Web Services Policy 1.5 - Attachment

W3C Recommendation 04 September 2007

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

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 the W3C Recommendation of the Web Services Policy 1.5 - Attachment 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

Appendices

---

1. Introduction

The Web Services Policy 1.5 - Framework [*Web Services Policy Framework [p.42]*] specification defines an abstract model and an XML-based language for expressing policies [p.8] 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 [p.8] 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.43]* , *WSDL 2.0 Core Language [p.43]*] definitions and UDDI [*UDDI API 2.0 [p.42]* , *UDDI Data Structure 2.0 [p.42]* , *UDDI 3.0 [p.42]*].

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.43]*]. 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/ns/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. namespace.

Normative text within this specification takes precedence over normative outlines, which in turn take precedence over the XML Schema [*XML Schema Structures [p.43]*] 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.43]*]).

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

Prefix	XML Namespace	Specification
`mtom`	`http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization`	[*WS-MTOMPolicy [p.44]*]
`rmp`	`http://docs.oasis-open.org/ws-rx/wsrmp/200702`	[*WS-RM Policy [p.44]*]
`sp`	`http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702`	[*WS-SecurityPolicy [p.44]*]
`wsa`	`http://www.w3.org/2005/08/addressing`	[*WS-Addressing Core [p.43]*]
`wsam`	`http://www.w3.org/2007/05/addressing/metadata`	[*WS-Addressing Metadata [p.44]*]
`wsdl11`	`http://schemas.xmlsoap.org/wsdl/`	[*WSDL 1.1 [p.43]*]
`wsdl20`	`http://www.w3.org/ns/wsdl`	[*WSDL 2.0 Core Language [p.43]*]
`wsoap12`	`http://schemas.xmlsoap.org/wsdl/soap12/`	[*WSDL 1.1 Binding for SOAP 1.2 [p.45]*]
`(none), wsp`	`http://www.w3.org/ns/ws-policy`	This specification
`wsse`	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd`	[*WS-Security 2004 [p.43]*]
`wsu`	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd`	[*WS-Security 2004 [p.43]*]
`xs`	`http://www.w3.org/2001/XMLSchema`	[*XML Schema Structures [p.43]*]

All information items defined by this specification are identified by the XML namespace URI [*XML Namespaces [p.43]*] http://www.w3.org/ns/ws-policy. A normative XML Schema [*XML Schema Structures [p.43]* , *XML Schema Datatypes [p.43]*] document can be obtained indirectly by dereferencing the namespace document at the WS-Policy 1.5 namespace URI.

In this document reference is made to the `wsu:Id` attribute in a utility schema (http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd). The `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 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 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 [p.42]*].

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

collection

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

effective policy [p.10]

> The **effective policy**, for a given policy subject [p.8] , is the combination of relevant policies. The relevant policies are those attached to policy scopes [p.8] that contain the policy subject [p.8] .

element policy [p.11]

The **element policy** is the policy [p.8] attached to the policy subjects [p.8] associated with the element information item that contains it.

ignorable policy assertion

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).

merge [p.10]

A **merge** consists of serializing each policy as a policy expression [p.8] , 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 [p.8] .

policy alternative

A **policy alternative** is a potentially empty collection [p.7] of policy assertions [p.8] .

policy assertion

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

policy attachment

A **policy attachment** is a mechanism for associating policy [p.8] with one or more policy scopes [p.8] .

policy expression

A **policy expression** is an XML Infoset representation of a policy [p.8] , either in a normal form or in an equivalent compact form.

policy scope

A **policy scope** is a collection of policy subjects [p.8] 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 [p.8] 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.42]*]) with various XML Web service entities. For brevity, we define two sample policy expressions [p.8] that the remainder of this document references.

Example 2-1 [p.9] indicates a policy [p.8] for reliable messaging [*WS-RM Policy [p.44]*]. Example 2-2 [p.9] is a policy for securing messages using X509 certificates [*WS-SecurityPolicy [p.44]*].

Example 2-1. Example RM Policy Expression.

```
<wsp:Policy
       xmlns:rmp="http://docs.oasis-open.org/ws-rx/wsrmp/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="RmPolicy" >
  <rmp:RMAssertion>
    <wsp:Policy/>
  </rmp:RMAssertion>
</wsp:Policy>
```

Example 2-2. Example X509 Security Policy Expression.

```
<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="X509EndpointPolicy" >
  <sp:AsymmetricBinding>
    <wsp:Policy>
      <sp:RecipientToken>
        <wsp:Policy>
          <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
            <wsp:Policy>
              <sp:WssX509V3Token10 />
            </wsp:Policy>
          </sp:X509Token>
        </wsp:Policy>
      </sp:RecipientToken>
      <sp:InitiatorToken>
        <wsp:Policy>
          <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient" >
            <wsp:Policy>
              <sp:WssX509V3Token10 />
            </wsp:Policy>
          </sp:X509Token>
        </wsp:Policy>
      </sp:InitiatorToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic256Rsa15 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Lax />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <sp:OnlySignEntireHeadersAndBody />
    </wsp:Policy>
  </sp:AsymmetricBinding>
</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.42]*], the IRIs used for these policy expressions [p.8] 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.9] and Example 2-2 [p.9] , respectively.

3. Policy Attachment

This section defines two general-purpose mechanisms for associating policies [p.8] with one or more policy subjects [p.8] . 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.8] will often be associated with a particular policy subject [p.8] using multiple policy attachments [p.8] . 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, their relevent policies can be combined. [Definition: The **effective policy**, for a given policy subject [p.8] , is the combination of relevant policies. The relevant policies are those attached to policy scopes [p.8] that contain the policy subject [p.8] .]

This combination can be achieved through a merge. [Definition: A **merge** consists of serializing each policy as a policy expression [p.8] , 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.42]*] with one or more policy subjects [p.8] . The first allows XML-based descriptions of resources to associate policy [p.8] as part of their intrinsic definition. The second allows policies to be associated with arbitrary policy subjects independently from their definition.

The policy scope [p.8] of an attachment is specific to the policy attachment [p.8] Mechanism using it. Accordingly, any policy attachment [p.8] mechanism MUST define the policy scope [p.8] of the attachment.

3.3 XML Element Attachment

It is often desirable to associate policies [p.8] with Web services policy subjects [p.8] represented as XML elements (i.e., WSDL 1.1 elements - Section **4. Attaching Policies Using WSDL 1.1** [p.15] for the specific details of WSDL attachment).

Since policy assertions [p.8] 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 [p.43]*] and Policy.

This specification defines a global attribute that allows policy expressions [p.8] to be attached to an arbitrary XML element. The following is the schema definition for the `wsp:PolicyURIs` attribute:

```
<xs:schema>
  <xs:attribute name="PolicyURIs">
    <xs:simpleType>
   <xs:list itemType="xs:anyURI" />
    </xs:simpleType>
  </xs:attribute>
</xs:schema>
```

The namespace URI [*XML Namespaces [p.43]*] for this attribute is `http://www.w3.org/ns/ws-policy`.

The `wsp:PolicyURIs` attribute contains a white space-separated list of one or more IRIs [*IETF RFC 3987 [p.42]*]. When this attribute is used, each of the values identifies a policy expression [p.8] as defined by [*Web Services Policy Framework [p.42]*]. If more than one IRI is specified, the individual referenced policies [p.8] need to be merged [p.10] together to form a single element policy expression [p.8] . The resultant policy [p.8] is then associated with the element information item’s element policy [p.11] property. [Definition: The **element policy** is the policy [p.8] attached to the policy subjects [p.8] associated with the element information item that contains it.]

An example of element policy [p.11] through the use of this global attribute is given below using the sample policies stated in Section **2.4 Example** [p.9] .

If the policies [p.8] referenced by the following XML element

```
<MyElement wsp:PolicyURIs="
   http://www.example.com/policies#RmPolicy
   http://www.example.com/policies#X509EndpointPolicy" />
```

have been processed and merged [p.10] , it would result in an element policy [p.11] whose XML 1.0 representation is listed in Example 3-1 [p.11] :

Example 3-1. Example Merged Policy Expression.

