Web Services Policy 1.5 - Attachment

W3C Working Draft 31 July 2006

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

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

Abstract

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 First Public Working Draft of the Web Services Policy 1.5 - Attachment specification. This Working Draft was produced by the members of the Web Services Policy Working Group. The Working Group expects to advance this Working Draft to Recommendation Status. It represents a transcription of the original Member Submission specification into the W3C style. Note that this Working Draft does not necessarily represent a consensus of the Working Group. Technical discussion related to the development of this specification takes place on the public-ws-policy mailing list (archive).

Table of Contents

Appendices

---

1. Introduction

The Web Services Policy 1.5 - Framework [*Web Services Policy Framework [p.25]*] specification defines an abstract model and an XML-based language for policies. This specification, Web Services Policy 1.5 - Attachment, defines two general-purpose mechanisms for associating policies with the subjects to which they apply; the policies may be defined as part of existing metadata about the subject or the policies may be defined independently and associated through an external binding to the subject.

To enable Web Services Policy to be used with existing Web service technologies, this specification describes the use of these general-purpose mechanisms with WSDL [*WSDL 1.1 [p.26]* , *WSDL 2.0 Core Language [p.26]*] and UDDI [*UDDI API 2.0 [p.25]* , *UDDI Data Structure 2.0 [p.25]* , *UDDI 3.0 [p.25]*]. Specifically, this specification defines the following:

- How to reference policies from WSDL definitions.
- How to associate policies with deployed Web service endpoints.
- How to associate policies with UDDI entities.

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.26]*]. Information items properties are indicated by the style **[infoset property]**.
- XML namespace prefixes (see Table 2-1 [p.4]) 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. Information Items MAY be added at the indicated extension points but MUST NOT contradict the semantics of the element information item indicated by the **[parent]** or **[owner]** property of the extension. If a processor does not recognize an Attribute Information Item, the processor SHOULD ignore it; if a processor does not recognize an Element Information Item, the processor SHOULD treat it as an assertion.

Normative text within this specification takes precedence over normative outlines, which in turn take precedence over the XML Schema [*XML Schema Structures [p.26]*] descriptions.

2.2 XML Namespaces

This specification uses a number of namespace prefixes throughout; they are listed in Table 2-1 [p.4] . Note that the choice of any namespace prefix is arbitrary and not semantically significant (see [*XML Namespaces [p.26]*]).

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

Prefix	XML Namespace	Specification
rmp	http://docs.oasis-open.org/ws-rx/wsrmp/200602	[*WS-RM Policy [p.27]*]
sp	http://schemas.xmlsoap.org/ws/2005/07/securitypolicy	[*WS-SecurityPolicy [p.27]*]
wsa	http://www.w3.org/2005/08/addressing	[*WS-Addressing Core [p.26]*]
wsdl11	http://schemas.xmlsoap.org/wsdl/	[*WSDL 1.1 [p.26]*]
wsdl20	http://www.w3.org/2006/01/wsdl	[*WSDL 2.0 Core Language [p.26]*]
wsoap12	http://schemas.xmlsoap.org/wsdl/soap12/	[*WSDL 1.1 Binding for SOAP 1.2 [p.27]*]
wsp	http://www.w3.org/2006/07/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.26]*]
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	[*WS-Security 2004 [p.26]*]
xs	http://www.w3.org/2001/XMLSchema	[*XML Schema Structures [p.26]*]

All information items defined by this specification are identified by the XML namespace URI [*XML Namespaces [p.26]*] http://www.w3.org/2006/07/ws-policy. A normative XML Schema [*XML Schema Structures [p.26]* , *XML Schema Datatypes [p.26]*] document can be obtained by dereferencing the XML 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 but rather change only when a subsequent revision, published as a WD, CR or PR draft results in non-backwardly compatible changes from a previously published WD, CR or PR draft of the specification.

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

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

2.3 Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [*IETF RFC 2119 [p.25]*].

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

[Definition: A **policy** is a collection of policy alternatives [p.6] .]

[Definition: A **policy alternative** is a collection of policy assertions [p.6] .]

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

[Definition: A **policy expression** is an XML Infoset representation of a policy [p.6] , either in a normal form or in an equivalent compact form.]

[Definition: A **policy subject** is an entity (e.g., an endpoint, message, resource, interaction) with which a policy [p.6] can be associated.]

[Definition: A **policy scope** is a collection of policy subjects [p.6] to which a policy may apply.]

[Definition: A **policy attachment** is a mechanism for associating policy [p.6] with one or more policy scopes [p.6] .]

[Definition: An **effective policy**, for a given policy subject [p.6] , is the resultant combination of relevant policies. The relevant policies are those attached to policy scopes [p.6] that contain the policy subject.]

