Abstract

This specification, Web Services Policy 1.5 - Attachment, defines two general-purpose mechanisms for associating policies, as defined in Web Services Policy 1.5 - Framework, with the subjects to which they apply. This specification also defines how these general-purpose mechanisms may be used to associate policies with WSDL and UDDI descriptions.

Status of this Document

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

This is a Last Call Working Draft of the Web Services Policy 1.5 - Attachment specification. The Last Call period ends 12 January 2007. This Working Draft was produced by the members of the Web Services Policy Working Group which is part of the W3C Web Services Activity. The Working Group expects to advance this Working Draft to Recommendation Status.
A list of changes in this version of the document and a diff-marked version against the previous version of this document are available. Major changes in this version of the document encompass the addition of a new domain expression in section 3.4.1 URI Domain Expression and the definition of equivalence of two policy properties in section 5.3 Extension to WSDL Component Model.

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

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

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

Table of Contents

1. Introduction [p.4]
2. Notations and Terminology [p.4]
   2.1 Notational Conventions [p.4]
   2.2 XML Namespaces [p.5]
   2.3 Terminology [p.6]
   2.4 Example [p.8]
3. Policy Attachment [p.9]
   3.1 Effective Policy [p.9]
   3.2 Policy Attachment Mechanisms [p.9]
   3.3 XML Element Attachment [p.9]
   3.4 External Policy Attachment [p.11]
      3.4.1 URI Domain Expression [p.13]
   4.1 Calculating Effective Policy in WSDL 1.1 [p.15]
      4.1.1 Service Policy Subject [p.17]
      4.1.2 Endpoint Policy Subject [p.17]
      4.1.3 Operation Policy Subject [p.18]
      4.1.4 Message Policy Subject [p.18]
      4.1.5 Example [p.19]
5. WS-Policy Attachment for WSDL 2.0 [p.21]
   5.1 Example [p.21]
   5.2 Attaching Policy Expressions [p.23]
   5.3 Extension to WSDL Component Model [p.26]
5.4 Effective Policy [p.28]
   5.4.1 Service Policy Subject [p.30]
   5.4.2 Endpoint Policy Subject [p.30]
   5.4.3 Operation Policy Subject [p.30]
   5.4.4 Message Policy Subject (input message) [p.31]
   5.4.5 Message Policy Subject (output message) [p.31]
   5.4.6 Message Policy Subject (input fault message) [p.31]
   5.4.7 Message Policy Subject (output fault message) [p.32]

6. Attaching Policies Using UDDI [p.32]
   6.1 Calculating Effective Policy and Element Policy in UDDI [p.33]
      6.1.1 Service Provider Policy Subject [p.33]
      6.1.2 Service Policy Subject [p.34]
      6.1.3 Endpoint Policy Subject [p.34]
   6.2 Referencing Remote Policy Expressions [p.34]
   6.3 Registering Reusable Policy Expressions [p.36]
   6.4 Registering Policies in UDDI Version 3 [p.38]

7. Security Considerations [p.39]

8. Conformance [p.39]
   8.1 External Policy Attachment Conformance [p.39]
   8.2 WSDL 1.1 Attachment Conformance [p.39]
   8.3 WSDL 2.0 Attachment Conformance [p.39]

Appendices

A. References [p.40]
   A.1 Normative References [p.40]
   A.2 Other References [p.41]

B. UDDI tModel Definitions [p.43]
   B.1 Remote Policy Reference Category System [p.43]
      B.1.1 Design Goals [p.43]
      B.1.2 tModel Definition [p.43]
      B.1.3 tModel Structure [p.43]
   B.2 Web Services Policy Types Category System [p.44]
      B.2.1 Design Goals [p.44]
      B.2.2 tModel Definition [p.44]
      B.2.3 tModel Structure [p.44]
   B.3 Local Policy Reference Category System [p.44]
      B.3.1 Design Goals [p.44]
      B.3.2 tModel Definition [p.45]
      B.3.3 tModel Structure [p.45]

C. Acknowledgements [p.45] (Non-Normative)

D. Changes in this Version of the Document [p.46] (Non-Normative)

E. Web Services Policy 1.5 - Attachment Change Log [p.46] (Non-Normative)
1. Introduction

The Web Services Policy 1.5 - Framework [Web Services Policy Framework][p.40] specification defines an abstract model and an XML-based language for expressing policies of entities in a Web services-based system. This specification, Web Services Policy 1.5 - Attachment, defines two general-purpose mechanisms for associating policies with the subjects to which they apply; the policies may be defined as part of existing metadata about the subject or the policies may be defined independently and associated through an external binding to the subject.

To enable Web Services Policy to be used with existing Web service technologies, this specification describes the use of these general-purpose mechanisms with WSDL [WSDL 1.1][p.40], [WSDL 2.0 Core Language][p.41] definitions and UDDI [UDDI API 2.0][p.40], [UDDI Data Structure 2.0][p.40], [UDDI 3.0][p.40].

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.41]. 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.41]] descriptions.

### 2.2 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.41]]).

<table>
<thead>
<tr>
<th>Prefix</th>
<th>XML Namespace</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsa</td>
<td><a href="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing</a></td>
<td>WS-Addressing Core[p.40]</td>
</tr>
<tr>
<td>wsap</td>
<td><a href="http://www.w3.org/2006/05/addressing/wsdl">http://www.w3.org/2006/05/addressing/wsdl</a></td>
<td>WS-Addressing Policy[p.42]</td>
</tr>
<tr>
<td>wsd111</td>
<td><a href="http://schemas.xmlsoap.org/wsdl/">http://schemas.xmlsoap.org/wsdl/</a></td>
<td>WSDL 1.1[p.40]</td>
</tr>
<tr>
<td>wsd120</td>
<td><a href="http://www.w3.org/2006/01/wsd1">http://www.w3.org/2006/01/wsd1</a></td>
<td>WSDL 2.0 Core Language[p.41]</td>
</tr>
<tr>
<td>wsoap12</td>
<td><a href="http://schemas.xmlsoap.org/wsd1/soap12/">http://schemas.xmlsoap.org/wsd1/soap12/</a></td>
<td>WSDL 1.1 Binding for SOAP 1.2[p.42]</td>
</tr>
<tr>
<td>(none), wsp</td>
<td><a href="http://www.w3.org/2006/07/ws-policy">http://www.w3.org/2006/07/ws-policy</a></td>
<td>This specification</td>
</tr>
<tr>
<td>xs</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
<td>XML Schema Structures[p.41]</td>
</tr>
</tbody>
</table>
All information items defined by this specification are identified by the XML namespace URI \[XML Namespaces\] [\pageref{XML Namespaces}] \(\text{http://www.w3.org/2006/07/ws-policy}\). A \[normative XML Schema\] [\pageref{XML Schema Structures}, \pageref{XML Schema Datatypes}] \[document\] can be obtained by dereferencing the XML namespace URI.

In this document reference is made to the \texttt{wsu:Id} attribute in a utility schema \[\text{http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd}\]. The \texttt{wsu:Id} attribute was added to the utility schema with the intent that other specifications requiring such an Id could reference it (as is done here).

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.

### 2.3 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 \[IETF RFC 2119\] [\pageref{IETF RFC 2119}].

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

- **effective policy** [\pageref{effective policy}] the effective policy, for a given policy subject [\pageref{policy subject}], is the combination of relevant policies. The relevant policies are those attached to policy scopes [\pageref{policy scopes}] that contain the policy subject [\pageref{policy subject}].

- **element policy** [\pageref{element policy}]
The **element policy** is the policy attached to the policy subjects associated with the element information item that contains it.

**Ignorable policy assertion**

An **ignorable policy assertion** is an assertion that may be ignored for policy intersection (as defined in 4.5 Policy Intersection).

**Merge**

A **merge** consists of serializing each policy as a policy expression, replacing their `wsp:Policy` element with a `wsp:All` element, and placing each as children of a wrapper `wsp:Policy` element.

**Policy**

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

**Policy alternative**

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

**Policy assertion**

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

**Policy attachment**

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

**Policy expression**

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

**Policy scope**

A **policy scope** is a collection of policy subjects to which a policy may apply.

**Policy subject**

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

This specification defines several mechanisms for associating policies (Web Services Policy 1.5 - Framework, [Web Services Policy Framework [p.40]]) with various XML Web service entities. For brevity, we define two sample [policy expressions [p.7]] that the remainder of this document references.


Example 2-1. Example RM Policy Expression.