```
<wsp:Policy
       xmlns:rmp="http://docs.oasis-open.org/ws-rx/wsrmp/200702"
       xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
       xmlns:wsp="http://www.w3.org/ns/ws-policy" >
  <wsp:All>
    <rmp:RMAssertion>
       <wsp:Policy/>
    </rmp:RMAssertion>
  </wsp:All>
  <wsp:All>
    <sp:AsymmetricBinding>
      <wsp:Policy>
```

```
        <!-- Details omitted for readability -->
        <sp:IncludeTimestamp />
        <sp:OnlySignEntireHeadersAndBody />
      </wsp:Policy>
    </sp:AsymmetricBinding>
  </wsp:All>
</wsp:Policy>
```

Note that this element policy [p.11] has no meaningful IRI.

The presence of the `wsp:PolicyURIs` attribute does not prohibit implementations from using additional mechanisms for associating policy expressions [p.8] with XML-based constructs.

Alternatively, rather than using the global attribute, XML elements MAY use the `wsp:Policy` or `wsp:PolicyReference` elements directly as children, in order to support element policy [p.11] (Per Section 4.3.4 Policy References of Web Services Policy 1.5 - Framework [*Web Services Policy Framework [p.42]*]), and the semantics for this are the same as for the use of the global attribute. For example, an alternative way of attaching the policies in the above example, using child elements, would be as follows:

```
<MyElement>
  <wsp:PolicyReference
     URI="http://www.example.com/policies#RmPolicy" />
  <wsp:PolicyReference
     URI="http://www.example.com/policies#X509EndpointPolicy" />
<MyElement/>
```

3.4 External Policy Attachment

This mechanism allows policies [p.8] to be associated with a policy subject [p.8] 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 [p.8] of the attachment, the policy expressions [p.8] being bound, and optional security information. The policy scope [p.8] of the attachment is defined using one or more extensible domain expressions that identify policy subjects [p.8] , typically using IRIs.

Domain expressions identify the domain of the association. That is, the set of policy subjects [p.8] that will be considered for inclusion in the scope using an extensible domain expression model. Domain expressions identify policy subjects [p.8] to be included within the policy scope [p.8] . Domain expressions yield an unordered set of policy subjects [p.8] for consideration.

For the purposes of attaching policy [p.8] to a policy subject [p.8] through this mechanism, any policy expression [p.8] 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 [p.8] within it, but this policy expressions [p.8] 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:

```
<wsp:PolicyAttachment … >
  <wsp:AppliesTo>
    <x:DomainExpression/> +
  </wsp:AppliesTo>
  ( <wsp:Policy>…</wsp:Policy> |
    <wsp:PolicyReference>…</wsp:PolicyReference> ) +
  <wsse:Security>…</wsse:Security> ?
  …
</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.8] .

`/wsp:PolicyAttachment/wsp:AppliesTo`

This required element’s children describe the policy scope [p.8] .

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

These child elements MUST specify and/or refine the domain expression(s) that define the policy scope [p.8] . 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.8] within a policy scope [p.8] . When more than one domain expression is present, the policy scope [p.8] contains the union of the policy subjects [p.8] identified by each expression.

`/wsp:PolicyAttachment/wsp:Policy`

This element is a policy expression [p.8] representing a policy [p.8] that is attached to the policy subjects [p.8] within the policy scope [p.8] .

`/wsp:PolicyAttachment/wsp:PolicyReference`

This element references a policy expression [p.8] to be attached to the policy subjects [p.8] that are in the policy scope [p.8] . Refer to Web Services Policy 1.5 - Framework [*Web Services Policy Framework [p.42]*] for additional details.

`/wsp:PolicyAttachment/wsse:Security`

This element is of type `wsse:Security` and allows security information such as signatures to be included. The syntax of this element is described in WS-Security [*WS-Security 2004 [p.43]*].

`/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.

`/wsp:PolicyAttachment/{any}`

> 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.43]*]:

```
<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsa:EndpointReference>
      <wsa:Address>http://www.example.com/acct</wsa:Address>
    </wsa:EndpointReference>
  </wsp:AppliesTo>
  <wsp:PolicyReference
     URI="http://www.example.com/policies#RmPolicy" />
</wsp:PolicyAttachment>
```

In this example, the policy expression [p.8] 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.8] for the external attachment mechanism. The following is a pseudo-schema for the URI domain expression:

```
<wsp:PolicyAttachment … >
  <wsp:AppliesTo>
    <wsp:URI … >xs:anyURI</wsp:URI> *
  </wsp:AppliesTo>
  …
</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 IRI that references a resource as a policy subject [p.8] . 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 [p.8] can be associated. For example, URI domain expressions can be used to identify WSDL 1.1 definitions, WSDL 2.0 components, etc. When a URI domain expression identifies multiple resources, i.e. WSDL 1.1 supports multiple operations with the same name (sometimes called operation name overloading), the Policy applies to all the resources that are identified.

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 [*WSDL 2.0 Core Language [p.43]*]. The following example illustrates the use of URI domain expression with a WSDL 2.0 IRI Reference:

```
<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 [p.8] at `http://www.example.com/policies#RmPolicy` applies to all interactions with the endpoint at `http://example.org/TicketAgent.wsdl20#wsdl.endpoint(TicketAgentService/Endpoint)`.

IRI References for WSDL 1.1 elements are defined in WSDL 1.1 Element Identifiers [*WSDL11 ElementIds [p.44]*].

The scope of URI domain expressions for WSDL 2.0 components or WSDL 1.1 elements is limited to the subjects defined in Section **5. WS-Policy Attachment for WSDL 2.0** [p.22] and **4. Attaching Policies Using WSDL 1.1** [p.15] .

3.5 Use of IRIs in Policy Attachment

Policy attachment mechanisms 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.43]*] and RFCs 3023 [*IETF RFC 3023 [p.42]*], 3986 [*IETF RFC 3986 [p.42]*] and 3987 [*IETF RFC 3987 [p.42]*] for establishing a base URI against which relative IRIs can be made absolute.

4. Attaching Policies Using WSDL 1.1

This section describes a mechanism for associating policy expressions with Web service constructs in WSDL 1.1 [*WSDL 1.1 [p.43]*]. The mechanism consists of:

- A model for attaching policies to WSDL 1.1 constructs. The model defines:
    - A partitioning of WSDL constructs into service, endpoint, operation and message policy subjects.
    - The semantics of attaching a policy [p.8] to each policy subject [p.8]

    - How to combine policies attached to more than one WSDL construct within a single policy subject.

- An XML representation of policy expressions attached to WSDL 1.1 constructs.
- The annotation of such policy expressions as required extensions using the WSDL-defined extensibility flag `@wsdl:required`.

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 [*BP 1.1 [p.42]*] overrules this restriction and allows element extensibility everywhere. Therefore, when attaching a reference directly to the WSDL element 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 [p.8] 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 [p.8] MAY be made available through some other means, such as WS-MetadataExchange [*WS-MetadataExchange [p.44]*].

To ensure that consumers of policy-annotated WSDL elements are capable of processing such policy attachments [p.8] , 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 [p.8] when they appear within a WSDL description.