[Definition: The **element policy** is the policy [p.6] attached to the policy subjects [p.6] associated with the element information item that contains it.]

2.4 Example

This specification defines several mechanisms for associating policies (Web Services Policy 1.5 - Framework, [*Web Services Policy Framework [p.25]*]) with various XML Web service entities. For brevity, we define two sample policy expressions [p.6] that the remainder of this document references.

The example in Example 2-1 [p.7] indicates a policy [p.6] for reliable messaging [*WS-RM Policy [p.27]*]. The example in Example 2-2 [p.7] is a policy for securing messages using X509 certificates [*WS-SecurityPolicy [p.27]*].

Example 2-1. Example RM Policy Expression.

```
(01) <wsp:Policy
        xmlns:rmp="http://docs.oasis-open.org/ws-rx/wsrmp/200602"
        xmlns:wsp="http://www.w3.org/2006/07/ws-policy"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
        wsu:Id="RmPolicy" >
(02)   <rmp:RMAssertion>
(03)     <rmp:InactivityTimeout Milliseconds="600000" />
(04)     <rmp:BaseRetransmissionInterval Milliseconds="3000" />
(05)     <rmp:ExponentialBackoff />
(06)     <rmp:AcknowledgementInterval Milliseconds="200" />
(07)   </rmp:RMAssertion>
(08) </wsp:Policy>
```

Example 2-2. Example X509 Security Policy Expression.

```
(01) <wsp:Policy
       xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
       xmlns:wsp="http://www.w3.org/2006/07/ws-policy"
       xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
       wsu:Id="X509EndpointPolicy" >
(02)   <sp:AsymmetricBinding>
(03)     <wsp:Policy>
(04)       <sp:RecipientToken>
(05)         <wsp:Policy>
(06)           <sp:X509Token sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/Never">
(07)             <wsp:Policy>
(08)               <sp:WssX509V3Token10 />
(09)             </wsp:Policy>
(10)           </sp:X509Token>
(11)         </wsp:Policy>
(12)       </sp:RecipientToken>
(13)       <sp:InitiatorToken>
(14)         <wsp:Policy>
(15)           <sp:X509Token sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient" >
(16)             <wsp:Policy>
(17)               <sp:WssX509V3Token10 />
(18)             </wsp:Policy>
(19)           </sp:X509Token>
(20)         </wsp:Policy>
(21)       </sp:InitiatorToken>
(22)       <sp:AlgorithmSuite>
(23)         <wsp:Policy>
(24)           <sp:Basic256Rsa15 />
(25)         </wsp:Policy>
(26)       </sp:AlgorithmSuite>
(27)       <sp:Layout>
(28)         <wsp:Policy>
(29)           <sp:Lax />
(30)         </wsp:Policy>
```

```
(31)       </sp:Layout>
(32)       <sp:IncludeTimestamp />
(33)       <sp:OnlySignEntireHeadersAndBody />
(34)     </wsp:Policy>
(35)   </sp:AsymmetricBinding>
(36) </wsp:Policy>
```

The document containing both of these policy expressions is assumed to be located at `http://www.example.com/policies`. Per Section 3.2 Policy Identification of Web Services Policy 1.5 - Framework [*Web Services Policy Framework [p.25]*], the URIs used for these policy expressions [p.6] 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.7] and Example 2-2 [p.7] , respectively.

3. Policy Attachment

This section defines two general-purpose mechanisms for associating policies [p.6] with one or more policy subjects [p.6] . 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.6] will often be associated with a particular policy subject [p.6] using multiple policy attachments [p.6] . For example, there may be attachments at different points in a WSDL description that apply to a subject, and other attachments may be made by UDDI and other mechanisms.

When multiple attachments are made, they must be combined to ascertain the effective policy [p.7] for a particular policy subject [p.6] . This is done by identifying which policy scopes [p.6] a particular subject is in and combining the individual policies associated with those scopes to form an effective policy.

This combination can be achieved by a *merge* operation. This consists of serializing each policy as a policy expression [p.6] , 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 subject.

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

3.2 Policy Attachment Mechanisms

This section defines two general-purpose mechanisms for associating policies [*Web Services Policy Framework [p.25]*] with one or more policy subjects [p.6] . The first allows XML-based descriptions of resources to associate policy [p.6] as part of their intrinsic definition. The second allows policies to be associated with arbitrary policy subjects independently from their definition.

3.3 XML Element Attachment

It is often desirable to associate policies with the XML elements describing a subject; this allows description formats such as WSDL to be easily used with the Web Services Policy Framework (see Section **4. Attaching Policies Using WSDL 1.1** [p.12] for the specific details of WSDL attachment).

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.26]*] and Policy.

