (W3C/Keio), Yakov Sverdlov (CA), Asir Vedamuthu (Microsoft Corporation), Sanjiva Weerawarana (WSO2), Ümit Yalçinalp (SAP AG), Prasad Yendluri (webMethods (A subsidiary of Software AG)).

Previous members of the Working Group were: Jeffrey Crump, Jong Lee, Bob Natale, Eugene Osovetsky, Bijan Parsia, Skip Snow, Seumas Soltysik, Mark Temple-Raston.

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