4.1 Calculating Effective Policy in WSDL 1.1

Policy attachments [p.8] in WSDL 1.1 can be used to associate policies [p.8] with four different types of policy subject [p.8] , 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 [p.8] to a WSDL element, a policy scope [p.8] is implied for that attachment. The policy scope [p.8] only contains the policy subject [p.8] associated with that element and not those associated with the children of that element. Therefore, it is RECOMMENDED that each policy assertion [p.8] contained within a WSDL element's element policy [p.11] 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 [p.10] , with regard to WSDL, are calculated for each of these policy subjects [p.8] . In the diagram, the dashed boxes represent policy scope [p.8] s implied by WSDL elements. For a particular policy subject [p.8] , the effective policy [p.10] MUST *merge* the element policy [p.11] of each element with a policy scope [p.8] that contains the policy subject [p.8] .

For abstract WSDL definitions, the element policy [p.11] is considered an intrinsic part of the definition and applies to all uses of that definition. In particular, it MUST be merged [p.10] into the effective policy [p.10] 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 [p.10] of that deployed resource itself.

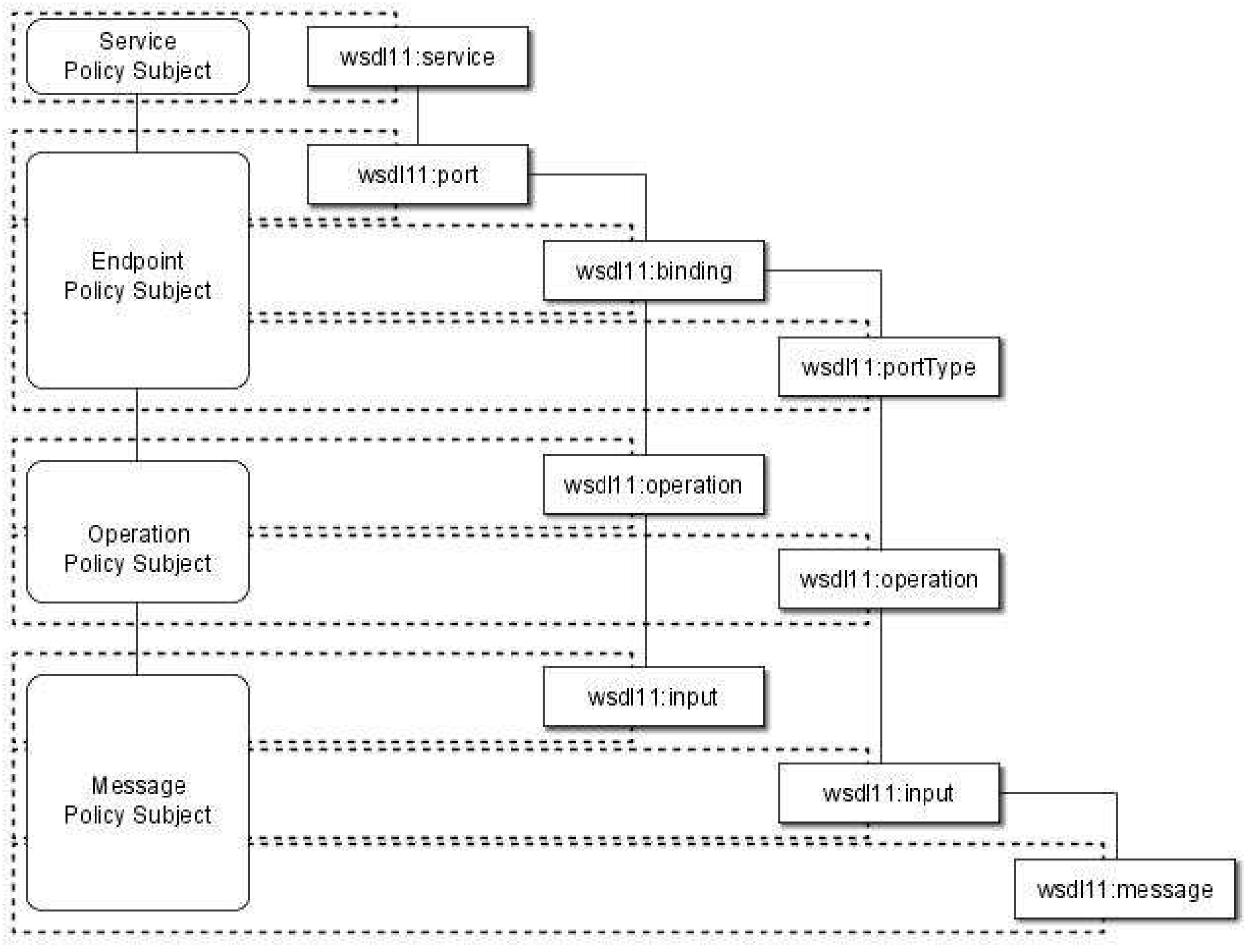

Figure 4-1. Effective Policy and Policy Scopes in WSDL

(This graphic is also available in SVG format here.)

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 [p.10] in all four subject types simultaneously.

For example, in Figure 4-1 [p.17] , for a particular input message to a deployed endpoint, there are four policy subjects [p.8] involved, each with their own effective policy [p.10] . There is an effective policy [p.10] for the message, as well as an effective policy [p.10] for the parent operation of that message, an effective policy [p.10] for the deployed endpoint, and the effective policy [p.10] for the service as a whole. All four effective policies [p.10] are applicable in relation to that specific input message.

It is RECOMMENDED that, where specific policy assertions [p.8] associated with one policy subject [p.8] are only compatible with specific policy assertions [p.8] on another policy subject [p.8] 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 [p.8] for each policy subject [p.8] type SHOULD be compatible with each of the policy alternatives [p.8] at each of the policy subjects [p.8] parent and child policy subjects [p.8] , such that choices between policy alternatives [p.8] at each level are independent of each other.

The rest of this section describes these policy subject [p.8] types, and how the effective policy [p.10] for each policy subject [p.8] 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 [p.11] as per Section **3. Policy Attachment** [p.10] , and if present MUST be merged [p.10] into the effective policy [p.10] 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 [p.11] as per Section **3. Policy Attachment** [p.10] . The policy scope [p.8] 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 [p.10] for a WSDL endpoint policy subject includes the element policy [p.11] of the `wsdl11:port` element that defines the endpoint merged [p.10] with the element policy [p.11] of the referenced `wsdl11:binding` element and the element policy [p.11] of the referenced `wsdl11:portType` 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.11] as per Section **3. Policy Attachment** [p.10] .

The policy scope [p.8] 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.

The effective policy [p.10] for a WSDL operation policy subject is calculated in relation to a specific port, and includes the element policy [p.11] of the `wsdl11:portType/wsdl11:operation` element that defines the operation merged [p.10] with that of the corresponding `wsdl11:binding/wsdl11:operation` element.

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.11] as per Section **3. Policy Attachment** [p.10] .

The policy scope [p.8] 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 [p.10] 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 [p.11] of the `wsdl11:message` element that defines the message's type merged [p.10] with the element policy [p.11] of the `wsdl11:binding` and `wsdl11:portType` message definitions that describe that message.

For example, the effective policy [p.10] of a specific input message for a specific port would be the *merge* of the `wsdl11:message` element defining the message type, the `wsdl11:portType/wsdl11:operation/wsdl11:input` element, and the corresponding `wsdl11:binding/wsdl11:operation/wsdl11:input` element for that message.

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

Since `wsdl11:input`, `wsdl11:output`, and `wsdl11:fault` elements in a `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 [p.8] on outbound messages SHOULD be avoided without the use of some form of mutual policy [p.8] exchange between the parties involved.