This specification defines a global attribute that allows policy expressions [p.6] 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" type="wsp:tPolicyURIs" />
</xs:schema>
```

The namespace URI [*XML Namespaces [p.26]*] for this attribute is `http://www.w3.org/2006/07/ws-policy`.

The `wsp:PolicyURIs` attribute contains a white space-separated list of one or more URIs [*IETF RFC 3986 [p.25]*]. When this attribute is used, each of the values identifies a policy expression [p.6] as defined by [*Web Services Policy Framework [p.25]*]. If more than one URI is specified, the individual referenced policies [p.6] need to be *merged* together to form a single element policy expression [p.6] . The resultant policy [p.6] is then associated with the element information item's element policy [p.7] property.

Note that the policy scope [p.6] of the attachment is specific to the policy attachment [p.6] Mechanism using it; accordingly, any policy attachment [p.6] mechanism using this attribute MUST define the policy scope [p.6] of the attachment.

An example of element policy [p.7] through the use of this global attribute is given below using the sample policies stated in Section **2.4 Example** [p.7] .

If the policies [p.6] 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*, it would result in an element policy [p.7] whose XML 1.0 representation is listed in Example 3-1 [p.9] :

Example 3-1. Example Merged Policy Expression.

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

```
(05)       <rmp:ExponentialBackoff />
(06)       <rmp:AcknowledgementInterval Milliseconds="200" />
(07)     </rmp:RMAssertion>
(08)   <sp:AsymmetricBinding>
(09)     <wsp:Policy>
(10)       <!-- Details omitted for readability -->
(11)       <sp:IncludeTimestamp />
(12)       <sp:OnlySignEntireHeadersAndBody />
(13)     </wsp:Policy>
(14)   </sp:AsymmetricBinding>
(15) </wsp:Policy>
```

Note that this element policy [p.7] has no meaningful URI.

The presence of the `wsp:PolicyURIs` attribute does not prohibit implementations from using additional mechanisms for associating policy expressions [p.6] 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.7] , 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.6] to be associated with a policy subject [p.6] independent of that subject's definition and/or representation through the use of a `wsp:PolicyAttachment` gelement.

This element has three components: the policy scope [p.6] of the attachment, the policy expressions [p.6] being bound, and optional security information. The policy scope [p.6] of the attachment is defined using one or more extensible domain expressions that identify policy subjects [p.6] , typically using URIs.

Domain expressions identify the domain of the association. That is, the set of policy subjects [p.6] that will be considered for inclusion in the scope using an extensible domain expression model. Domain expressions identify policy subjects [p.6] to be included within the policy scope [p.6] . Domain expressions yield an unordered set of policy subjects [p.6] for consideration.

For the purposes of attaching policy [p.6] to a policy subject [p.6] through this mechanism, any policy expression [p.6] 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.6] within it, but this policy expressions [p.6] 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.6] .

/wsp:PolicyAttachment/wsp:AppliesTo

This required element’s children describe the policy scope [p.6] .

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

These child elements MUST specify and/or refine the domain expression(s) that define the policy scope [p.6] . 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.6] within a policy scope [p.6] . When more than one domain expression is present, the policy scope [p.6] contains the union of the policy subjects [p.6] identified by each expression.

This document defines one domain expression syntax; others may be defined in subsequent specifications.

/wsp:PolicyAttachment/wsp:Policy

This element is a policy expression [p.6] representing a policy [p.6] that is attached to the policy subjects [p.6] within the policy scope [p.6] .

/wsp:PolicyAttachment/wsp:PolicyReference

This element references a policy expression [p.6] to be attached to the policy subjects [p.6] that are in the policy scope [p.6] . Refer to Web Services Policy 1.5 - Framework [*Web Services Policy Framework [p.25]*] for additional details.

/wsp:PolicyAttachment/wsse:Security

This optional element allows security information such as signatures to be included. The syntax of this element is described in WS-Security [*WS-Security 2004 [p.26]*].

`/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.

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.26]*]:

```
<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.6] at `http://www.example.com/policies#RmPolicy` applies to all interactions with the endpoint at `http://www.example.com/acct`.

4. Attaching Policies Using WSDL 1.1

The RECOMMENDED means of associating a policy [p.6] with a policy subject [p.6] that has a WSDL 1.1 [*WSDL 1.1 [p.26]*] description is to attach a reference to the policy [p.6] within the WSDL component corresponding to the target policy subject [p.6] .

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.25]*] overrules this restriction and allows element extensibility everywhere. Therefore, the policy [p.6] 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 should be used for those restricted cases.

If it is necessary to include the actual policy expressions [p.6] 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.6] MAY be made available through some other means, such as WS-MetadataExchange [*WS-MetadataExchange [p.27]*].