(01) <wsp:Policy
   xmlns:rmp="http://docs.oasis-open.org/ws-rx/wsrmp/200602"
   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="RmPolicy" >
   (02) <rmp:RMAssertion>
   (03)   <rmp:InactivityTimeout Milliseconds="600000" />
   (04)   <rmp:BaseRetransmissionInterval Milliseconds="3000" />
   (05)   <rmp:ExponentialBackoff />
   (06)   <rmp:AcknowledgementInterval Milliseconds="200" />
   (07) </rmp:RMAssertion>
   (08) </wsp:Policy>

Example 2-2. Example X509 Security 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"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
   wsu:Id="X509EndpointPolicy" >
   (02) <sp:AsymmetricBinding>
   (03)   <sp:RecipientToken>
   (04)     <wsp:Policy>
   (05)       <sp:X509Token sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/Never">
   (06)         <wsp:Policy>
   (07)           <sp:WssX509V3Token10 />
   (08)       </wsp:Policy>
   (09)     </sp:X509Token>
   (10)   </wsp:Policy>
   (11)  </sp:RecipientToken>
   (12)  <sp:InitiatorToken>
   (13)     <wsp:Policy>
   (14)       <sp:X509Token sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
   (15)         <wsp:Policy>
   (16)           <sp:WssX509V3Token10 />
   (17)         </wsp:Policy>
   (18)     </wsp:Policy>
   (19)   </sp:X509Token>
   (20)   </sp:InitiatorToken>
   (21)  </sp:AsymmetricBinding>
(22)  </wsp:Policy>
(23) </wsp:Policy>
(24) </wsp:Policy>
(25) </wsp:Policy>
(26) </wsp:Policy>
(27) </wsp:Policy>
(28) </wsp:Policy>
(29) </wsp:Policy>
(30) </wsp:Policy>
(31) </wsp:Policy>
(32) </wsp:Policy>
(33) </wsp:Policy>
(34) </wsp:Policy>
(35) </wsp:Policy>
(36) </wsp:Policy>
The document containing both of these policy expressions is assumed to be located at http://www.example.com/policies. Per Section 3.2 Policy Identification of Web Services Policy 1.5 - Framework [Web Services Policy Framework [p.40]], the IRIs used for these policy expressions [p.7] in the remainder of this document are http://www.example.com/policies#RmPolicy and http://www.example.com/policies#X509EndpointPolicy, for the examples in Example 2-1 [p.8] and Example 2-2 [p.8], respectively.

3. Policy Attachment

This section defines two general-purpose mechanisms for associating policies [p.7] with one or more policy subjects [p.7]. The first allows XML-based descriptions of resources (represented as XML elements) to associate policy as part of their intrinsic definition. The second allows policies to be associated with arbitrary policy subjects independently from their definition.

In addition it defines the processing rules for scenarios where multiple policies are attached to a policy subject.

3.1 Effective Policy

Policies [p.7] will often be associated with a particular policy subject [p.7] using multiple policy attachments [p.7]. For example, there may be attachments at different points in a WSDL description that apply to one policy subject, and other attachments may be made by UDDI and other mechanisms.

When multiple attachments are made, [Definition: the effective policy, for a given policy subject [p.7], is the combination of relevant policies. The relevant policies are those attached to policy scopes [p.7] that contain the policy subject [p.7].]

This combination can be achieved by: [Definition: a merge consists of serializing each policy as a policy expression [p.7], replacing their wsp:Policy element with a wsp:All element, and placing each as children of a wrapper wsp:Policy element.] The resulting policy expression is considered to represent the combined policy of all of the attachments to that policy subject.

Such calculated policy expressions have no meaningful IRI of their own.

3.2 Policy Attachment Mechanisms

This section defines two general-purpose mechanisms for associating policies [Web Services Policy Framework [p.40]] with one or more policy subjects [p.7]. The first allows XML-based descriptions of resources to associate policy as part of their intrinsic definition. The second allows policies to be associated with arbitrary policy subjects independently from their definition.

3.3 XML Element Attachment

Since policy assertions[7] are strongly typed by the authors, the precise semantics of how element policy is to be processed once discovered is domain-specific; however, implementations are likely to follow the precedent specified in the section below on WSDL [WSDL 1.1][40] and Policy.

This specification defines a global attribute that allows policy expressions[7] to be attached to an arbitrary XML element. The following is the schema definition for the wsp:PolicyURIs attribute:

```
(01) <xs:schema>
(02)   <xs:attribute name="PolicyURIs">
(03)     <xs:simpleType>
(04)       <xs:list itemType="xs:anyURI" />
(05)     </xs:simpleType>
(06)   </xs:attribute>
(07) </xs:schema>
```

The namespace URI [XML Namespaces][41] for this attribute is http://www.w3.org/2006/07/ws-policy.

The wsp:PolicyURIs attribute contains a white space-separated list of one or more IRIs [IETF RFC 3987][40]. When this attribute is used, each of the values identifies a policy expression[7] as defined by [Web Services Policy Framework][40]. If more than one IRI is specified, the individual referenced policies[7] need to be merged[9] together to form a single element policy expression[7]. The resultant policy[7] is then associated with the element information item's element policy[10] property. [Definition: The element policy is the policy[7] attached to the policy subjects[7] associated with the element information item that contains it.]


An example of element policy[10] through the use of this global attribute is given below using the sample policies stated in Section 2.4 Example[8].

If the policies[7] referenced by the following XML element

```
(01) <MyElement wsp:PolicyURIs="
(02)     http://www.example.com/policies#RmPolicy
(03)     http://www.example.com/policies#X509EndpointPolicy" />
```

have been processed and merged[9], it would result in an element policy[10] whose XML 1.0 representation is listed in Example 3-1[10]:

```
Example 3-1. Example Merged Policy Expression.
```

```
(01) <wsp:Policy
 xmlns:rmp="http://docs.oasis-open.org/ws-rx/wsramp/200602"
 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
 xmlns:wsp="http://www.w3.org/2006/07/ws-policy">
(02)   <rmp:RMAssertion>
(03)     <rmp:InactivityTimeout Milliseconds="600000" />
(04)       <rmp:BaseRetransmissionInterval Milliseconds="3000" />
(05)     </rmp:RMAssertion>
(06)   </wsp:Policy>
```
3.4 External Policy Attachment

This mechanism allows policies to be associated with a policy subject independent of that subject’s definition and/or representation through the use of a wsp:PolicyAttachment element.

This element has three components: the policy scope of the attachment, the policy expressions being bound, and optional security information. The policy scope of the attachment is defined using one or more extensible domain expressions that identify policy subjects, typically using IRIs.

Domain expressions identify the domain of the association. That is, the set of policy subjects that will be considered for inclusion in the scope using an extensible domain expression model. Domain expressions identify policy subjects to be included within the policy scope. Domain expressions yield an unordered set of policy subjects for consideration.

For the purposes of attaching policy to a policy subject through this mechanism, any policy expression contained inside of the wsp:AppliesTo element MUST NOT be considered in scope. For example, an Endpoint Reference may be used as a domain expression, and it may contain policy expressions within it, but this policy expressions are not considered in scope with respect to the wsp:PolicyAttachment element using it.
The following is the pseudo-schema for the `wsp:PolicyAttachment` element:

(01) `<wsp:PolicyAttachment ... >
(02)   `<wsp:AppliesTo`
(03)     `<x:DomainExpression/> +
(04)   `</wsp:AppliesTo>`
(05)   (`<wsp:Policy>...</wsp:Policy>` |
(06)   `<wsp:PolicyReference>...</wsp:PolicyReference>` ) +
(07)   `<wsse:Security>...</wsse:Security> ?
(08)   ...
(09) `</wsp:PolicyAttachment>`

The following describes the attributes and elements listed in the pseudo-schema outlined above:

/wsp:PolicyAttachment

  This describes an external [policy attachment] [p.7].

/wsp:PolicyAttachment/wsp:AppliesTo

  This required element’s children describe the [policy scope] [p.7].

/wsp:PolicyAttachment/wsp:AppliesTo/{any}

  These child elements MUST specify and/or refine the domain expression(s) that define the [policy scope] [p.7]. They MUST NOT contradict the semantics of their root element; if an element is not recognized, it SHOULD be ignored. Domain expressions are XML elements that describe [policy subjects] [p.7] within a [policy scope] [p.7]. When more than one domain expression is present, the [policy scope] [p.7] contains the union of the [policy subjects] [p.7] identified by each expression.

/wsp:PolicyAttachment/wsp:Policy

  This element is a [policy expression] [p.7] representing a [policy] [p.7] that is attached to the [policy subjects] [p.7] within the [policy scope] [p.7].


  This element references a [policy expression] [p.7] to be attached to the [policy subjects] [p.7] that are in the [policy scope] [p.7]. Refer to Web Services Policy 1.5 - Framework [Web Services Policy Framework] [p.40] for additional details.

/wsp:PolicyAttachment/wsse:Security

  This element is of type `xs:boolean` and allows security information such as signatures to be included. The syntax of this element is described in WS-Security [WS-Security 2004] [p.41].

/wsp:PolicyAttachment/@{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.
3.4 External Policy Attachment

Other child elements for binding constructs MAY be specified but MUST NOT contradict the semantics of the parent element; if an element is not recognized, it SHOULD be ignored.

Domain expressions are used to identify entities such as endpoints, messages or resources with which a policy can be associated. For example, domain expressions may be used to refer to WSDL 1.1 definitions, WSDL 2.0 components, endpoint references, etc.

The following example illustrates the use of this mechanism with an EndpointReference domain expression for a deployed endpoint as defined in Web Services Addressing [WS-Addressing Core][p.40]:

```
(01) <wsp:PolicyAttachment>
    (02)   <wsp:AppliesTo>
        (03)     <wsa:EndpointReference>
            (04)       <wsa:Address>http://www.example.com/acct</wsa:Address>
        (05)     </wsa:EndpointReference>
    (06)   </wsp:AppliesTo>
    (07)   <wsp:PolicyReference URI="http://www.example.com/policies#RmPolicy" />
    (08) </wsp:PolicyAttachment>
```

In this example, the policy expression [p.7] at http://www.example.com/policies#RmPolicy applies to all interactions with the endpoint at http://www.example.com/acct.