4.1.5 Example

As an example of the combination of these policy subjects [p.8] and effective policy [p.10] calculation, consider the WSDL type definition in Example 4-1 [p.21] that references policies.

Example 4-1. Example Policy Attached to WSDL.

```
<wsdl11:definitions name="StockQuote"
   targetNamespace="http://www.example.com/stock/binding"
   xmlns:tns="http://www.example.com/stock/binding"
   xmlns:fab="http://www.example.com/stock"
   xmlns:rmp="http://docs.oasis-open.org/ws-rx/wsrmp/200702"
   xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsoap12="http://schemas.xmlsoap.org/wsdl/soap12/"
   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" >
  <wsp:Policy wsu:Id="RmPolicy" >
    <rmp:RMAssertion>
      <wsp:Policy/>
    </rmp:RMAssertion>
  </wsp:Policy>
    <wsp:Policy wsu:Id="X509EndpointPolicy" >
      <sp:AsymmetricBinding>
        <wsp:Policy>
          <!-- Details omitted for readability -->
          <sp:IncludeTimestamp />
          <sp:OnlySignEntireHeadersAndBody />
        </wsp:Policy>
      </sp:AsymmetricBinding>
    </wsp:Policy>
    <wsp:Policy wsu:Id="SecureMessagePolicy" >
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
    </wsp:Policy>
    <wsdl11:import namespace="http://www.example.com/stock"
        location="http://www.example.com/stock/stock.wsdl" />
    <wsdl11:binding name="StockQuoteSoapBinding" type="fab:Quote" >
      <wsoap12:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http" />
      <wsp:PolicyReference URI="#RmPolicy" wsdl11:required="true" />
      <wsp:PolicyReference URI="#X509EndpointPolicy" wsdl11:required="true" />
      <wsdl11:operation name="GetLastTradePrice" >
        <wsoap12:operation soapAction="http://www.example.com/stock/Quote/GetLastTradePriceRequest" />
          <wsdl11:input>
            <wsoap12:body use="literal" />
            <wsp:PolicyReference URI="#SecureMessagePolicy"
                                 wsdl11:required="true" />
          </wsdl11:input>
          <wsdl11:output>
            <wsoap12:body use="literal" />
            <wsp:PolicyReference URI="#SecureMessagePolicy"
                                 wsdl11:required="true" />
          </wsdl11:output>
      </wsdl11:operation>
    </wsdl11:binding>
</wsdl11:definitions>
```

For endpoints bound to `StockQuoteSoapBinding`, the effective policy [p.10] of the endpoint is listed in Example 3-1 [p.11] (above). For the `GetLastTradePrice` operation, an additional message-level effective policy [p.10] is in effect for the input message, whose XML 1.0 representation is listed in

Example 4-2 [p.22] .

Example 4-2. Example Message Security Policy Expression.

```
<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="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.
- The annotation of such 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

```
<wsdl20:description>
…
  <wsp:Policy wsu:Id="common">
    <mtom:OptimizedMimeSerialization wsp:Optional="true"/>
    <wsam:Addressing>…</wsam:Addressing>
  </wsp:Policy>
  <wsp:Policy wsu:Id="secure">
    <wsp:ExactlyOne>
      <sp:TransportBinding>…</sp:TransportBinding>
      <sp:AsymmetricBinding>…</sp:AsymmetricBinding >
    </wsp:ExactlyOne>
  </wsp:Policy>

  <wsdl20:binding name="SecureBinding"
      interface="tns:RealTimeDataInterface" >
    <wsp:PolicyReference URI="#secure" />
    <wsdl20:operation name="GetRealQuote" >…</wsdl20:operation>
    …
  </wsdl20:binding>

  <wsdl20:service name="RealTimeDataService"
       interface="tns:RealTimeDataInterface" >
    <wsdl20:endpoint name="RealTimeDataPort"
         binding="tns:SecureBinding">
      <wsp:PolicyReference URI="#common" />
      …
    </wsdl20:endpoint>
  </wsdl20:service>
…
</wsdl20:description>
```

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.8] that requires the use of either transport-level or message-level security is attached to the binding description. The policy expression [p.8] 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.44]*]), a policy expression [p.8] that represents the addressing requirement and optimization capability is attached to the endpoint description. The policy expression [p.8] applies to any message exchange associated with the `RealTimeDataPort` endpoint.

In the above example, the `#secure` and `#common` policy expressions [p.8] 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.8] , that is, the effective policy [p.10] for the `RealTimeDataPort` endpoint.

Example 5-2. Effective Policy for the RealTimeDataPort endpoint

```
<wsp:Policy>
  <wsp:All>
    <wsp:Policy>
      <mtom:OptimizedMimeSerialization wsp:Optional="true"/>
      <wsam:Addressing>…</wsam:Addressing>
    </wsp:Policy>
    <wsp:Policy>
      <wsp:ExactlyOne>
        <sp:TransportBinding>...</sp:TransportBinding>
        <sp:AsymmetricBinding>...</sp:AsymmetricBinding >
      </wsp:ExactlyOne>
    </wsp:Policy>
  </wsp:All>
</wsp: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:binding/wsdl20:operation/wsdl20:output`
- `wsdl20:binding/wsdl20:operation/wsdl20:infault`
- `wsdl20:binding/wsdl20:operation/wsdl20:outfault`
- `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` and
- `wsdl20:interface/wsdl20:operation/wsdl20:outfault.`

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

Policy attachment points in a WSDL document are associated with specific policy subjects [p.8] as described in the table below. There are four policy subjects [p.8] in WSDL: the service policy subject, the endpoint policy subject, the operation policy subject and the message policy subject. When a policy expression [p.8] is attached to a policy subject [p.8] in a WSDL document, capabilities and requirements represented by the policy expression [p.8] apply to any message exchange or message associated with (or described by) the policy subject [p.8] .

Table 5-1. Association of Policy Attachment Points with Policy Subjects

Policy Attachment Point in a WSDL document	**WSDL Component**	**Policy Subject**
`wsdl20:service`	Service	Service
`wsdl20:endpoint`	Endpoint	Endpoint
`wsdl20:binding`	Binding	
`wsdl20:interface`	Interface	
`wsdl20:binding/wsdl20:operation`	Binding Oper-ation	Operation
`wsdl20:interface/wsdl20:operation`	Interface Operation	
`wsdl20:binding/wsdl20:operation/ wsdl20:input`	Binding Message Reference	Message for an input message
`wsdl20:interface/wsdl20:operation/wsdl20:input`	Interface Message Reference whose {direc-tion} property is ‘in’	
`wsdl20:binding/wsdl20:operation/ wsdl20:output`	Binding Message Reference	Message for an output message
`wsdl20:interface/wsdl20:operation/ wsdl20:output`	Interface Message Reference whose {direc-tion} property is ‘out’	

Policy Attachment Point in a WSDL document	WSDL Component	Policy Subject
`wsdl20:binding/wsdl20:fault`	Binding Fault	Message for an input fault message
`wsdl20:binding/wsdl20:operation/ wsdl20:infault`	Binding Fault Reference	
`wsdl20:interface/wsdl20:fault`	Interface Fault	
`wsdl20:interface/wsdl20:operation/wsdl20:infault`	Interface Fault Reference whose {direction} property is ‘in’	
`wsdl20:binding/wsdl20:fault`	Binding Fault	Message for an output fault message
`wsdl20:binding/wsdl20:operation/wsdl20:outfault`	Binding Fault Reference	
`wsdl20:interface/wsdl20:fault`	Interface Fault	
`wsdl20:interface/wsdl20:operation/wsdl20:outfault`	Interface Fault Reference whose {direction} property is ‘out’	

For a WSDL component, the attached policy [p.8] (extension to the WSDL component model is described in **5.3 Extension to WSDL Component Model** [p.27]) 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 a policy [p.8] apply to all the use of this WSDL Interface description. When attached to a WSDL Binding component, capabilities and requirements represented by a policy [p.8] apply to all the Endpoints that support this binding description.