To ensure that consumers of policy-annotated WSDL elements are capable of processing such policy attachments [p.6] , 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.6] when they appear within a WSDL description.

4.1 Calculating Effective Policy in WSDL 1.1

Policy attachments [p.6] in WSDL 1.1 can be used to associate policies [p.6] with four different types of policy subject [p.6] , identified as the service policy subject, the endpoint policy subject, the operation policy subject, and the message policy subject. These subjects should be considered as nested, due to the hierarchical nature of WSDL.

When attaching a policy [p.6] to a WSDL element, a policy scope [p.6] is implied for that attachment. The policy scope [p.6] only contains the policy subject [p.6] associated with that element and not those associated with the children of that element. Therefore, it is RECOMMENDED that each policy assertion [p.6] contained within a WSDL element's element policy [p.7] should have the correct semantic such that the 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.7] , with regard to WSDL, are calculated for each of these policy subjects [p.6] . In the diagram, the dashed boxes represent policy scope [p.6] s implied by WSDL elements. For a particular policy subject [p.6] , the effective policy [p.7] MUST *merge* the element policy [p.7] of each element with a policy scope [p.6] that contains the policy subject [p.6] .

For abstract WSDL definitions, the element policy [p.7] is considered an intrinsic part of the definition and applies to all uses of that definition. In particular, it MUST be *merged* into the effective policy [p.7] 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.7] 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 MAY be described by the effective policies [p.7] in all four subject types simultaneously.

For example, in Figure 4-1 [p.13] , for a particular input message to a deployed endpoint, there are four policy subjects [p.6] involved, each with their own effective policy [p.7] . There is an effective policy [p.7] for the message, as well as an effective policy [p.7] for the parent operation of that message, an effective policy [p.7] for the deployed endpoint, and the effective policy [p.7] for the service as a whole. All four effective policies [p.7] are applicable in relation to that specific input message.

It is RECOMMENDED that, where specific policy assertions [p.6] associated with one policy subject [p.6] are only compatible with specific policy assertions [p.6] on another policy subject [p.6] 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.6] for each policy subject [p.6] type SHOULD be compatible with each of the policy alternatives [p.6] at each of the policy subjects [p.6] parent and child policy subjects [p.6] , such that choices between policy alternatives [p.6] at each level are independent of each other.

The rest of this section describes these policy subject [p.6] types, and how the effective policy [p.7] for each policy subject [p.6] 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.7] as per Section **3. Policy Attachment** [p.8] , and if present MUST be merged into the effective policy [p.7] 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.7] as per Section **3. Policy Attachment** [p.8] . The policy scope [p.6] 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.7] for a WSDL endpoint policy subject includes the element policy [p.7] of the `wsdl11:port` element that defines the endpoint *merged* with the element policy [p.7] of the referenced `wsdl11:binding` element and the element policy [p.7] 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.7] as per Section **3. Policy Attachment** [p.8] .

The policy scope [p.6] 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.7] for a WSDL operation policy subject is calculated in relation to a specific port, and includes the element policy [p.7] of the `wsdl11:portType/wsdl11:operation` element that defines the operation *merged* 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.7] as per Section **3. Policy Attachment** [p.8] .

The policy scope [p.6] 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.7] 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.7] of the `wsdl11:message` element that defines the message's type *merged* with the element policy [p.7] of the `wsdl11:binding` and `wsdl11:portType` message definitions that describe that message.

For example, the effective policy [p.7] 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.6] on outbound messages SHOULD be avoided without the use of some form of mutual policy [p.6] exchange between the parties involved.

4.1.5 Example

As an example of the combination of these policy subjects [p.6] and effective policy [p.7] calculation, consider the WSDL type definition in Example 4-1 [p.16] that references policies.

