

Web Services Policy Attachment (WS-PolicyAttachment)

September 2004

Authors

Siddharth Bajaj, VeriSign
Don Box, Microsoft
Dave Chappell, Sonic Software
Francisco Curbera, IBM
Glen Daniels, Sonic Software
Phillip Hallam-Baker, VeriSign
Maryann Hondo, IBM
Chris Kaler, Microsoft
Ashok Malhotra, Microsoft
Hiroshi Maruyama, IBM
Anthony Nadalin, IBM
Mark Nottingham, BEA Systems
David Orchard, BEA Systems
Hemma Prafullchandra, VeriSign
Claus von Riegen, SAP
Jeffrey Schlimmer, Microsoft
Chris Sharp (editor), IBM
John Shewchuk, Microsoft

Copyright Notice

(c) 2001-2004 [BEA Systems Inc.](#), [International Business Machines Corporation](#), [Microsoft Corporation, Inc.](#), [SAP AG](#), [Sonic Software](#), and [VeriSign Inc.](#) All rights reserved.

Permission to copy and display the WS-PolicyAttachment Specification (the "Specification", which includes WSDL and schema documents), in any medium without fee or royalty is hereby granted, provided that you include the following on ALL copies of the WS-PolicyAttachment Specification, that you make:

1. A link or URL to the WS-PolicyAttachment Specification at one of the Authors' websites
2. The copyright notice as shown in the WS-PolicyAttachment Specification.

BEA Systems, IBM, Microsoft, SAP, Sonic Software, and VeriSign (collectively, the "Authors") each agree to grant you a license, under royalty-free and otherwise reasonable, non-discriminatory terms and conditions, to their respective essential patent claims that they deem necessary to implement the WS-PolicyAttachment Specification.

THE WS-POLICYATTACHMENT SPECIFICATION IS PROVIDED "AS IS," AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE WS-POLICYATTACHMENT SPECIFICATION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE WS-POLICYATTACHMENT SPECIFICATION.

The name and trademarks of the Authors may NOT be used in any manner, including advertising or publicity pertaining to the WS-PolicyAttachment Specification or its contents without specific, written prior permission. Title to copyright in the WS-PolicyAttachment Specification will at all times remain with the Authors.

No other rights are granted by implication, estoppel or otherwise.

Abstract

The Web services Policy Framework (WS-Policy) specification defines an abstract model and an XML-based expression grammar for policies. This specification, Web Services Policy Attachment (WS-PolicyAttachment) defines two general-purpose mechanisms for associating such policies with the subjects to which they apply. This specification also defines how these general-purpose mechanisms may be used to associate WS-Policy with WSDL and UDDI descriptions.

Composable Architecture

By using the XML, WSDL, and SOAP extensibility models, the WS* specifications are designed to be composed with each other to provide a rich Web services environment. WS-PolicyAttachment by itself does not provide a negotiation solution for Web services. WS-PolicyAttachment is a building block that is used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of Policy exchange models.

Status

This WS-PolicyAttachment specification is a publicly released draft and is provided for review and evaluation only. BEA Systems, IBM, Microsoft, SAP, Sonic Software, and VeriSign hope to solicit your contributions and suggestions in the near future. BEA Systems, IBM, Microsoft, SAP, Sonic Software, and VeriSign make no warranties or representations regarding the specifications in any manner whatsoever.

Table of Contents

1. Introduction

2. Notations and Terminology

2.1 Notational Conventions

2.2 Namespaces

2.3 Terminology

2.4 Example WS-Policy Expressions

3. Policy Attachment

3.1 Effective Policy

3.2 Policy Attachment Mechanisms

3.3 XML Element Attachment

3.4 External Policy Attachment

4. Attaching Policies Using WSDL 1.1

4.1 Calculating Effective Policy in WSDL 1.1

- 4.1.1 Service Policy Subject
- 4.1.2 Endpoint Policy Subject
- 4.1.3 Operation Policy Subject
- 4.1.4 Message Policy Subject
- 4.1.5 Example
- 4.2 External Attachment to Deployed Endpoints

5. Attaching Policies Using UDDI

- 5.1 Calculating Effective Policy and Element Policy in UDDI
 - 5.1.1 Service Provider Policy Subject
 - 5.1.2 Service Policy Subject
 - 5.1.3 Endpoint Policy Subject
- 5.2 Referencing Remote Policy Expressions
- 5.3 Registering Reusable Policy Expressions
- 5.4 Registering Policies in UDDI Version 3

6. Security Considerations

7. Acknowledgements

8. References

Appendix A: UDDI tModel Definitions

- A.1 Remote Policy Reference Category System
- A.2 WS-Policy Types Category System
- A.3 Local Policy Reference Category System

1. Introduction

The WS-Policy specification defines an abstract model and an XML-based grammar for policies. This specification, Web Services Policy Attachment (WS-PolicyAttachment), 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 WS-Policy to be used with existing Web service technologies, this specification describes the use of these general-purpose mechanisms with WSDL 1.1 [[WSDL11](#)] and UDDI [[UDDI API 20](#), [UDDI Data Structure 20](#), [UDDI 30](#)]. 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

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 [RFC 2119].