A policy [p.8] 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.

Policies [p.8] 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.8] containing policy assertions [p.8] that apply to any possible binding description should be attached.

Policies [p.8] 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.8] containing policy assertions [p.8] that apply to any possible binding description should be attached.

Policies [p.8] 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.8] containing policy assertions [p.8] that apply to any possible binding should be attached.

Policies [p.8] MAY be attached at different levels of the WSDL component hierarchy. A message exchange with an endpoint MAY be described by the policies [p.8] in all four policy subjects [p.8] simultaneously.

The common mechanism of associating a policy expression [p.8] with a policy subject [p.8] is to attach a reference to the policy expression [p.8] to the policy subject [p.8] . As described in the WS-Policy specification [*Web Services Policy Framework [p.42]*], a reference to a policy expression [p.8] is represented using the `wsp:PolicyReference` element. A policy attachment to a WSDL element is represented by attaching a `wsp:PolicyReference` element as a child element of the WSDL element.

Policy expressions [p.8] can be included within a WSDL document or may reside external to a WSDL document. If including policy expressions [p.8] with a WSDL document is the chosen approach, it is RECOMMENDED that the `wsp:Policy` elements are included as children of the `wsdl20:description` element after the `wsdl20:types` element and referenced using the `wsp:PolicyReference` elements.

To mandate the processing of a policy expression [p.8] attached to a policy attachment point in a WSDL document, the expression MUST be marked as required using the `@wsdl20:required` flag.

If the `wsp: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.8] 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
- 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 [p.8] . The value of the {policy} property is a policy [p.8] as defined by Section 3 - Policy Model in the WS-Policy specification [*Web Services Policy Framework [p.42]*]. The following table describes the mapping from XML representation to the {policy} property.

Table 5-2. Mapping from XML representation to the {policy} property

Component	**Value**
Service	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:service` element.
Endpoint	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:endpoint` element.
Binding	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:binding` element.
Binding Operation	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:binding/wsdl20:operation` element.
Binding Fault	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:binding/wsdl20:fault` element.
Binding Message Reference	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:binding/wsdl20:operation/wsdl20:input` or `wsdl20:binding/wsdl20:operation/wsdl20:output` element.

Component	Value
Binding Fault Reference	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:binding/wsdl20:operation/wsdl20:infault` or `wsdl20:binding/wsdl20:operation/wsdl20:outfault` element.
Interface	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:interface` element.
Interface Operation	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:interface/wsdl20:operation` element, if any.
Interface Fault	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:interface/wsdl20:fault` element.
Interface Message Reference	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:interface/wsdl20:operation/wsdl20:input` or `wsdl20:interface/wsdl20:operation/wsdl20:output` element.
Interface Fault Reference	A policy [p.8] corresponding to the merge [p.10] of `wsp:Policy` or `wsp:PolicyReference` elements, if any, in the `[children]` of the `wsdl20:interface/wsdl20:operation/wsdl20:infault` or `wsdl20:interface/wsdl20:operation/wsdl20:outfault` element.

Two {policy} properties are equivalent when they represent policies that contain the same number of policy alternatives [p.8] , and each policy alternative [p.8] in the first policy is equivalent to some policy alternative [p.8] in the second policy, and conversely.

Two policy alternatives [p.8] are equivalent when each policy assertion [p.8] in the first policy alternative [p.8] is equivalent to some policy assertion [p.8] in the second policy alternative [p.8] , and conversely. If either policy alternative [p.8] contains multiple policy assertions [p.8] of the same type, policy alternative [p.8] equality is dependent on the semantics of that assertion type.

Two policy assertions [p.8] are equivalent if they have the same QName, if either policy assertion is an ignorable policy assertion [p.8] , both assertions must be ignorable policy assertions [p.8] and, if either policy assertion [p.8] 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 [p.8] equality may be further refined by the corresponding policy assertion [p.8] specification.

5.4 Effective Policy

The following diagram illustrates the four policy subjects [p.8] in WSDL and how the effective policy [p.10] is calculated for each of these policy subjects [p.8] .

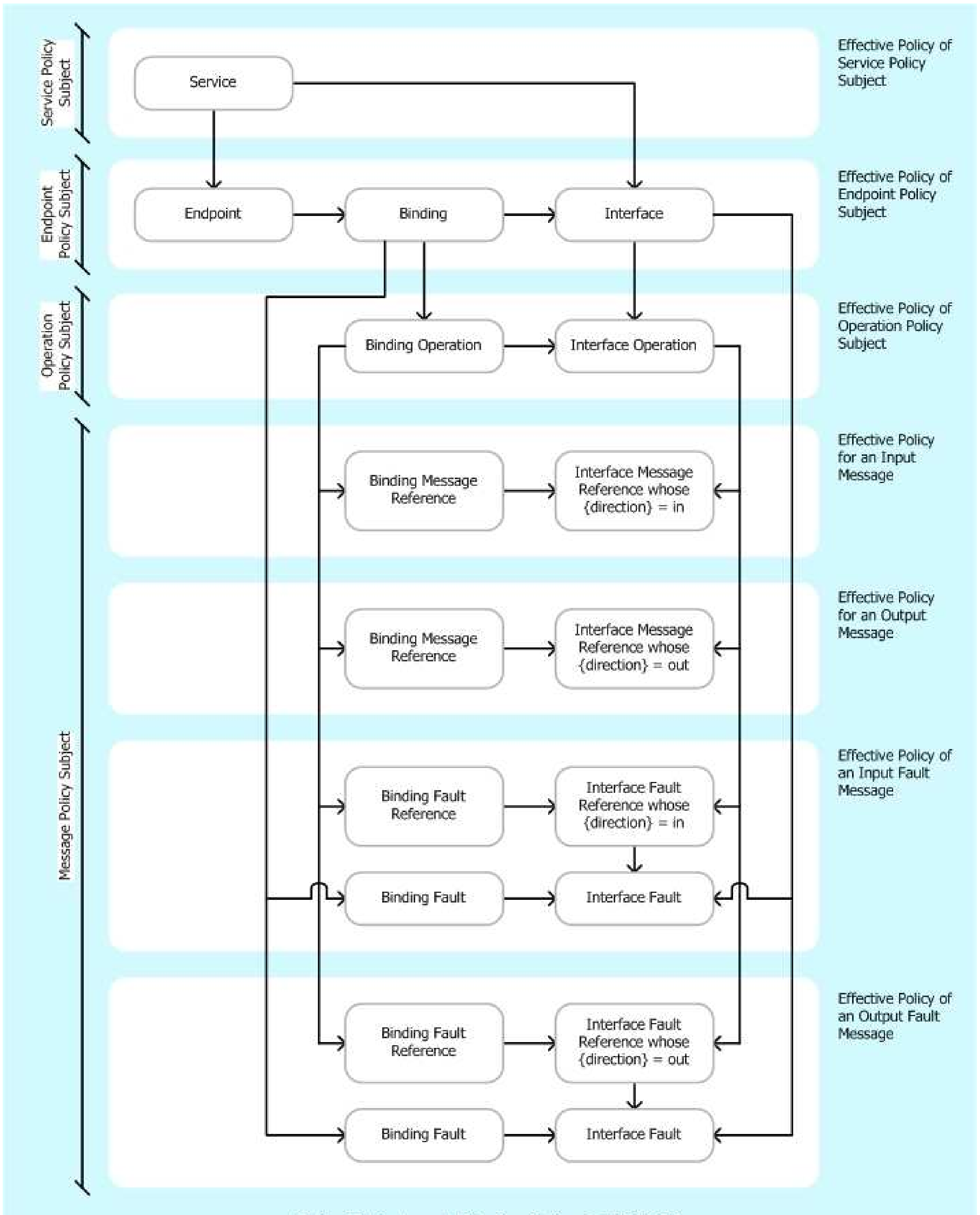
Service Policy Subject
Endpoint Policy Subject
Operation Policy Subject
Message Policy Subject
Service
Endpoint
Binding
Interface
Binding Operation
Interface Operation
Binding Message Reference
Interface Message Reference whose {direction} = in
Binding Message Reference
Interface Message Reference whose {direction} = out
Binding Fault Reference
Interface Fault Reference whose {direction} = in
Binding Fault
Interface Fault
Binding Fault Reference
Interface Fault Reference whose {direction} = out
Binding Fault
Interface Fault
Effective Policy of Service Policy Subject
Effective Policy of Endpoint Policy Subject
Effective Policy of Operation Policy Subject
Effective Policy for an Input Message
Effective Policy for an Output Message
Effective Policy of an Input Fault Message
Effective Policy of an Output Fault Message
Policy Subjects and Effective Policy in WSDL 2.0