Example 4-1. Example Policy Attached to WSDL.

```
(01) <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/200602"
        xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:wsoap12="http://schemas.xmlsoap.org/wsdl/soap12/"
        xmlns:wsp="http://www.w3.org/2006/07/ws-policy"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" >
(02)   <wsp:Policy wsu:Id="RmPolicy" >
(03)     <rmp:RMAssertion>
(04)       <rmp:InactivityTimeout Milliseconds="600000" />
(05)       <rmp:BaseRetransmissionInterval Milliseconds="3000" />
```

```
(06)       <rmp:ExponentialBackoff />
(07)       <rmp:AcknowledgementInterval Milliseconds="200" />
(08)     </rmp:RMAssertion>
(09)   </wsp:Policy>
(10)     <wsp:Policy wsu:Id="X509EndpointPolicy" >
(11)       <sp:AsymmetricBinding>
(12)         <wsp:Policy>
               <!-- Details omitted for readability -->
(13)           <sp:IncludeTimestamp />
(14)           <sp:OnlySignEntireHeadersAndBody />
(15)         </wsp:Policy>
(16)       </sp:AsymmetricBinding>
(17)     </wsp:Policy>
(18)     <wsp:Policy wsu:Id="SecureMessagePolicy" >
(19)       <sp:SignedParts>
(20)         <sp:Body />
(21)       </sp:SignedParts>
(22)       <sp:EncryptedParts>
(23)         <sp:Body />
(24)       </sp:EncryptedParts>
(25)     </wsp:Policy>
(26)     <wsdl11:import namespace="http://www.example.com/stock"
            location="http://www.example.com/stock/stock.wsdl" />
(27)     <wsdl11:binding name="StockQuoteSoapBinding" type="fab:Quote" >
(28)       <wsoap12:binding style="document"
(29)          transport="http://schemas.xmlsoap.org/soap/http" />
(30)       <wsp:PolicyReference URI="#RmPolicy" wsdl11:required="true" />
(31)       <wsp:PolicyReference URI="#X509EndpointPolicy" wsdl11:required="true" />
(32)       <wsdl11:operation name="GetLastTradePrice" >
(33)         <wsoap12:operation soapAction="http://www.example.com/stock/Quote/GetLastTradePriceRequest" />
(34)           <wsdl11:input>
(35)             <wsoap12:body use="literal" />
(36)             <wsp:PolicyReference URI="#SecureMessagePolicy"
                                      wsdl11:required="true" />
(37)           </wsdl11:input>
(38)           <wsdl11:output>
(39)             <wsoap12:body use="literal" />
(40)             <wsp:PolicyReference URI="#SecureMessagePolicy"
(41)                                  wsdl11:required="true" />
(42)           </wsdl11:output>
(43)       </wsdl11:operation>
(44)     </wsdl11:binding>
(45) </wsdl11:definitions>
```

For endpoints bound to `StockQuoteSoapBinding`, the effective policy [p.7] of the endpoint is listed in Example 3-1 [p.9] (above). For the `GetLastTradePrice` operation, an additional message-level effective policy [p.7] is in effect for the input message, whose XML 1.0 representation is listed in Example 4-2 [p.17] .

Example 4-2. Example Message Security Policy Expression.

```
(01) <wsp:Policy
       xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
       xmlns:wsp="http://www.w3.org/2006/07/ws-policy"
       xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
       wsu:Id="SecureMessagePolicy" >
(02)   <sp:SignedParts>
(03)     <sp:Body />
(04)   </sp:SignedParts>
(05)   <sp:EncryptedParts>
(06)     <sp:Body />
(07)   </sp:EncryptedParts>
(08) </wsp:Policy>
```

4.2 External Attachment to Deployed Endpoints

This section defines a *domain expression* based on Web Services Addressing [*WS-Addressing Core [p.26]*] to allow the use of the `wsp:PolicyAttachment` mechanism to reference a specific endpoint of a deployed Web service.

The following schema outline illustrates this extension:

```
<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsa:EndpointReference>... </wsa:EndpointReference>
  </wsp:AppliesTo>
  ( <wsp:Policy>...</wsp:Policy>
  | <wsp:PolicyReference>...</wsp:PolicyReference>
  ) +
</wsp:PolicyAttachment>
```

An example of an Endpoint Reference is given at the end of Section **3.4 External Policy Attachment** [p.10] to illustrate the extensibility of the `wsp:AppliesTo` element.

Use of this domain expression is equivalent to policy attachment [p.6] to a deployed endpoint in WSDL, using the `wsdl11:port` element, i.e., the effective policy [p.7] resulting from the combination of policies [p.6] declared should be considered a part of the endpoint policy scope.

5. Attaching Policies Using UDDI

This section defines a mechanism for associating policies with policy subjects [p.6] through the use of UDDI. It defines a minimum level of support for associating policy expressions [p.6] with entities in a UDDI registry. The calculation of effective policy [p.7] for UDDI entities is described in Section **5.1 Calculating Effective Policy and Element Policy in UDDI** [p.19] . While the general concept for associating policy expressions [p.6] with UDDI entities, which is specified in Sections **5.2 Referencing Remote Policy Expressions** [p.20] and **5.3 Registering Reusable Policy Expressions** [p.21] , is based on UDDI Version 2 [*UDDI API 2.0 [p.25]* , *UDDI Data Structure 2.0 [p.25]*], the necessary changes with respect to UDDI Version 3 [*UDDI 3.0 [p.25]*] are explained in Section **5.4 Registering Policies in UDDI Version 3** [p.23] .