3.4.1 URI Domain Expression

This section defines a domain expression for identifying resources as policy subjects [p.7] for the external attachment mechanism. The following is a pseudo-schema for the URI domain expression:

```
(01) <wsp:PolicyAttachment ...
    (02)   <wsp:AppliesTo>
        (03)     <wsp:URI ...
            (04)       >xs:anyURI</wsp:URI> *
        (05)     </wsp:AppliesTo>
        (06)     ...
    (07)   </wsp:PolicyAttachment>
```

The following describes the URI domain expression element listed in the pseudo-schema outlined above:

```
/wsp:PolicyAttachment/wsp:AppliesTo/wsp:URI
```

This element is an absolute IRI that references a resource as a policy subject [p.7]. There is no requirement that the IRI be resolvable; retrieval mechanisms are beyond the scope of this specification.

```
/wsp:PolicyReference/wsp:AppliesTo/wsp:URI/@{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.
URI domain expressions are used to identify resources that are identified using IRI or IRI References (such as endpoint, message or operation definitions) with which policies can be associated. For example, URI domain expressions can be used to identify WSDL 1.1 definitions, WSDL 2.0 components, etc. IRI References for WSDL 2.0 components are defined in Appendix C of the Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. The following example illustrates the use of URI domain expression with a WSDL 2.0 IRI Reference:

```xml
<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsp:URI>http://example.org/TicketAgent.wsdl20#wsdl.endpoint(TicketAgentService/Endpoint)</wsp:URI>
  </wsp:AppliesTo>
  <wsp:PolicyReference URI="http://www.example.com/policies#RmPolicy" />
</wsp:PolicyAttachment>
```

In this example, the policy expression applies to all interactions with the endpoint at http://example.org/TicketAgent.wsdl20#wsdl.endpoint(TicketAgentService/Endpoint). Similarly, an XPointer can be used as an IRI Reference to identify a description in a WSDL 1.1 document.

### 4. Attaching Policies Using WSDL 1.1

The RECOMMENDED means of associating a policy with a policy subject that has a WSDL 1.1 description is to attach a reference to the policy within the WSDL component corresponding to the target policy subject.

WSDL 1.1 disallows the use of extensibility elements on certain elements and the use of extensibility attributes on others. However, the WS-I Basic Profile 1.1 overrules this restriction and allows element extensibility everywhere. Therefore, the policy reference SHOULD be attached using `wsp:PolicyReference` as child element unless it is absolutely necessary to maintain the original WSDL 1.1 restriction, in which case the `@wsp:PolicyURIs` attribute MAY be used for the following WSDL elements:

- `wsdl11:portType`
- `wsdl11:portType/wsdl11:operation/wsdl11:input`
- `wsdl11:portType/wsdl11:operation/wsdl11:output`
- `wsdl11:portType/wsdl11:operation/wsdl11:fault`

If it is necessary to include the actual policy expressions within the WSDL description itself, it is RECOMMENDED that their `wsp:Policy` elements be included as children of the `wsdl11:definition` element, and referenced using the mechanisms just described. Alternatively, the policy expressions MAY be made available through some other means, such as WS-MetadataExchange.
To ensure that consumers of policy-annotated WSDL elements are capable of processing such policy attachments, attachments using `wsp:PolicyReference` SHOULD be marked as a mandatory extension (e.g., with a `@wsdl11:required="true"` attribute).

The rest of this section defines how to interpret the policy attachments when they appear within a WSDL description.

### 4.1 Calculating Effective Policy in WSDL 1.1

Policy attachments in WSDL 1.1 can be used to associate policies with four different types of policy subject, identified as the service policy subject, the endpoint policy subject, the operation policy subject, and the message policy subject. These policy subjects should be considered as nested, due to the hierarchical nature of WSDL.

When attaching a policy to a WSDL element, a policy scope is implied for that attachment. The policy scope only contains the policy subject associated with that element and not those associated with the children of that element. Therefore, it is RECOMMENDED that each policy assertion contained within a WSDL element’s element policy should have the correct semantic such that the policy subject for that assertion is that WSDL element. For example, assertions that describe behaviours regarding the manipulation of messages should only be contained within policies attached to WSDL message elements.

Figure 1 represents how the effective policies, with regard to WSDL, are calculated for each of these policy subjects. In the diagram, the dashed boxes represent policy scope s implied by WSDL elements. For a particular policy subject, the effective policy MUST merge the element policy of each element with a policy scope that contains the policy subject.

For abstract WSDL definitions, the element policy is considered an intrinsic part of the definition and applies to all uses of that definition. In particular, it MUST be merged into the effective policy of every implementation of that abstract WSDL definition.

Policies that are attached to a deployed resource (e.g., services or ports) are only considered in the effective policy of that deployed resource itself.
When attaching policies at different levels of the WSDL hierarchy, care must be taken. A message exchange with an endpoint MAY be described by the effective policies in all four subject types simultaneously.

For example, in Figure 4-1, for a particular input message to a deployed endpoint, there are four policy subjects involved, each with their own effective policy. There is an effective policy for the message, as well as an effective policy for the parent operation of that message, an effective policy for the deployed endpoint, and the effective policy for the service as a whole. All four effective policies are applicable in relation to that specific input message.

It is RECOMMENDED that, where specific policy assertions associated with one policy subject are only compatible with specific policy assertions on another policy subject in the same hierarchical chain, the policies containing these assertions should be attached within a single WSDL binding hierarchy.
For any given port, the policy alternatives for each policy subject type SHOULD be compatible with each of the policy alternatives at each of the policy subjects' parent and child policy subjects, such that choices between policy alternatives at each level are independent of each other.

The rest of this section describes these policy subject types, and how the effective policy for each policy subject is calculated.

### 4.1.1 Service Policy Subject

The following WSDL 1.1 element is considered as the service policy subject:

- `wsdl11:service`

This element MAY have `element policy` as per Section 3. Policy Attachment, and if present MUST be merged into the effective policy of the WSDL service policy subject.

A policy associated with a service policy subject applies to any message exchange using any of the endpoints offered by that service.

### 4.1.2 Endpoint Policy Subject

The following WSDL 1.1 elements collectively describe an endpoint:

- `wsdl11:port`
- `wsdl11:portType`
- `wsdl11:binding`

These elements MAY have `element policy` as per Section 3. Policy Attachment. The policy scope implied by each of these elements contains the endpoint policy subject representing the deployed endpoint.

Since the `wsdl11:portType` may be used by more than one binding, it is RECOMMENDED that only policies containing abstract (i.e., binding independent) assertions should be attached to this type of element.

Policies associated with an endpoint policy subject apply to any message exchange made using that endpoint.

The effective policy for a WSDL endpoint policy subject includes the `element policy` of the `wsdl11:port` element that defines the endpoint merged with the `element policy` of the referenced `wsdl11:binding` element and the `element policy` of the referenced `wsdl11:port-Type` element that defines the interface of the endpoint.
4.1.3 Operation Policy Subject

The following WSDL 1.1 elements collectively describe an operation:

- `wsdl11:portType/wsdl11:operation`
- `wsdl11:binding/wsdl11:operation`

These elements MAY have [element policy][p.10] as per Section [3. Policy Attachment][p.9].

The [policy scope][p.7] implied by each of these elements contains the operation policy subject representing the specific operation of the endpoint policy subject.

Since the `wsdl11:portType/wsdl11:operation` may be used by more than one binding, it is RECOMMENDED that only policies containing abstract (i.e., binding independent) assertions should be attached to this type of element.

Policies associated with an operation policy subject apply to the message exchange described by that operation.


4.1.4 Message Policy Subject

The following WSDL 1.1 elements are used to describe messages:

- `wsdl11:message`
- `wsdl11:portType/wsdl11:operation/wsdl11:input`
- `wsdl11:portType/wsdl11:operation/wsdl11:output`
- `wsdl11:portType/wsdl11:operation/wsdl11:fault`
- `wsdl11:binding/wsdl11:operation/wsdl11:input`
- `wsdl11:binding/wsdl11:operation/wsdl11:output`
- `wsdl11:binding/wsdl11:operation/wsdl11:fault`

These elements MAY have [element policy][p.10] as per Section [3. Policy Attachment][p.9].

The [policy scope][p.7] implied by these elements contains the message policy subject representing the specific input, output, or fault message in relation to the operation policy subject.
Policies associated with a message policy subject apply to that message (i.e. input, output or fault message).

The effective policy for a specific WSDL message (i.e., input, output, or fault message) is calculated in relation to a specific port, and includes the element policy of the \texttt{wsdl11:message} element that defines the message’s type merged with the element policy of the \texttt{wsdl11:binding} and \texttt{wsdl11:portType} message definitions that describe that message.

For example, the effective policy of a specific input message for a specific port would be the merge of the \texttt{wsdl11:message} element defining the message type, the \texttt{wsdl11:portType/wsdl11:operation/wsdl11:input} element, and the corresponding \texttt{wsdl11:binding/wsdl11:operation/wsdl11:input} element for that message.

Since a \texttt{wsdl11:message} may be used by more than one \texttt{wsdl11:portType}, it is RECOMMENDED that only policies containing abstract (i.e., binding independent) assertions should be attached to this type of element.

Since \texttt{wsdl11:input}, \texttt{wsdl11:output}, and \texttt{wsdl11:fault} elements in a \texttt{wsdl11:portType/wsdl11:operation} may be used by more than one binding, it is RECOMMENDED that only policies containing abstract (i.e., binding independent) assertions should be attached to these types of elements.

Care should be taken when attaching policies to outbound messages as the result may not be what is expected. For example, expressing a choice on a service’s outbound message without a mechanism for a requester of that service to communicate its choice to the service before the outbound message is sent may not result in the desired behaviours. It is therefore RECOMMENDED that policy alternatives on outbound messages SHOULD be avoided without the use of some form of mutual policy exchange between the parties involved.

4.1.5 Example

As an example of the combination of these policy subjects and effective policy calculation, consider the WSDL type definition in Example 4-1 that references policies.

Example 4-1. Example Policy Attached to WSDL.