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

If multiple policies [p.8] are attached to WSDL components that collectively represent a policy subject [p.8] then the effective policy [p.10] of these policies [p.8] applies. (For example, there is a policy [p.8] attached to an Endpoint component that describes the component and there is a policy [p.8] attached to the Binding component in the Endpoint component's {binding} property.) The effective policy [p.10] is the merge [p.10] of the policies [p.8] that are attached to the same policy subject [p.8] . The rest of this section describes how the effective policy [p.10] is calculated for each of these policy subjects [p.8] .

5.4.1 Service Policy Subject

The effective policy [p.10] of a service policy subject is the policy [p.8] in the {policy} property of a Service component that describes the service.

5.4.2 Endpoint Policy Subject

The effective policy [p.10] of an endpoint policy subject is the merge [p.10] of policies [p.8] 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 [p.10] of an operation policy subject MAY be calculated by merging [p.10] the policies [p.8] 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 [p.10] of an operation policy subject MUST be calculated in relation to a specific endpoint, and is the policy [p.8] 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 [p.10] of an input message MAY be calculated by merging [p.10] the policies [p.8] 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 [p.10] of an input message MUST be calculated in relation to a specific endpoint, and is the policy [p.8] 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 [p.10] of an output message MAY be calculated by merging [p.10] the policies [p.8] 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 [p.8] 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 [p.10] of an input fault message MAY be calculated by merging [p.10] the policies [p.8] 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 [p.10] of an input fault message MUST be calculated in relation to a specific endpoint, and is the merge [p.10] of policies [p.8] 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 [p.10] of an output fault message MAY be calculated by merging [p.10] the policies [p.8] 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 [p.10] of an output fault message MUST be calculated in relation to a specific endpoint, and is the merge [p.10] of policies [p.8] 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 policy subjects [p.8] through the use of UDDI. It defines a minimum level of support for associating policy expressions [p.8] with entities in a UDDI registry. The calculation of effective policy [p.10] for UDDI entities is described in Section **6.1 Calculating Effective Policy and Element Policy in UDDI** [p.35] . While the general concept for associating policy expressions [p.8] with UDDI entities, which is specified in Sections **6.2 Referencing Remote Policy Expressions** [p.36] and **6.3 Registering Reusable Policy Expressions** [p.38] , is based on UDDI Version 2 [*UDDI API 2.0 [p.42]* , *UDDI Data Structure 2.0 [p.42]*], the necessary changes with respect to UDDI Version 3 [*UDDI 3.0 [p.42]*] are explained in Section **6.4 Registering Policies in UDDI**

Version 3 [p.40] .

There are essentially two approaches for registering policies in UDDI. One approach is to directly reference remotely accessible policy expressions [p.8] in UDDI entities, the other is to register policy expressions [p.8] as distinct tModels and then reference these tModels in each UDDI entity that is using the policy expression [p.8] . While the former approach (see Section **6.2 Referencing Remote Policy Expressions** [p.36]) is expected to be used for policy expressions [p.8] that are mainly unique for a given Web service, the latter approach (see Section **6.3 Registering Reusable Policy Expressions** [p.38]) is expected to be used for more modular and reusable policy expressions [p.8] .

6.1 Calculating Effective Policy and Element Policy in UDDI

When attaching a policy [p.8] to a UDDI entity a policy scope [p.8] is implied for that attachment. The policy scope [p.8] only contains the policy subjects [p.8] associated with that entity, and not those associated with the children of that entity. This policy [p.8] is the entity's element policy [p.11] .

Each policy assertion [p.8] contained within a UDDI entity's element policy [p.11] 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 [p.11] is considered an intrinsic part of the tModel and applies to all uses of that tModel. In particular, it MUST be merged [p.10] into the effective policy [p.10] of every bindingTemplate that references that tModel.

Policies that apply to deployed Web services (bindingTemplates) are only considered in the effective policy [p.10] of that deployed resource itself.

Each of these entities MAY have an element policy [p.11] per Section **3. Policy Attachment** [p.10] . The remainder of this section defines how that element policy [p.11] is interpreted to calculate the effective policy [p.10] .

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 element policy [p.11] as per Section **3. Policy Attachment** [p.10] , and if present MUST be merged [p.10] into the effective policy [p.10] 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 [p.11] as per Section **3. Policy Attachment** [p.10] , and if present MUST be merged [p.10] into the effective policy [p.10] 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 [p.11] as per Section **3. Policy Attachment** [p.10] . The policy scope [p.8] 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 [p.10] for a UDDI endpoint includes the element policy [p.11] of the `uddi:bindingTemplate` element that defines the endpoint merged [p.10] with the element policy [p.11] 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 [p.8] 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** [p.45] .

UDDI registries MUST use the (UDDI V2 [*UDDI Data Structure 2.0 [p.42]*]) `tModelKey` `uuid:2ea5f9d3-9b84-39f9-b334-e9d5f535b4c1` 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.8] ; that is, when referencing this category system in a `categoryBag`, the corresponding `keyValue` of the `keyedReference` is the IRI of the policy expression [p.8] .

Using the remote policy reference category system, one can then associate a policy expression [p.8] with a `businessEntity`, a `businessService`, and a tModel using the entity's `categoryBag`. For example, associating the policy expression [p.8] that is identified by the IRI `http://www.example.com/myservice/policy` with a `businessService` is done as follows:

```
<businessService serviceKey="…" >
  <name>…</name>
  <description>…</description>
  <bindingTemplates>…</bindingTemplates>
  <categoryBag>
    <keyedReference
       keyName="Policy Expression for example's Web services"
       keyValue="http://www.example.com/myservice/policy"
       tModelKey="uuid:2ea5f9d3-9b84-39f9-b334-e9d5f535b4c1" />
  </categoryBag>
</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.8] .

A different approach has to be taken to associate a policy expression [p.8] 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:

```
<bindingTemplate bindingKey="…" >
  <accessPoint>…</accessPoint>
  <tModelInstanceDetails>
    <tModelInstanceInfo
       tModelKey="uuid:2ea5f9d3-9b84-39f9-b334-e9d5f535b4c1" >
      <instanceDetails>
        <instanceParms>
          http://www.example.com/myservice/policy
        </instanceParms>
      </instanceDetails>
    </tModelInstanceInfo>
  </tModelInstanceDetails>
</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.8] .