There are essentially two approaches for registering policies in UDDI. One approach is to directly reference remotely accessible policy expressions [p.6] in UDDI entities, the other is to register policy expressions [p.6] as distinct tModels and then reference these tModels in each UDDI entity that is using the policy expression [p.6] . While the former approach (see Section **5.2 Referencing Remote Policy Expressions** [p.20]) is expected to be used for policy expressions [p.6] that are mainly unique for a given Web service, the latter approach (see Section **5.3 Registering Reusable Policy Expressions** [p.21]) is expected to be used for more modular and reusable policy expressions [p.6] .

5.1 Calculating Effective Policy and Element Policy in UDDI

When attaching a policy [p.6] to a UDDI entity a policy scope [p.6] is implied for that attachment. The policy scope [p.6] only contains the policy subjects [p.6] associated with that entity, and not those associated with the children of that entity. This policy [p.6] is the entity's element policy [p.7] .

Each policy assertion [p.6] contained within a UDDI entity's element policy [p.7] should have the correct semantic such that the 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.7] is considered an intrinsic part of the tModel and applies to all uses of that tModel. In particular, it MUST be *merged* into the effective policy [p.7] of every bindingTemplate that references that tModel.

Policies that apply to deployed Web services (bindingTemplates) are only considered in the effective policy [p.7] of that deployed resource itself.

Each of these entities MAY have an element policy [p.7] per Section **3. Policy Attachment** [p.8] . The remainder of this section defines how that element policy [p.7] is interpreted to calculate the effective policy [p.7] .

5.1.1 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.7] as per Section **3. Policy Attachment** [p.8] , and if present MUST be merged into the effective policy [p.7] 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.

5.1.2 Service Policy Subject

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

- `uddi:businessService`

This element MAY have element policy [p.7] as per Section **3. Policy Attachment** [p.8] , and if present MUST be merged into the effective policy [p.7] 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.

5.1.3 Endpoint Policy Subject

The following UDDI elements collectively describe an endpoint:

- `uddi:bindingTemplate`
- `uddi:tModel`

These elements MAY have element policy [p.7] as per Section **3. Policy Attachment** [p.8] . The policy scope [p.6] 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.7] for a UDDI endpoint includes the element policy [p.7] of the `uddi:bindingTemplate` element that defines the endpoint *merged* with the element policy [p.7] of those `uddi:tModel` elements that are referenced in contained `uddi:tModelInstanceInfo` elements.

5.2 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.6] 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.27] .

UDDI registries MUST use the `tModelKey` `uuid:a27078e4-fd38-320a-806f-6749e84f8005` to uniquely identify this tModel so that UDDI registry users can expect the same behavior across different UDDI registries.

The tModel's valid values are those URIs that identify external policy expressions [p.6] ; that is, when referencing this category system in a `categoryBag` , the corresponding `keyValue` of the `keyedReference` is the URI of the policy expression [p.6] .

Using the remote policy reference category system, one can then associate a policy expression [p.6] with a `businessEntity` , a `businessService` , and a tModel using the entity's `categoryBag` . For example, associating the policy expression [p.6] that is identified by the URI `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:a27078e4-fd38-320a-806f-6749e84f8005" />
  </categoryBag>
</businessService>
```

The `tModelKey` of the `keyedReference` MUST match the fixed `tModelKey` from the remote policy reference category system. The `keyValue` MUST be the URI that identifies the policy expression [p.6] .

A different approach has to be taken to associate a policy expression [p.6] 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:a27078e4-fd38-320a-806f-6749e84f8005" >
      <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 URI that identifies the policy expression [p.6] .

5.3 Registering Reusable Policy Expressions

In addition to using the approach outlined in the section above, publishers may register a specific policy expression [p.6] in a UDDI registry as a distinct tModel. To properly categorize tModels as policy expressions [p.6] , 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.28] .

The following illustrates a tModel for the policy expression [p.6] identified by the URI `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:fa1d77dc-edf0-3a84-a99a-5972e434e993" />
    <keyedReference
       keyName="Policy Expression for example’s Web services"
       keyValue="http://www.example.com/myservice/policy"
       tModelKey="uuid:a27078e4-fd38-320a-806f-6749e84f8005" />
  </categoryBag>
</tModel>
```

The first `keyedReference` specifies that the tModel represents a policy expression [p.6] — 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.6] in general. The second `keyedReference` designates the policy expression [p.6] 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.6] based on their URI.

Note that the policy expression [p.6] URI is also specified in the tModel’s overview URL to indicate that it is a resolvable URL to actually retrieve the policy expression [p.6] .

Web Services Policy 1.5 - Attachment pre-defines another tModel that is used to associate such a pre-registered, locally available policy expressions [p.6] 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.29] .