```xml
<wsp:Policy wsu:Id="RmPolicy">
    <rmp:RMAssertion>
        <rmp:InactivityTimeout Milliseconds="600000"/>
        <rmp:BaseRetransmissionInterval Milliseconds="3000"/>
        <rmp:ExponentialBackoff/>
        <rmp:AcknowledgementInterval Milliseconds="200"/>
    </rmp:RMAssertion>
</wsp:Policy>
```
For endpoints bound to StockQuoteSoapBinding, the effective policy of the endpoint is listed in Example 3-1 (above). For the GetLastTradePrice operation, an additional message-level effective policy is in effect for the input message, whose XML 1.0 representation is listed in Example 4-2.

**Example 4-2. Example Message Security Policy Expression.**

```xml
<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="SecureMessagePolicy" >
  <sp:SignedParts>
    <sp:Body />
  </sp:SignedParts>
  <sp:EncryptedParts>
    <sp:Body />
  </sp:EncryptedParts>
</wsp:Policy>
```
5. WS-Policy Attachment for WSDL 2.0

This section describes a mechanism for associating policy expressions with Web service constructs in WSDL 2.0. The mechanism consists of:

- A model for attaching policies to WSDL 2.0 constructs. The model defines:
  - A partitioning of WSDL constructs into service, endpoint, operation and message policy subjects.
  - The semantics of attaching a policy to each policy subject.
  - How to combine policies attached to more than one WSDL component within a single policy subject.

- An XML representation of policy expressions attached to WSDL 2.0 constructs and annotating those policy expressions as required extensions using the WSDL-defined extensibility flag @wsdl20:required.

2.2 XML Namespaces[p.5] lists all the XML Namespaces that are used in this section. (XML elements without a namespace prefix are from the Web Services Policy XML Namespace.)

5.1 Example

The example below illustrates the use of WS-Policy Attachment for WSDL 2.0:

Example 5-1. Example Policy Attached to WSDL 2.0

```
(01) <wsdl20:description>
(02) ...
(03)   <Policy wsu:Id="common">
(04)     <mtom:OptimizedMimeSerialization wsp:Optional="true"/>
(05)     <wsap:UsingAddressing />
(06)   </Policy>
(07)   <Policy wsu:Id="secure">
(08)     <ExactlyOne>
(09)       <sp:TransportBinding>...<sp:TransportBinding>
(10)       <sp:AsymmetricBinding>...<sp:AsymmetricBinding>
(11)     </ExactlyOne>
(12)   </Policy>
(13)   <wsdl20:binding name="SecureBinding"
(14)       interface="tns:RealTimeDataInterface" />
(15)   <PolicyReference URI="#secure" />
(16)   <wsdl20:operation name="GetRealQuote">...<wsdl20:operation>
(17)   ...
(18)   </wsdl20:binding>
(19)   <wsdl20:service name="RealTimeDataService"
(20)       interface="tns:RealTimeDataInterface" />
(21)   <wsdl20:endpoint name="RealTimeDataPort"
```
The SecureBinding WSDL binding description describes a binding for an interface that provides real-time quotes and book information on securities. (The prefixes wsdl20 and tns are used here to denote the Web Services Description Language 2.0 XML Namespace and the target namespace of this WSDL document respectively.) To require the use of security for these offerings, a [policy expression][p.7] that requires the use of either transport-level or message-level security is attached to the binding description. The [policy expression][p.7] applies to any message exchange associated with any endpoint that supports this binding description.

The RealTimeDataPort WSDL endpoint description describes an endpoint that supports the SecureBinding WSDL binding description. To require the use of addressing and allow the use of optimization (Optimized MIME Serialization as defined in the MTOM specification [MTOM][p.41]), a [policy expression][p.7] that represents the addressing requirement and optimization capability is attached to the endpoint description. The [policy expression][p.7] applies to any message exchange associated with the RealTimeDataPort endpoint.

In the above example, the #secure and #common [policy expressions][p.7] attached to the SecureBinding WSDL binding and RealTimeDataPort WSDL endpoint descriptions collectively apply to any message exchange associated with the RealTimeDataPort endpoint. The example below represents the combination of these two [policies][p.7], that is, the [effective policy][p.9] for the RealTimeDataPort endpoint.

**Example 5-2. Effective Policy for the RealTimeDataPort endpoint**