6.3 Registering Reusable Policy Expressions

In addition to using the approach outlined in the section above, publishers may register a specific policy expression [p.8] in a UDDI registry as a distinct tModel. To properly categorize tModels as policy expressions [p.8] , 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** [p.46] .

The following illustrates a tModel for the policy expression [p.8] identified by the IRI `http://www.example.com/myservice/policy`.

```
<tModel tModelKey="uuid:04cfa…">
  <name>…</name>
  <description xml:lang="EN">
    Policy Expression for example’s Web services
  </description>
  <overviewDoc>
    <description xml:lang="EN">Web Services Policy Expression</description>
    <overviewURL>http://www.example.com/myservice/policy</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
       keyName="Reusable policy Expression"
       keyValue="policy"
       tModelKey="uuid:90eda65e-ffd7-33df-965f-98ebb1c2a941" />
    <keyedReference
       keyName="Policy Expression for example’s Web services"
       keyValue="http://www.example.com/myservice/policy"
       tModelKey="uuid:2ea5f9d3-9b84-39f9-b334-e9d5f535b4c1" />
  </categoryBag>
</tModel>
```

The first `keyedReference` specifies that the tModel represents a policy expression [p.8] — 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 [p.8] in general. The second `keyedReference` designates the policy expression [p.8] the tModel represents by using the approach from the section above. This is necessary in order to enable UDDI inquiries for particular policy expressions [p.8] based on their IRI.

Note that the policy expression [p.8] IRI is also specified in the tModel’s overview URL to indicate that it is a resolvable URL to actually retrieve the policy expression [p.8] .

Web Services Policy 1.5 - Attachment pre-defines another tModel that is used to associate such a pre-registered, locally available policy expressions [p.8] 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** [p.47] .

UDDI registries MUST use the `tModelKey` `uuid:5da4fc61-a302-35ad-91d3-775150429035` 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.8] .

Given the local policy reference category system, one can then associate a policy expression [p.8] tModel with a `businessEntity`, a `businessService`, and a tModel using the entity's `categoryBag`. For example, associating the policy expression [p.8] tModel with the `tModelKey` `"uuid:04cfa…"` from above with a `businessService` is done as follows:

```
<businessService serviceKey="…" >
  <name>…</name>
  <description>…</description>
  <bindingTemplates>…</bindingTemplates>
  <categoryBag>
    <keyedReference
       keyName="Policy Expression for example's Web services"
       keyValue="uuid:04cfa…"
       tModelKey="uuid:5da4fc61-a302-35ad-91d3-775150429035" />
  </categoryBag>
</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.8] that is registered with the UDDI registry.

A different approach has to be taken to associate a policy expression [p.8] 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:

```
<bindingTemplate bindingKey="…" >
  <accessPoint>…</accessPoint>
  <tModelInstanceDetails>
    <tModelInstanceInfo
       tModelKey="uuid:5da4fc61-a302-35ad-91d3-775150429035" >
      <instanceDetails>
        <instanceParms>uuid:04cfa…</instanceParms>
      </instanceDetails>
    </tModelInstanceInfo>
  </tModelInstanceDetails>
</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.8] that is registered with the UDDI registry.

6.4 Registering Policies in UDDI Version 3

UDDI Version 3 [*UDDI 3.0 [p.42]*] 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 3.0 [p.42]*]. 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:2ea5f9d3-9b84-39f9-b334-e9d5f535b4c1"` to `"uddi:w3.org:ws-policy:v1.5:attachment:remotepolicyreference"`.

The `tModelKey` for the Web Services Policy Types tModel changes from `"uuid:90eda65e-ffd7-33df-965f-98ebb1c2a941"` to `"uddi:w3.org:ws-policy:v1.5:attachment:policytypes"`.

The `tModelKey` for the local policy reference tModel changes from `"uuid:5da4fc61-a302-35ad-91d3-775150429035"` to `"uddi:w3.org:ws-policy:v1.5:attachment:localpolicyreference"`.

Second, rather than putting policy expression [p.8] 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** [p.35] would be changed as follows:

```
<bindingTemplate bindingKey="…" >
  <accessPoint>…</accessPoint>
  <tModelInstanceDetails>…</tModelInstanceDetails>
  <categoryBag>
    <keyedReference
       keyName="Policy Expression for example's Web services"
       keyValue="http://www.example.com/myservice/policy"
       tModelKey="uddi:w3.org:ws-policy:v1.5:attachment:remotepolicyreference"
    />
  </categoryBag>
</bindingTemplate>
```

Third, inquiries for reusable policy expression [p.8] tModels described in Section **6.3 Registering Reusable Policy Expressions** [p.38] and UDDI tModel entities that are associated with remote policy expression [p.8] is enhanced by the wildcard mechanism for keyValues in keyedReferences. For example, searching for all policy expression [p.8] tModels whose IRI starts with `http://www.example.com/`, the following `find_tModel` API call can be used:

```
<find_tModel
      xmlns="urn:uddi-org:api_v3" >
  <categoryBag>
    <keyedReference
       keyValue="http://www.example.com/"
       tModelKey="uddi:w3.org:ws-policy:v1.5:attachment:remotepolicyreference"
    />
  </categoryBag>
  <findQualifiers>
    <findQualifier>approximateMatch</findQualifier>
  </findQualifiers>
</find_tModel>
```

Fourth, all UDDI entities may be digitally signed using XML digital signatures [*XML-Signature [p.45]*]. Publishers who want to digitally sign their policy expression [p.8] tModels or policy expression [p.8] references in UDDI MUST use the Schema-centric canonicalization algorithm [*SCC14N [p.44]*].

7. Security Considerations

It is RECOMMENDED that policy attachments [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.45]*], SSL/TLS [*IETF RFC 2246 [p.44]*], or WS-Security 2004 [*WS-Security 2004 [p.43]*]. This also provides a mechanism for authenticating policy attachments [p.8] by determining if the signer has the right to "speak for" the scope of the policy attachment [p.8] .

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 [p.8] .

A more complete discussion of security considerations can be found in the Security Considerations section of the Web Services Policy 1.5 - Framework document [*Web Services Policy Framework [p.42]*].

8. Conformance

8.1 External Policy Attachment Conformance

An element information item whose namespace name is "http://www.w3.org/ns/ws-policy" and whose local part is PolicyAttachment conforms to this specification if it is valid according to the XML Schema [*XML Schema Structures [p.43]*] 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 Section **3.4 External Policy Attachment** [p.12] 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 [*WSDL 1.1 [p.43]*] description conforms to this specification when it incorporates one or more element policies and additionally adheres to all the constraints contained in section **4. Attaching Policies Using WSDL 1.1** [p.15]

8.3 WSDL 2.0 Attachment Conformance

A WSDL 2.0 [*WSDL 2.0 Core Language [p.43]*] description conforms to this specification when it incorporates one or more element policies and additionally adheres to all the constraints contained in section **5. WS-Policy Attachment for WSDL 2.0** [p.22]

A. References

A.1 Normative References

[BP 1.1]
Basic Profile Version 1.1, K. Ballinger, et al, Editors. The Web Services-Interoperability Organization, 10 April 2006. This version of the Basic Profile Version 1.1 is http://www.ws-i.org/Profiles/BasicProfile-1.1-2006-04-10.html. The latest version of the Basic Profile Version 1.1 is available at http://www.ws-i.org/Profiles/BasicProfile-1.1.html

[IETF RFC 3023]
IETF "RFC 3023: XML Media Types", M. Murata, S. St. Laurent, D. Kohn, January 2001. (See *http://www.ietf.org/rfc/rfc3023.txt*.)

[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.

[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 3.0 specification 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://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi_v3.htm.