UDDI registries MUST use the `tModelKey` `uuid:a27f7d45-ec90-31f7-a655-efe91433527c` to uniquely identify this tModel so that UDDI registry users can expect the same behavior across different UDDI registries.

The local policy reference category system is based on tModelKeys. The valid values of this category system are those tModelKeys identifying tModels that

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

That is, when referencing this category system in a category bag, the corresponding `keyValue` of the `keyedReference` is the `tModelKey` of the tModel that represents the policy expression [p.6] .

Given the local policy reference category system, one can then associate a policy expression [p.6] tModel with a `businessEntity` , a `businessService` , and a tModel using the entity's `categoryBag` . For example, associating the policy expression [p.6] 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:a27f7d45-ec90-31f7-a655-efe91433527c" />
  </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.6] that is registered with the UDDI registry.

A different approach has to be taken to associate a policy expression [p.6] 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:a27f7d45-ec90-31f7-a655-efe91433527c" >
      <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.6] that is registered with the UDDI registry.

5.4 Registering Policies in UDDI Version 3

UDDI Version 3 [*UDDI 3.0 [p.25]*] 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.25]*]. The changes with respect to the previous sections are as follows.

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

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

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

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

Second, rather than putting policy expression [p.6] 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 **5.1 Calculating Effective Policy and Element Policy in UDDI** [p.19] 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:schemas.xmlsoap.org:remotepolicyreference:2003_03"
    />
  </categoryBag>
</bindingTemplate>
```

Third, inquiries for reusable policy expression [p.6] tModels and UDDI entities that are associated with remote policy expression [p.6] is enhanced by the wildcard mechanism for keyValues in keyedReferences. For example, searching for all policy expression [p.6] tModels whose URI 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:schemas.xmlsoap.org:remotepolicyreference:2003_03"
    />
  </categoryBag>
  <findQualifiers>
    <findQualifier>approximateMatch</findQualifier>
  </findQualifiers>
</find_tModel>
```

Fourth, all UDDI entities may be digitally signed using XML digital signatures [*XML-Signature [p.27]*]. Publishers who want to digitally sign their policy expression [p.6] tModels or policy expression [p.6] references in UDDI MUST use the Schema-centric canonicalization algorithm [*SCC14N [p.26]*].

6. Security Considerations

It is RECOMMENDED that policy attachments [p.6] be signed to prevent tampering. This also provides a mechanism for authenticating policy attachments [p.6] by determining if the signer has the right to "speak for" the scope of the policy attachment [p.6] .

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.6] .

A. References

A.1 Normative References

[BP 1.1]
Basic Profile Version 1.1, K. Ballinger, et al, Editors. The Web Services-Interoperability Organization, 24 August 2004. This version of the Basic Profile Version 1.1 is http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.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 2119]
Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, Author. Internet Engineering Task Force, June 1999. Available at http://www.ietf.org/rfc/rfc2119.txt.

[IETF RFC 3986]
Uniform Resource Identifier (URI): Generic Syntax, T. Berners-Lee, R. Fielding, and L. Masinter, Authors. Internet Engineering Task Force, January 2005. Available at http://www.ietf.org/rfc/rfc3986.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 3.0.1, L. Clément, et al, Editors. Organization for the Advancement of Structured Information Standards, 14 October 2003. This version of the UDDI Version 3.0 is http://uddi.org/pubs/uddi-v3.0.1-20031014.htm. The latest version of the UDDI 3.0 specification is available at http://uddi.org/pubs/uddi_v3.htm.

[Web Services Policy Framework]
Web Services Policy 1.5 - Framework, A. S. Vedamuthu, D. Orchard, M. Hondo, T. Boubez and P. Yendluri, Editors. World Wide Web Consortium, 31, July 2006. This version of the specification of the Web Services Policy 1.5 - Framework specification is http://www.w3.org/TR/2006/WD-ws-policy-20060731. 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, 27 March 2006. This version of the WSDL 2.0 specification is http://www.w3.org/TR/2006/CR-wsdl20-20060327. The latest version of WSDL 2.0 is available at http://www.w3.org/TR/wsdl20.

[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 Information Set]
XML Information Set (Second Edition), J. Cowan and R. Tobin, Editors. World Wide Web Consortium, 4 February 2004, revised 24 October 2001. 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, T. Bray, D. Hollander, and A. Layman, Editors. World Wide Web Consortium, 14 January 1999. This version of the XML Information Set Recommendation is http://www.w3.org/TR/1999/REC-xml-names-19990114. 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, 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, 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

[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-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, September 2004. 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, 24 April 2006. Available at http://www.oasis-open.org/committees/download.php/17838/wsrmp-1.1-spec-wd-08.pdf

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

[WSDL 1.1 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/.