```xml
(01) <Policy>
(02)  <All>
(03)    <Policy>
(04)      <mtom:OptimizedMimeSerialization wsp:Optional="true"/>
(05)      <wsap:UsingAddressing/>
(06)    </Policy>
(07)  </Policy>
(08)  <ExactlyOne>
(09)    <sp:TransportBinding>.../sp:TransportBinding>
(10)    <sp:AsymmetricBinding>.../sp:AsymmetricBinding>
(11)  </ExactlyOne>
(12) </Policy>
(13) </All>
(14) </Policy>
```
5.2 Attaching Policy Expressions

Policy attachment points in a WSDL 2.0 document are:

- `wsdl20:service`
- `wsdl20:endpoint`
- `wsdl20:binding`
- `wsdl20:binding/wsdl20:operation`
- `wsdl20:binding/wsdl20:fault`
- `wsdl20:binding/wsdl20:operation/wsdl20:input`
- `wsdl20:interface`
- `wsdl20:interface/wsdl20:operation`
- `wsdl20:interface/wsdl20:fault`
- `wsdl20:interface/wsdl20:operation/wsdl20:input`
- `wsdl20:interface/wsdl20:operation/wsdl20:output`
- `wsdl20:interface/wsdl20:operation/wsdl20:infault`

Any of these elements MAY have one or more `Policy` or `PolicyReference` child elements.

Policy attachment points in a WSDL document are associated with specific policy subjects as described in the table below. There are four policy subjects in WSDL: the service policy subject, the endpoint policy subject, the operation policy subject and the message policy subject. When a policy expression is attached to a policy subject in a WSDL document, capabilities and requirements represented by the policy expression apply to any message exchange or message associated with (or described by) the policy subject.
### Table 5-1. Association of Policy Attachment Points with Policy Subjects

<table>
<thead>
<tr>
<th>Policy Attachment Point in a WSDL document</th>
<th>WSDL Component</th>
<th>Policy Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wsdl20:service</code></td>
<td>Service</td>
<td>Service</td>
</tr>
<tr>
<td><code>wsdl20:endpoint</code></td>
<td>Endpoint</td>
<td>Endpoint</td>
</tr>
<tr>
<td><code>wsdl20:binding</code></td>
<td>Binding</td>
<td></td>
</tr>
<tr>
<td><code>wsdl20:interface</code></td>
<td>Interface</td>
<td></td>
</tr>
<tr>
<td><code>wsdl20:binding/wsl20:operation</code></td>
<td>Binding Operation</td>
<td>Operation</td>
</tr>
<tr>
<td><code>wsdl20:interface/wsl20:operation</code></td>
<td>Interface Operation</td>
<td>Operation</td>
</tr>
<tr>
<td><code>wsdl20:interface/wsl20:operation/wsl20:input</code></td>
<td>Interface Message Reference whose <code>{direction}</code> property is ‘in’</td>
<td></td>
</tr>
<tr>
<td><code>wsdl20:interface/wsl20:operation/wsl20:output</code></td>
<td>Interface Message Reference whose <code>{direction}</code> property is ‘out’</td>
<td></td>
</tr>
</tbody>
</table>
### Policy Attachment Point in a WSDL document

<table>
<thead>
<tr>
<th>Policy Attachment Point in a WSDL document</th>
<th>WSDL Component</th>
<th>Policy Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wsdl20:binding/wsdl20:fault</code></td>
<td>Binding Fault</td>
<td></td>
</tr>
<tr>
<td><code>wsdl20:interface/wsdl20:fault</code></td>
<td>Interface Fault</td>
<td></td>
</tr>
<tr>
<td><code>wsdl20:interface/wsdl20:operation/wsdl20:infault</code></td>
<td>Interface Fault Reference whose <code>{direction}</code> property is ‘in’</td>
<td></td>
</tr>
<tr>
<td><code>wsdl20:binding/wsdl20:fault</code></td>
<td>Binding Fault</td>
<td></td>
</tr>
<tr>
<td><code>wsdl20:interface/wsdl20:fault</code></td>
<td>Interface Fault</td>
<td></td>
</tr>
<tr>
<td><code>wsdl20:interface/wsdl20:operation/wsdl20:outfault</code></td>
<td>Interface Fault Reference whose <code>{direction}</code> property is ‘out’</td>
<td></td>
</tr>
</tbody>
</table>

For a WSDL component, the attached [policy][p.7] (extension to the WSDL component model is described in [5.3 Extension to WSDL Component Model][p.26]) is considered an intrinsic part of the WSDL component definition and applies to all uses of that definition. For example, when attached to a WSDL Interface component, capabilities and requirements represented by [policy][p.7] apply to all the use of this WSDL Interface description. When attached to a WSDL Binding component, capabilities and requirements represented by [policy][p.7] apply to all the Endpoints that support this binding description.

A [policy][p.7] associated with a service policy subject applies to any message exchange (that is explicitly described by the Interface component in the Service component’s `{interface}` property) using any of the endpoints offered by that service.

A [policy][p.7] associated with an endpoint policy subject apply to any message exchange (that is explicitly described by the Interface component in the Service component’s `{interface}` property of the Endpoint component’s `{parent}` property) made using that endpoint. Given that a WSDL Interface component may be used by one or more binding descriptions, it is RECOMMENDED that only a [policy][p.7] containing [policy assertions][p.7] that apply to any possible binding description should be attached.
Policies [p.7] associated with an operation policy subject apply to the message exchange described by that operation. Given that a WSDL Interface Operation component may be used by one or more binding descriptions, it is RECOMMENDED that only a policy [p.7] containing policy assertions [p.7] that apply to any possible binding description should be attached.

Policies [p.7] associated with a message policy subject apply to that message (input, output or fault). Given that a WSDL Interface Message Reference, Interface Fault, Interface Fault Reference components may be used by one or more binding descriptions, it is RECOMMENDED that only a policy [p.7] containing policy assertions [p.7] that apply to any possible binding should be attached.


Policy expressions [p.7] can be included within a WSDL document or may reside external to a WSDL document. If including policy expressions [p.7] with a WSDL document is the chosen approach, it is RECOMMENDED that the Policy elements are included as children of the wsdl20:description element after the wsdl20:types element and referenced using the PolicyReference elements.

To mandate the processing of a policy expression [p.7] attached to a policy attachment point in a WSDL document, the expression MUST be marked as required using the @wsdl20:required flag.

If the Policy elements are included as children of the wsdl20:description element, these Policy elements MUST NOT be marked as required using the @wsdl20:required. (Note: these policy expressions [p.7] may be included as children of the wsdl20:description element and may not be attached to any policy attachment point in a WSDL document.)

5.3 Extension to WSDL Component Model

This document adds an optional {policy} property to the following WSDL components:

- Service
- Endpoint
- Binding
- Binding Operation
- Binding Fault
5.3 Extension to WSDL Component Model

- Binding Message Reference
- Binding Fault Reference
- Interface
- Interface Operation
- Interface Fault
- Interface Message Reference
- Interface Fault Reference

The \{policy\} property, when present, represents the capabilities and requirements as a \[policy\] . The value of the \{policy\} property is a \[policy\] as defined by Section 3 - Policy Model in the WS-Policy specification [Web Services Policy Framework\[p.40\]]. The following table describes the mapping from XML representation to the \{policy\} property.

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>A [policy] corresponding to the \merge [p.9] of Policy or PolicyReference elements, if any, in the [children] of the \wsdl20:service\ element.</td>
</tr>
<tr>
<td>Endpoint</td>
<td>A [policy] corresponding to the \merge [p.9] of Policy or PolicyReference elements, if any, in the [children] of the \wsdl20:endpoint\ element.</td>
</tr>
<tr>
<td>Binding</td>
<td>A [policy] corresponding to the \merge [p.9] of Policy or PolicyReference elements, if any, in the [children] of the \wsdl20:binding\ element.</td>
</tr>
<tr>
<td>Binding Operation</td>
<td>A [policy] corresponding to the \merge [p.9] of Policy or PolicyReference elements, if any, in the [children] of the \wsdl20:binding/wsd120:operation\ element.</td>
</tr>
<tr>
<td>Binding Fault</td>
<td>A [policy] corresponding to the \merge [p.9] of Policy or PolicyReference elements, if any, in the [children] of the \wsdl20:binding/wsd120:fault\ element.</td>
</tr>
<tr>
<td>Binding Message Reference</td>
<td>A [policy] corresponding to the \merge [p.9] of Policy or PolicyReference elements, if any, in the [children] of the \wsdl20:binding/wsd120:operation/wsd120:input or \wsdl20:binding/wsd120:operation/wsd120:output\ element.</td>
</tr>
<tr>
<td>Binding Fault Reference</td>
<td>A [policy] corresponding to the \merge [p.9] of Policy or PolicyReference elements, if any, in the [children] of the \wsdl20:binding/wsd120:operation/wsd120:infault or \wsdl20:binding/wsd120:operation/wsd120:outfaul\ element.</td>
</tr>
</tbody>
</table>

27
## 5.4 Effective Policy

The following diagram illustrates the four policy subjects in WSDL and how the effective policy is calculated for each of these:

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>A <code>policy</code> corresponding to the <code>merge</code> of Policy or PolicyReference elements, if any, in the <code>children</code> of the <code>wsdl20:interface</code> element.</td>
</tr>
<tr>
<td>Interface Operation</td>
<td>A <code>policy</code> corresponding to the <code>merge</code> of Policy or PolicyReference elements, if any, in the <code>children</code> of the <code>wsdl20:operation</code> element, if any.</td>
</tr>
<tr>
<td>Interface Fault</td>
<td>A <code>policy</code> corresponding to the <code>merge</code> of Policy or PolicyReference elements, if any, in the <code>children</code> of the <code>wsdl20:fault</code> element.</td>
</tr>
<tr>
<td>Interface Message Reference</td>
<td>A <code>policy</code> corresponding to the <code>merge</code> of Policy or PolicyReference elements, if any, in the <code>children</code> of the <code>wsdl20:input</code> or <code>wsdl20:output</code> element.</td>
</tr>
<tr>
<td>Interface Fault Reference</td>
<td>A <code>policy</code> corresponding to the <code>merge</code> of Policy or PolicyReference elements, if any, in the <code>children</code> of the <code>wsdl20:infault</code> or <code>wsdl20:outfault</code> element.</td>
</tr>
</tbody>
</table>

Two `policy` properties are equivalent when they represent policies that contain the same number of `policy alternatives`, and each `policy alternative` in the first policy is equivalent to some `policy alternative` in the second policy, and conversely.

Two `policy alternatives` are equivalent when each `policy assertion` in the first `policy alternative` is equivalent to some `policy assertion` in the second `policy alternative`, and conversely. If either `policy alternative` contains multiple `policy assertions` of the same type, `policy alternative` equality is dependent on the semantics of that assertion type.

Two `policy assertions` are equivalent if they have the same QName, if either policy assertion is an `ignorable policy assertion`, both assertions must be `ignorable policy assertions` and, if either `policy assertion` has a nested policy, both assertions must have a nested policy and the nested policies must be equal. If either assertion contains policy assertion parameters, then the policy assertion parameters SHOULD be compared for equality. Comparing policy assertion parameters for equality is not defined by this document, but `policy assertion` equality may be further refined by the corresponding `policy assertion` specification.
5.4 Effective Policy

Policy Subjects and Effective Policy in WSDL 2.0
Figure 5-1. Policy Subjects and Effective Policy in WSDL 2.0

If multiple policies are attached to WSDL components that collectively represent a policy subject then the effective policy of these policies applies. (For example, there is a policy attached to an Endpoint component that describes the component and there is a policy attached to the Binding component in the Endpoint component’s binding property.) The effective policy is the merge of the policies that are attached to the same policy subject. The rest of this section describes how the effective policy is calculated for each of these policy subjects.

5.4.1 Service Policy Subject

The effective policy of a service policy subject is the policy in the policy property of a Service component that describes the service.

5.4.2 Endpoint Policy Subject

The effective policy of an endpoint policy subject is the merge of policies in the policy properties of:

- An Endpoint component that describes the endpoint,
- The Binding component in the Endpoint component’s binding property,
- The Interface component in the Service component’s interface property of the Endpoint component’s parent property and
- The Interface components in the extended interfaces property of the Interface component in the Service component’s interface property of the Endpoint component’s parent property.

5.4.3 Operation Policy Subject

If the Binding component has an Interface component in the interface property, then the effective policy of an operation policy subject MAY be calculated by merging the policies in the policy properties of:

- The Interface Operation component that describes the operation and
- The Binding Operation component (if any) whose interface operation property has the Interface Operation component.

If the Binding component does not have an Interface component in the interface property, then the effective policy of an operation policy subject MUST be calculated in relation to a specific endpoint, and is the policy in the policy property of the Interface Operation component that describes the operation.
5.4.4 Message Policy Subject (input message)

If the Binding component has an Interface component in the \{interface\} property, then the effective policy of an input message MAY be calculated by merging the policies in the \{policy\} properties of:

- The Interface Message Reference component that describes the input message and
- The Binding Message Reference component whose \{interface message reference\} property has the Interface Message Reference component.

If the Binding component does not have an Interface component in the \{interface\} property, then the effective policy of an input message MUST be calculated in relation to a specific endpoint, and is the policy in the \{policy\} property of the Interface Message Reference component that describes the input message.

5.4.5 Message Policy Subject (output message)

If the Binding component has an Interface component in the \{interface\} property, then the effective policy of an output message MAY be calculated by merging the policies in the \{policy\} properties of:

- The Interface Message Reference component that describes the output message and
- The Binding Message Reference component whose \{interface message reference\} property has the Interface Message Reference component.

If the Binding component does not have an Interface component in the \{interface\} property, then the effective policy of an output message MUST be calculated in relation to a specific endpoint, and is the policy in the \{policy\} property of the Interface Message Reference component that describes the output message.

5.4.6 Message Policy Subject (input fault message)

If the Binding component has an Interface component in the \{interface\} property, then the effective policy of an input fault message MAY be calculated by merging the policies in the \{policy\} properties of:

- The Interface Fault Reference component that describes the input fault message,
- The Interface Fault component in the Interface Fault Reference component’s \{interface fault\} property,
- The Binding Fault Reference component whose \{interface fault reference\} property has the Interface Fault Reference component and
• The Binding Fault component whose {interface fault} property has the Interface Fault component in the Interface Fault Reference component’s {interface fault} property.

If the Binding component does not have an Interface component in the {interface} property, then the effective policy of an input fault message MUST be calculated in relation to a specific endpoint, and is the merge of policies in the {policy} properties of:

• The Interface Fault Reference component that describes the input fault message and

• The Interface Fault component in the Interface Fault Reference component’s {interface fault} property.

5.4.7 Message Policy Subject (output fault message)

If the Binding component has an Interface component in the {interface} property, then the effective policy of an output fault message MAY be calculated by merging the policies in the {policy} properties of:

• The Interface Fault Reference component that describes the output fault message,

• The Interface Fault component in the Interface Fault Reference component’s {interface fault} property,

• The Binding Fault Reference component whose {interface fault reference} property has the Interface Fault Reference component and

• The Binding Fault component whose {interface fault} property has the Interface Fault component in the Interface Fault Reference component’s {interface fault} property for the endpoint.

If the Binding component does not have an Interface component in the {interface} property, then the effective policy of an output fault message MUST be calculated in relation to a specific endpoint, and is the merge of policies in the {policy} properties of:

• The Interface Fault Reference component that describes the output fault message and

• The Interface Fault component in the Interface Fault Reference component’s {interface fault} property.

6. Attaching Policies Using UDDI

This section defines a mechanism for associating policies with entities in a UDDI registry. The calculation of effective policy for UDDI entities is described in Section 6.1 Calculating Effective Policy and Element Policy in UDDI. While the general concept for associating policies with UDDI entities, which is specified in Sections 6.2 Referencing Remote Policy Expressions and 6.3 Registering Reusable Policy Expressions, is based on UDDI Version 2, the necessary changes with respect to UDDI Version 3 are explained in Section 6.4 Registering Policies in UDDI.
There are essentially two approaches for registering policies in UDDI. One approach is to directly reference remotely accessible policy expressions in UDDI entities, the other is to register policy expressions as distinct tModels and then reference these tModels in each UDDI entity that is using the policy expressions. While the former approach (see Section 6.2 Referencing Remote Policy Expressions) is expected to be used for policy expressions that are mainly unique for a given Web service, the latter approach (see Section 6.3 Registering Reusable Policy Expressions) is expected to be used for more modular and reusable policy expressions.

6.1 Calculating Effective Policy and Element Policy in UDDI

When attaching a policy to a UDDI entity a policy scope is implied for that attachment. The policy scope only contains the policy subjects associated with that entity, and not those associated with the children of that entity. This is the entity’s element policy.

Each policy assertion contained within a UDDI entity’s element policy should have the correct semantic such that the policy subject for that assertion is that UDDI entity. For example, assertions that describe behaviours regarding a service provider should only be contained within policies attached to a businessEntity structure.

For UDDI tModels that represent Web service types, the element policy is considered an intrinsic part of the tModel and applies to all uses of that tModel. In particular, it MUST be merged into the effective policy of every bindingTemplate that references that tModel.

Policies that apply to deployed Web services (bindingTemplates) are only considered in the effective policy of that deployed resource itself.

Each of these entities MAY have an element policy per Section 3. Policy Attachment. The remainder of this section defines how that element policy is interpreted to calculate the effective policy.

6.1.1 Service Provider Policy Subject

The following UDDI element is considered as the service provider policy subject:

- uddi:businessEntity

This element MAY have an element policy as per Section 3. Policy Attachment, and if present MUST be merged into the effective policy of the UDDI businessEntity Subject.

Policy attached to the service provider policy subject applies to behaviors or aspects of the service provider as a whole, irrespective of interactions over any particular service. This includes — but is not limited to — acting as a service consumer or a service provider in general.
6.1.2 Service Policy Subject

The following UDDI element is considered as the service policy subject:

- uddi:businessService

This element MAY have element policy as per Section 3. Policy Attachment, and if present MUST be merged into the effective policy of the UDDI businessService Subject.

Policy attached to the service policy subject applies to behaviors or aspects of the service as a whole, irrespective of interactions over any particular endpoint. This includes — but is not limited to — acting as a consumer or a provider of the service.

6.1.3 Endpoint Policy Subject

The following UDDI elements collectively describe an endpoint:

- uddi:bindingTemplate
- uddi:tModel

These elements MAY have element policy as per Section 3. Policy Attachment. The policy scope implied by each of these elements contains the endpoint policy subject representing the deployed endpoint.

An endpoint policy subject applies to behaviours associated with an entire endpoint of the service, irrespective of any message exchange made. This includes — but is not limited to — aspects of communicating with or instantiating the endpoint.

The effective policy for a UDDI endpoint includes the element policy of the uddi:bindingTemplate element that defines the endpoint merged with the element policy of those uddi:tModel elements that are referenced in contained uddi:tModelInstanceInfo elements.

6.2 Referencing Remote Policy Expressions

UDDI tModels provide a generic mechanism for associating arbitrary metadata with services and other entities in a UDDI registry. To properly integrate Web Services Policy into the UDDI model, Web Services Policy 1.5 - Attachment pre-defines one tModel that is used to associate a remotely accessible policy with an entity in a UDDI registry.

This new tModel is called the remote policy reference category system and is defined in Appendix B.1 [Remote Policy Reference Category System].

UDDI registries MUST use the (UDDI V2 [UDDI Data Structure 2.0]) tModelKey uuid:a27078e4-fd38-320a-806f-6749e84f8005 to uniquely identify this tModel so that UDDI registry users can expect the same behavior across different UDDI registries.
The tModel’s valid values are those IRIs that identify external [policy expressions][p.7]; that is, when referencing this category system in a categoryBag, the corresponding keyValue of the keyedReference is the IRI of the [policy expression][p.7].

Using the remote policy reference category system, one can then associate a [policy expression][p.7] with a businessEntity, a businessService, and a tModel using the entity’s categoryBag. For example, associating the [policy expression][p.7] that is identified by the IRI http://www.example.com/myservice/policy with a businessService is done as follows:

```xml
<businessService serviceKey="...">
  ...</businessService>
```

The tModelKey of the keyedReference MUST match the fixed tModelKey from the remote policy reference category system. The keyValue MUST be the IRI that identifies the [policy expression][p.7].

A different approach has to be taken to associate a [policy expression][p.7] with a bindingTemplate, since bindingTemplates do not contain a categoryBag in UDDI Version 2. Therefore, the bindingTemplate’s tModelInstanceInfo and instanceParms MUST be used as follows:

```xml
<bindingTemplate bindingKey="...">
  ...</bindingTemplate>
```

The tModelKey of the tModelInstanceInfo MUST match the fixed tModelKey from the remote policy reference category system as defined above. The instanceParms MUST be the IRI that identifies the [policy expression][p.7].
6.3 Registering Reusable Policy Expressions

In addition to using the approach outlined in the section above, publishers may register a specific policy expression in a UDDI registry as a distinct tModel. To properly categorize tModels as policy expressions, Web Services Policy 1.5 - Attachment pre-defines the Web Services Policy Types category system as a tModel. This tModel is defined in Appendix B.2 Web Services Policy Types Category System.

The following illustrates a tModel for the policy expression identified by the IRI http://www.example.com/myservice/policy.

```
(01) <tModel tModelKey="uuid:04cf..."> 
(02)   <name>...</name> 
(03)   <description xml:lang="EN">Policy Expression for example’s Web services</description> 
(04) </tModel>
```

The first keyedReference specifies that the tModel represents a policy expression — rather than only being associated with one — by using the Web Services Policy Types category system’s built-in category "policy", which is its single valid value. This is necessary in order to enable UDDI inquiries for policy expressions in general. The second keyedReference designates the policy expression the tModel represents by using the approach from the section above. This is necessary in order to enable UDDI inquiries for particular policy expressions based on their IRI.

Note that the policy expression IRI is also specified in the tModel’s overview URL to indicate that it is a resolvable URL to actually retrieve the policy expression.

Web Services Policy 1.5 - Attachment pre-defines another tModel that is used to associate such a pre-registered, locally available policy expressions with an entity in a UDDI registry.

This new tModel is called the local policy reference category system and is defined in Appendix B.3 Local Policy Reference Category System.

UDDI registries MUST use the tModelKey uuid:a27f7d45-ec90-31f7-a655-efe91433527c to uniquely identify this tModel so that UDDI registry users can expect the same behavior across different UDDI registries.
The local policy reference category system is based on tModelKeys. The valid values of this category system are those tModelKeys identifying tModels that

- exist in the same UDDI registry
- and are categorized as "policy" using the Web Services Policy Types category system.

That is, when referencing this category system in a category bag, the corresponding keyValue of the keyedReference is the tModelKey of the tModel that represents the policy expression [p.7].

Given the local policy reference category system, one can then associate a policy expression [p.7] tModel with a businessEntity, a businessService, and a tModel using the entity’s categoryBag. For example, associating the policy expression [p.7] tModel with the tModelKey "uuid:04cfa..." from above with a businessService is done as follows:

```xml
(01) <businessService serviceKey="..." />
(02) <name>...</name>
(03) <description>...</description>
(04) <bindingTemplates>...</bindingTemplates>
(05) <categoryBag>
(06)   <keyedReference
(07)     keyName="Policy Expression for example’s Web services"
(08)     keyValue="uuid:04cfa...""
(09)     tModelKey="uuid:a27f7d45-ec90-31f7-a655-efe91433527c" />
(10) </categoryBag>
(11) </businessService>
```

The tModelKey of the keyedReference MUST match the fixed tModelKey from the local policy reference category system. The keyValue MUST be the tModelKey of the policy expression [p.7] that is registered with the UDDI registry.

A different approach has to be taken to associate a policy expression [p.7] with a bindingTemplate, since bindingTemplates do not contain a categoryBag in UDDI Version 2. Therefore, the bindingTemplate’s tModelInstanceInfo and instanceParms MUST be used as follows:

```xml
(01) <bindingTemplate bindingKey="..." />
(02) <accessPoint>...</accessPoint>
(03) <tModelInstanceDetails>
(04)   <tModelInstanceInfo
(05)     tModelKey="uuid:a27f7d45-ec90-31f7-a655-efe91433527c" >
(06)     <instanceDetails>
(07)       <instanceParms>uuid:04cfa...</instanceParms>
(08)     </instanceDetails>
(09)   </tModelInstanceInfo>
(10) </tModelInstanceDetails>
(11) </bindingTemplate>
```

The tModelKey of the tModelInstanceInfo MUST match the fixed tModelKey from the local policy reference category system. The instanceParms MUST be the tModelKey of the policy expression [p.7] that is registered with the UDDI registry.
6.4 Registering Policies in UDDI Version 3

UDDI Version 3 provides a number of enhancements in the areas of modeling and entity keying. Special considerations for UDDI multi-version support are outlined in chapter 10 of UDDI. The changes with respect to the previous sections are as follows.

First, the tModelKeys of the pre-defined tModels are migrated to domain-based keys. The migration is unique since the Version 2 keys introduced in this specification are already programmatically derived from the Version 3 keys given below.

The tModelKey for the remote policy reference tModel changes from "uuid:a27078e4-fd38-320a-806f-6749e84f8005" to "uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03".

The tModelKey for the Web Services Policy Types tModel changes from "uuid:fa1d77dc-edf0-3a84-a99a-5972e434e993" to "uddi:schemas.xmlsoap.org:policytypes:2003_03".

The tModelKey for the local policy reference tModel changes from "uuid:a27f7d45-ec90-31f7-a655-efe91433527c" to "uddi:schemas.xmlsoap.org:localpolicyreference:2003_03".

Second, rather than putting policy expression references in a bindingTemplate’s tModelInstanceInfo, they are added to the bindingTemplate’s categoryBag, analogous to the mechanism described for other UDDI entities. For example, the example bindingTemplate from section 6.1 Calculating Effective Policy and Element Policy in UDDI would be changed as follows:

```
(01) <bindingTemplate bindingKey="..." >
(02)   <accessPoint>...
(03)   <tModelInstanceDetails>...
(04)   <categoryBag>
(05)     <keyedReference
(06)       keyName="Policy Expression for example’s Web services"
(07)       keyValue="http://www.example.com/myservice/policy"
(08)       tModelKey="uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03"
(09)     />
(10) </categoryBag>
(11) </bindingTemplate>
```

Third, inquiries for reusable policy expression tModels described in Section 6.3 Registering Reusable Policy Expressions and UDDI tModel entities that are associated with remote policy expression is enhanced by the wildcard mechanism for keyValues in keyedReferences. For example, searching for all policy expression tModels whose IRI starts with http://www.example.com/, the following find_tModel API call can be used:

```
(01) <find_tModel
   xmlns="urn:uddi-org:api_v3" >
(02)   <categoryBag>
(03)     <keyedReference
(04)       keyValue="http://www.example.com/
(05)       tModelKey="uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03"
```
Fourth, all UDDI entities may be digitally signed using XML digital signatures. Publishers who want to digitally sign their tModels or references in UDDI MUST use the Schema-centric canonicalization algorithm.

7. Security Considerations

It is RECOMMENDED that policy attachments be signed to prevent tampering. This also provides a mechanism for authenticating by determining if the signer has the right to "speak for" the scope containing the policy.

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.

A more complete discussion of security considerations can be found in the Security Considerations section of the Web Services Policy 1.5 - Framework document.

8. Conformance

8.1 External Policy Attachment Conformance

An element information item whose namespace name is "http://www.w3.org/2006/07/ws-policy” and whose local part is PolicyAttachment conforms to this specification if it is valid according to the XML Schema for that element as defined by this specification and additionally adheres to all the constraints contained in Section of this specification. Such a conformant element information item constitutes an external policy attachment.

8.2 WSDL 1.1 Attachment Conformance

A WSDL 1.1 description conforms to this specification when it incorporates one or more element policies and additionally adheres to all the constraints contained in section.

8.3 WSDL 2.0 Attachment Conformance

A WSDL 2.0 description conforms to this specification when it incorporates one or more element policies and additionally adheres to all the constraints contained in section.
A. References

A.1 Normative References


A.2 Other References

[WSDL 2.0 Core Language]

[WS-Security 2004]

[XML Information Set]

[XML Namespaces]

[XML Schema Datatypes]

[XML Schema Structures]

A.2 Other References

[MTOM]

[WS-MTOMPolicy]

[SCC14N]
Schema Centric XML Canonicalization Version 1.0 S. Aissi, A. Hately, and M. Hondo, Editors.

[WS-Addressing Policy]

[WS-MetadataExchange]

[WS-RM Policy]

[WS-SecurityPolicy]

[WS-DL 1.1 Binding for SOAP 1.2]

[XML-Signature]

[XOP]

[XPointer Framework]
B. UDDI tModel Definitions

This section contains the UDDI tModel definitions for the canonical tModels used in Section 6. Attaching Policies Using UDDI [p.32]. The tModelKeys shown in the tModel structure sections are valid UDDI Version 2 keys. When using UDDI Version 3, the corresponding derived UDDI Version 2 keys must be used.

B.1 Remote Policy Reference Category System

B.1.1 Design Goals

This tModel is used to attach a policy [p.7] to a UDDI entity by referencing the policy’s IRI.

B.1.2 tModel Definition

<table>
<thead>
<tr>
<th>Name:</th>
<th><a href="http://schemas.xmlsoap.org/ws/2003/03/remotepolicyreference">http://schemas.xmlsoap.org/ws/2003/03/remotepolicyreference</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>Category system used for UDDI entities to point to an external Web services policy attachment policy that describes their characteristics. See Web Services Policy 1.5 - Attachment specification for further details.</td>
</tr>
<tr>
<td>UDDI Key (V3):</td>
<td>uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03</td>
</tr>
<tr>
<td>UDDI V1,V2 format key:</td>
<td>uuid:a27078e4-fd38-320a-806f-6749e84f8005</td>
</tr>
<tr>
<td>Categorization:</td>
<td>categorization</td>
</tr>
<tr>
<td>Checked:</td>
<td>No</td>
</tr>
</tbody>
</table>

B.1.3 tModel Structure

```
(01) <tModel tModelKey="uuid:a27078e4-fd38-320a-806f-6749e84f8005" >
(02)  <name>http://schemas.xmlsoap.org/ws/2003/03/remotepolicyreference</name>
(03)  <description xml:lang="EN">Category system used for UDDI entities to point to an external Web Services Policy Attachment policy expression that describes their characteristics. See Web Services Policy 1.5 - Attachment specification for further details.</description>
(04)  <categoryBag>
(05)    <keyedReference
(06)      keyValue="categorization"
(07)      keyName="uddi-org:types:categorization"
(08)    </keyedReference>
(09)  </categoryBag>
(10) </tModel>
```
B.2 Web Services Policy Types Category System

B.2.1 Design Goals

This tModel is used to categorize tModels as representing policy expressions. There is only one valid value, namely "policy", that indicates this very fact. It is RECOMMENDED that tModels categorized as representing policy expressions reference no more and no less than this very policy expression using the remote policy reference category system.

B.2.2 tModel Definition

<table>
<thead>
<tr>
<th>Name:</th>
<th><a href="http://schemas.xmlsoap.org/ws/2003/03/policytypes">http://schemas.xmlsoap.org/ws/2003/03/policytypes</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>Web services policy types category system used for UDDI tModels to characterize them as Web services policy–based policy expressions.</td>
</tr>
<tr>
<td>UDDI Key (V3):</td>
<td>uddi:schemas.xmlsoap.org:policytypes:2003_03</td>
</tr>
<tr>
<td>UDDI V1,V2 format key:</td>
<td>uuid:fa1d77dc-edf0-3a84-a99a-5972e434e993</td>
</tr>
<tr>
<td>Categorization:</td>
<td>categorization</td>
</tr>
<tr>
<td>Checked:</td>
<td>No</td>
</tr>
</tbody>
</table>

B.2.3 tModel Structure

(01) <tModel tModelKey="uuid:fa1d77dc-edf0-3a84-a99a-5972e434e993" >
(02)  <name>http://schemas.xmlsoap.org/ws/2003/03/policytypes</name>
(03)  <description xml:lang="EN">Web Services Policy Types category system used for UDDI tModels to characterize them as Web Services Policy–based policy expressions.</description>
(04)  <categoryBag>
(05)    <keyedReference
(06)      keyName="uddi-org:types:categorization"
(07)      keyValue="categorization"
(08)    </keyedReference>
(09)  </categoryBag>
(10) </tModel>

B.3 Local Policy Reference Category System

B.3.1 Design Goals

This tModel is used to attach a policy expression to a UDDI entity by referencing the UDDI entity that represents this policy expression. The local policy reference category system is based on tModelKeys. It is expected that referenced tModels are registered with the same UDDI registry and are categorized as representing policy expressions using the Web services policy types category system.
B.3.2 tModel Definition

<table>
<thead>
<tr>
<th>Name:</th>
<th><a href="http://schemas.xmlsoap.org/ws/2003/03/localpolicyreference">http://schemas.xmlsoap.org/ws/2003/03/localpolicyreference</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>Category system used for UDDI entities to point to a Web Services Policy expression tModel that describes their characteristics. See Web Services Policy 1.5 - Attachment specification for further details.</td>
</tr>
<tr>
<td>UDDI Key (V3):</td>
<td>uddi:schemas.xmlsoap.org:localpolicyreference:2003_03</td>
</tr>
<tr>
<td>UDDI V1,V2 format key:</td>
<td>uuid:a27f7d45-ec90-31f7-a655-efe91433527c</td>
</tr>
<tr>
<td>Categorization:</td>
<td>categorization</td>
</tr>
<tr>
<td>Checked:</td>
<td>Yes</td>
</tr>
</tbody>
</table>

B.3.3 tModel Structure