[Web Services Policy Framework]
Web Services Policy 1.5 - Framework, 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 - Framework specification is http://www.w3.org/TR/2007/REC-ws-policy-20070904. The latest version of Web Services Policy 1.5 - Framework is available at http://www.w3.org/TR/ws-policy.

[WS-Addressing Core]
Web Services Addressing 1.0 - Core, M. Gudgin, M. Hadley, and T. Rogers, Editors. World Wide Web Consortium, 9 May 2006. This version of the Web Services Addressing 1.0 - Core Recommendation is http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/. The latest version of Web Services Addressing 1.0 - Core is available at http://www.w3.org/TR/ws-addr-core.

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

[XML Schema 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.

[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.

A.2 Other References

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

[WSDL11 ElementIds]
WSDL 1.1 Element Identifiers D. Orchard, A. S. Vedamuthu, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, Ü. Yalçinalp, Editors. World Wide Web Consortium, 20 July 2007. This version of the WSDL 1.1 Element Identifiers Working Group Note is http://www.w3.org/TR/2007/NOTE-wsdl11elementidentifiers-20070720/. The latest version of WSDL 1.1 Element Identifiers is available at http://www.w3.org/TR/wsdl11elementidentifiers/.

[MTOM]
SOAP Message Transmission Optimization Mechanism, M. Gudgin, N. Mendelsohn, M. Nottingham and H. Ruellan, Editors. World Wide Web Consortium, 25 January 2005. This version of the SOAP Message Transmission Optimization Mechanism Recommendation is http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/. The latest version of SOAP Message Transmission Optimization Mechanism is available at http://www.w3.org/TR/soap12-mtom/.

[WS-MTOMPolicy]
MTOM Serialization Policy Assertion (WS-MTOMPolicy), C. Ferris, et al, Authors. International Business Machines Corporation and Microsoft Corporation, Inc., September 2006. Available at http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization/

[SCC14N]
Schema Centric XML Canonicalization Version 1.0, S. Aissi, A. Hately, and M. Hondo, Editors. Organization for the Advancement of Structured Information Standards, 23 May 2005. This version of the Schema Centric XML Canonicalization Version 1.0 is http://uddi.org/pubs/SchemaCentric-Canonicalization-20050523.htm. The latest version of Schema Centric XML Canonicalization Version 1.0 is available at http://uddi.org/pubs/SchemaCentricCanonicalization.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-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/

[WS-RM Policy]
Web Services Reliable Messaging Policy Assertion (WS-RM Policy), D. David, A. Kamarkar, G. Pilz, and Ü. Yalçinalp, Editors. Organization for the Advancement of Structured Information Standards, 14 June 2007. Available at http://docs.oasis-open.org/ws-rx/wsrmp/200702.

[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 Binding for SOAP 1.2]
WSDL 1.1 Binding for SOAP 1.2, D. Angelov, et al, Authors. World Wide Web Consortium, 5 April 2006. Available at http://www.w3.org/Submission/2006/SUBM-wsdl11soap12-20060405/.

[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/.

[XOP]
XML-binary Optimized Packaging, M. Gudgin, N. Mendelsohn, M. Nottingham and H. Ruellan, Editors. World Wide Web Consortium, 25 January 2005. This version of the XML-binary Optimized Packaging Recommendation is http://www.w3.org/TR/2005/REC-xop10-20050125/. The latest version of XML-binary Optimized Packaging is available at http://www.w3.org/TR/xop10/.

[XPointer Framework]
XPointer Framework, Paul Grosso, Eve Maler, Jonathan Marsh, Norman Walsh, Editors. World Wide Web Consortium, 25 March 2003. This version of the XPointer Framework Proposed Recommendation is http://www.w3.org/TR/2003/REC-xptr-framework-20030325/ The latest version of XPointer Framework is available at http://www.w3.org/TR/xptr-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.34] . The tModelKeys shown in the tModel structure sections are valid UDDI Version 3 keys. When using UDDI Version 2, 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.8] to a UDDI entity by referencing the policy's IRI.

B.1.2 tModel Definition

Name:	http://www.w3.org/ns/ws-policy/remotepolicyreference
Description:	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.
UDDI Key (V3):	`uddi:w3.org:ws-policy:v1.5:attachment:remotepolicyreference`
UDDI V1,V2 format key:	`uuid:2ea5f9d3-9b84-39f9-b334-e9d5f535b4c1`
Categorization:	categorization
Checked:	No

B.1.3 tModel Structure

```
<tModel tModelKey="uuid:2ea5f9d3-9b84-39f9-b334-e9d5f535b4c1" >
    <name>http://www.w3.org/ns/ws-policy/remotepolicyreference</name>
  <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>
  <categoryBag>
    <keyedReference
       keyName="uddi-org:types:categorization"
       keyValue="categorization"
       tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4" />
  </categoryBag>
</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 [p.8] . There is only one valid value, namely `"policy"`, that indicates this very fact. It is RECOMMENDED that tModels categorized as representing policy expressions [p.8] reference no more and no less than this very policy expression [p.8] using the remote policy reference category system.

B.2.2 tModel Definition

Name:	http://www.w3.org/ns/ws-policy/policytypes
Description:	Web services policy types category system used for UDDI tModels to characterize them as Web services policy–based policy expressions [p.8] .
UDDI Key (V3):	`uddi:w3.org:ws-policy:v1.5:attachment:policytypes`
UDDI V1,V2 format key:	`uuid:90eda65e-ffd7-33df-965f-98ebb1c2a941`
Categorization:	categorization
Checked:	No

B.2.3 tModel Structure

```
<tModel tModelKey="uuid:90eda65e-ffd7-33df-965f-98ebb1c2a941" >
  <name>http://www.w3.org/ns/ws-policy/policytypes</name>
  <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>
  <categoryBag>
    <keyedReference
       keyName="uddi-org:types:categorization"
       keyValue="categorization"
       tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4" />
  </categoryBag>
</tModel>
```

B.3 Local Policy Reference Category System

B.3.1 Design Goals

This tModel is used to attach a policy expression [p.8] to a UDDI entity by referencing the UDDI entity that represents this policy expression [p.8] . 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 [p.8] using the Web services policy types category system.

B.3.2 tModel Definition

Name:	http://www.w3.org/ns/ws-policy/localpolicyreference
Description:	Category system used for UDDI entities to point to a Web services policy policy expression [p.8] tModel that describes their characteristics. See Web Services Policy 1.5 - Attachment specification for further details.
UDDI Key (V3):	`uddi:w3.org:ws-policy:v1.5:attachment:localpolicyreference`
UDDI V1,V2 format key:	`uuid:5da4fc61-a302-35ad-91d3-775150429035`
Categorization:	categorization
Checked:	Yes

B.3.3 tModel Structure

```
<tModel tModelKey="uuid:5da4fc61-a302-35ad-91d3-775150429035" >
  <name>http://www.w3.org/ns/ws-policy/localpolicyreference</name>
  <description xml:lang="en">Category system used for UDDI entities to point to a
   Web Services Policy policy expression tModel that describes their characteristics.
   See Web Services Policy 1.5 - Attachment specification for further details.</description>
  <categoryBag>
    <keyedReference
       keyName="uddi-org:types:categorization"
       keyValue="categorization"
       tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62aB4" />
    <keyedReference
       keyName="uddi-org:types:checked"
       keyValue="checked"
       tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62aB4" />
      <keyedReference
       keyName="uddi-org:entityKeyValues"
       keyValue="tModelKey"
       tModelKey="uuid:916b87bf-0756-3919-8eae-97dfa325e5a4" />
  </categoryBag>
</tModel>
```

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.