B. UDDI tModel Definitions

This section contains the UDDI tModel definitions for the canonical tModels used in Section **5. Attaching Policies Using UDDI** [p.18] . The tModelKeys shown in the tModel structure sections are valid UDDI Version 2 keys. When using UDDI Version 3, the corresponding derived UDDI Version 2 keys must be used.

B.1 Remote Policy Reference Category System

B.1.1 Design Goals

This tModel is used to attach a policy [p.6] to a UDDI entity by referencing the policy's URI.

B.1.2 tModel Definition

Name:	http://schemas.xmlsoap.org/ws/2003/03/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:schemas.xmlsoap.org:remotepolicyreference:2003_03`
UDDI V1,V2 format key:	`uuid:a27078e4-fd38-320a-806f-6749e84f8005`
Categorization:	categorization
Checked:	No

B.1.3 tModel Structure

```
<tModel tModelKey="uuid:a27078e4-fd38-320a-806f-6749e84f8005" >
  <name>http://schemas.xmlsoap.org/ws/2003/03/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 Poli
  <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.6] . There is only one valid value, namely `"policy"`, that indicates this very fact. It is RECOMMENDED that tModels categorized as representing policy expressions [p.6] reference no more and no less than this very policy expression [p.6] using the remote policy reference category system.

B.2.2 tModel Definition

Name:	http://schemas.xmlsoap.org/ws/2003/03/policytypes
Description:	Web services policy types category system used for UDDI tModels to characterize them as Web services policy–based policy expressions [p.6] .
UDDI Key (V3):	`uddi:schemas.xmlsoap.org:policytypes:2003_03`
UDDI V1,V2 format key:	`uuid:fa1d77dc-edf0-3a84-a99a-5972e434e993`
Categorization:	categorization
Checked:	No

B.2.3 tModel Structure

```
<tModel tModelKey="uuid:fa1d77dc-edf0-3a84-a99a-5972e434e993" >
  <name>http://schemas.xmlsoap.org/ws/2003/03/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.6] to a UDDI entity by referencing the UDDI entity that represents this policy expression [p.6] . 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.6] using the Web services policy types category system.

B.3.2 tModel Definition

Name:	http://schemas.xmlsoap.org/ws/2003/03/localpolicyreference
Description:	Category system used for UDDI entities to point to a Web services policy policy expression [p.6] tModel that describes their characteristics. See Web Services Policy 1.5 - Attachment specification for further details.
UDDI Key (V3):	`uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03`
UDDI V1,V2 format key:	`uuid:a27f7d45-ec90-31f7-a655-efe91433527c`
Categorization:	categorization
Checked:	Yes

B.3.3 tModel Structure

```
<tModel tModelKey="uuid:a27f7d45-ec90-31f7-a655-efe91433527c" >
  <name>http://schemas.xmlsoap.org/ws/2003/03/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 - Attac
  <categoryBag>
    <keyedReference
       keyName="uddi-org:types:categorization"
       keyValue="categorization"
       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, Abbie Barbir, Charlton Barreto, Toufic Boubez (Layer 7 Technologies), Paul Cotton (Microsoft Corporation), Jeffrey Crump, Glen Daniels, Ruchith Fernando (WSO2), Christopher Ferris, William Henry, Frederick Hirsch, Maryann Hondo, Tom Jordahl, Philippe Le Hégaret (W3C/MIT), Jong Lee (BEA Systems, Inc.), Mark Little (JBoss Inc.), Ashok Malhotra, Monica Martin, Jeff Mischkinsky, Dale Moberg, Anthony Nadalin, David Orchard (BEA Systems, Inc.), Bijan Parsia (University of Manchester), Fabian Ritzmann, Daniel Roth (Microsoft Corporation), Sanka Samaranayake (WSO2), Felix Sasaki (W3C/Keio), Seumas Soltysik, Yakov Sverdlov (Computer Associates), Asir Vedamuthu (Microsoft Corporation), Sanjiva Weerawarana (WSO2), Ümit Yalçinalp, Prasad Yendluri.

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

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

Date	Author	Description
20060712	ASV	Updated the list of editors. Completed action items 20 from the Austin F2F.
20060712	DBO	Completed action item 12
20060718	DBO	Completed action items Editors to remove extraneous namespace decl in the example at the end of section 3.4 18, RFC2606 for domain names 09 (note: PLH had already done but it didn’t show up in the change log) editors to straighten up Note after example 3-1 11
20060719	TIB	Completed action item 22: Linked SVG graphic
20060721	ASV	Completed action items 23, 25 and 26 from the Austin F2F.
20060721	ASV	Completed action item 29 from the Austin F2F.
20060726	ASV	Incorporated the XML namespace URI versioning policy adopted by the WG.