```xml
(01) <tModel tModelKey="uuid:a27f7d45-ec90-31f7-a655-efe91433527c" >
(02)   <name>http://schemas.xmlsoap.org/ws/2003/03/localpolicyreference</name>
(03)   <description xml:lang="en">Category system used for UDDI entities to point to a Web Services Policy expression tModel that describes their characteristics. See Web Services Policy 1.5 - Attachment specification for further details.</description>
(04)   <categoryBag>
(05)     <keyedReference
(06)       keyName="uddi-org:types:categorization"
(07)       keyValue="categorization"
(08)       tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4" />
(09)     <keyedReference
(10)       keyName="uddi-org:types:checked"
(11)       keyValue="checked"
(12)       tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4" />
(13)     <keyedReference
(14)       keyName="uddi-org:entityKeyValues"
(15)       keyValue="tModelKey"
(16)       tModelKey="uuid:916b87bf-0756-3919-8eae-97d8a329e5a4" />
(17)   </categoryBag>
(18) </tModel>
```

C. Acknowledgements (Non-Normative)

This document is the work of the [W3C Web Services Policy Working Group](http://www.w3.org/2002/06/Policy/)

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 Boubeze (Layer 7 Technologies), Paul Cotton (Microsoft Corporation), Jeffrey Crump (Sonic Software), Glen Daniels (Sonic Software), Jacques Durand (Fujitsu Limited), Ruchith Fernando (WSO2), Christopher Ferris (IBM Corporation), William Henry (IONA Technologies, Inc.), Frederick Hirsch (Nokia), Maryann Hondo (IBM Corporation), Tom Jordahl (Adobe Systems Inc.), Philippe Le Hégaret (W3C/MIT), Jong Lee (BEA Systems, Inc.), Mark
Little (JBoss Inc.), Ashok Malhotra (Oracle Corporation), Monica Martin (Sun Microsystems, Inc.), Jeff Mischkinsky (Oracle Corporation), Dale Moberg (Cyclone Commerce, Inc.), Anthony Nadalin (IBM Corporation), David Orchard (BEA Systems, Inc.), Fabian Ritzmann (Sun Microsystems, Inc.), Daniel Roth (Microsoft Corporation), Tom Rutt (Fujitsu Limited), Sanka Samaranayake (WSO2), Felix Sasaki (W3C/Keio), Skip Snow (Citigroup), Yakov Sverdlov (Computer Associates), Mark Temple-Raston (Citigroup), Asir Vedamuthu (Microsoft Corporation), Sanjiva Weerawarana (WSO2), Ümit Yalçinalp (SAP AG), Prasad Yendluri (webMethods, Inc.).

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

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

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

A list of substantive changes since the Working Draft dated 02 November, 2006 is below:

- Added a new domain expression: [3.4.1 URI Domain Expression][p.13].
- Defined the equivalence of two {policy} properties in Section 5.3 Extension to WSDL Component Model [p.26].

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20060712</td>
<td>ASV</td>
<td>Updated the list of editors. Completed action items from the Austin F2F.</td>
</tr>
<tr>
<td>20060712</td>
<td>DBO</td>
<td>Completed action item</td>
</tr>
<tr>
<td>20060718</td>
<td>DBO</td>
<td>Completed action items Editors to remove extraneous namespace decl in the example at the end of section 3.4 RFC2606 for domain names (note: PLH had already done but it didn’t show up in the change log) editors to straighten up Note after example 3-1</td>
</tr>
<tr>
<td>20060719</td>
<td>TIB</td>
<td>Completed action item 22: Linked SVG graphic</td>
</tr>
<tr>
<td>20060721</td>
<td>ASV</td>
<td>Completed action items 23, 25 and 26 from the Austin F2F.</td>
</tr>
<tr>
<td>20060721</td>
<td>ASV</td>
<td>Completed action item 29 from the Austin F2F.</td>
</tr>
<tr>
<td>20060726</td>
<td>ASV</td>
<td>Incorporated the XML namespace URI versioning policy adopted by the WG.</td>
</tr>
<tr>
<td>20060808</td>
<td>DBO</td>
<td>Completed action items: as early as possible in the doc, use the definition that are defined in the doc. Issue 3545 use of {any} and {@any} in xpath-like expressions not defined in Notational Conventions section</td>
</tr>
<tr>
<td>Date</td>
<td>Entity</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>20060808</td>
<td>ASV</td>
<td>Implemented the resolution for issue 3543</td>
</tr>
<tr>
<td>20060809</td>
<td>DBO</td>
<td>Implemented the resolution for issue 3546 wsdl 2.0 status mention.</td>
</tr>
<tr>
<td>20060809</td>
<td>ASV</td>
<td>Implemented the resolution for issue 3556 and the resolution for issue 3558.</td>
</tr>
<tr>
<td>20060811</td>
<td>DBO</td>
<td>Completed action items: remove use if emph/ital terms. Framework: removed emph on conceptually replace and support; attachment: make merge a termdef</td>
</tr>
<tr>
<td>20060813</td>
<td>ASV</td>
<td>Added a new Section [D. Changes in this Version of the Document][p.46] (that provides a list of substantive changes since the previous publication).</td>
</tr>
<tr>
<td>20060825</td>
<td>PY</td>
<td>Implemented the resolution for issue 3544.</td>
</tr>
<tr>
<td>20060827</td>
<td>TIB</td>
<td>Completed action item: resolution for adding Conformance section.</td>
</tr>
<tr>
<td>20060827</td>
<td>TIB</td>
<td>Implemented the resolution for issue 3605 typo in example.</td>
</tr>
<tr>
<td>20060829</td>
<td>ASV</td>
<td>Implemented the resolution for issue 3561 replaced URI with IRI.</td>
</tr>
<tr>
<td>20060906</td>
<td>ASV</td>
<td>Implemented the resolution for issue 3557 to clarify the use of domain expressions.</td>
</tr>
<tr>
<td>20060924</td>
<td>TIB</td>
<td>Implemented the resolution for issue 3709 Editorial corrections from UDDI team.</td>
</tr>
<tr>
<td>20060925</td>
<td>ASV</td>
<td>Created [5. WS-Policy Attachment for WSDL 2.0][p.21] per action item 2 from the Bellevue F2F. This section is based on a contribution from BEA, IBM, Microsoft and Oracle.</td>
</tr>
<tr>
<td>20061002</td>
<td>DBO</td>
<td>Implemented the for issue 3559 Conformance Section.</td>
</tr>
<tr>
<td>20061002</td>
<td>DBO</td>
<td>Implemented the resolution for issue 3712 wsp:PolicyReference can be used in any place where you can use wsp:Policy.</td>
</tr>
<tr>
<td>20061008</td>
<td>MH</td>
<td>Implemented the resolution for issue summary of Action 10 from F2F replaced &quot;subject&quot; instances with &quot;policy subject&quot;.</td>
</tr>
<tr>
<td>20061012</td>
<td>PY</td>
<td>Updated &quot;Changes in this Version&quot; section (Appendix D)</td>
</tr>
<tr>
<td>20061019</td>
<td>PY</td>
<td>Completed action item: Paul’s comments.</td>
</tr>
<tr>
<td>20061027</td>
<td>FS</td>
<td>Changed the SVG link and re-formatted the examples in [B. UDDI tModel Definitions][p.43] (these changes were necessary for PDF creation).</td>
</tr>
<tr>
<td>20061030</td>
<td>DBO</td>
<td>Implemented the resolution titled &quot;Hyperlink terms such as policy expression...&quot;</td>
</tr>
<tr>
<td>20061031</td>
<td>TIB</td>
<td>Completed action item: to add a pointer to the Security Considerations section of the framwork document. This is a follow-up to closed issue 110</td>
</tr>
<tr>
<td>Date</td>
<td>Name</td>
<td>Action Description</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>--------------------</td>
</tr>
<tr>
<td>20061102</td>
<td>ASV</td>
<td>Implemented the resolution for Editors’ Action [12]</td>
</tr>
<tr>
<td>20061102</td>
<td>ASV</td>
<td>Reset Section D. Changes in this Version of the Document [p.46]</td>
</tr>
<tr>
<td>20061109</td>
<td>PY</td>
<td>Implemented the resolution for Issue 3894, Editors’ Action [71]</td>
</tr>
<tr>
<td>20061109</td>
<td>TIB</td>
<td>Implemented Editors’ Action [74]</td>
</tr>
<tr>
<td>20061109</td>
<td>ASV</td>
<td>Implemented the resolution for issues 3721 and 3789</td>
</tr>
<tr>
<td>20061109</td>
<td>ASV</td>
<td>Implemented the resolution for issues 3599 and 3730</td>
</tr>
<tr>
<td>20061109</td>
<td>ASV</td>
<td>Updated Section D. Changes in this Version of the Document [p.46]</td>
</tr>
</tbody>
</table>