Namespace [XML-NS] URIs (of the general form "some-URI") represents some application-dependent or context-dependent URI as defined in RFC 2396 [RFC 2396].

Qualified names (QNames) were introduced by XML Namespaces [XML-NS]. They were defined for element and attribute *names* (only) and provide a mechanism for concisely identifying a URI/local-name pair. The W3C has documented its views on their usage in <http://www.w3.org/2001/tag/doc/qnameids>.

WS-PolicyAttachment is designed to work with the general Web services framework including WSDL service descriptions [WSDL11], UDDI service registrations [UDDI API20, UDDI DataStructure20, UDDI30] and SOAP message structure and message processing model [SOAP11, SOAP12]. The general approaches described in this specification should be applicable to any version of SOAP, WSDL, or UDDI.

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 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.

2.2 Namespaces

The following namespaces are used in this document:

Prefix	Namespace
xs	http://www.w3.org/2001/XMLSchema
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing
wSDL	http://schemas.xmlsoap.org/wSDL/
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd

A normative copy of the XML Schema [[XMLSchema](#)] for WS-PolicyAttachment constructs may be retrieved by resolving the URI "<http://schemas.xmlsoap.org/ws/2004/09/policy>".

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).

2.3 Terminology

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

Policy – A *Policy* is a collection of Policy Alternatives.

Policy Alternative – A *Policy Alternative* is a collection of Policy Assertions.

Policy Assertion – A *Policy Assertion* represents an individual requirement, capability, or other property of a behaviour.

Policy Expression – A *Policy Expression* is an XML Infoset [[Infoset](#)] representation of a Policy.

Policy Subject – A *Policy Subject* is an entity (e.g., an endpoint, message, resource, interaction) with which a Policy can be associated.

Policy Scope – A *Policy Scope* is the collection of *Policy Subjects* to which a Policy may apply.

Policy Attachment – A *Policy Attachment* is a mechanism for associating Policy with one or more Policy Scopes.

Effective Policy – An *Effective Policy*, for a given *Policy Subject*, is the resultant combination of relevant policies. The relevant policies are those attached to Policy Scopes that contain the Policy Subject.

2.4 Example WS-Policy Expressions

This specification defines several mechanisms for associating policies [[WS-Policy](#)] with various XML Web service entities. For brevity, we define three sample Policy Expressions that the remainder of this document references. Note that these Policy Expressions include fictitious assertions using XML namespace extensibility to express domain-specific behavioral semantics.

The first example indicates a Policy for digitally signing an element. The second is a Policy for use of a security token. The third is a Policy for reliable messaging sequence creation. The fourth is a Policy for auditing.

```
<wsp:Policy xml:base="http://www.fabrikam123.com/policies" wsu:Id="DSIG" >
  <wsse:Integrity>
    <wsse:Algorithm Type="wsse:AlgSignature"
      URI="http://www.w3.org/2000/09/xmlenc#aes" />
  </wsse:Integrity>
</wsp:Policy>
```

```
<wsp:Policy xml:base="http://www.fabrikam123.com/policies" wsu:Id="TOK" >
  <wsse:SecurityToken>
    <wsse:TokenType>wsse:X509v3</wsse:TokenType>
  </wsse:SecurityToken>
</wsp:Policy>
```

```
<wsp:Policy xmlns:wsm="http://schemas.xmlsoap.org/ws/2004/03/rm"
  xml:base="http://www.fabrikam123.com/policies"
  wsu:Id="RM" >
  <wsm:SequenceCreation />
</wsp:Policy>
```

```
<wsp:Policy xmlns:wsxx="..."
  xml:base="http://www.fabrikam123.com/policies"
  wsu:Id="AUDIT" >
  <wsxx:Audit />
</wsp:Policy>
```

The URIs used for these Policy Expressions in the remainder of this document are <http://www.fabrikam123.com/policies#DSIG>, <http://www.fabrikam123.com/policies#TOK>, <http://www.fabrikam123.com/policies#RM>, and <http://www.fabrikam123.com/policies#AUDIT> respectively.

3. Policy Attachment

This section defines two general-purpose mechanisms for associating Policies [[WS-Policy](#)] with one or more Policy Subjects. 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 will often be associated with a particular Policy Subject using multiple Policy Attachments. 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 for a particular Policy Subject. This is done by identifying which Policy Scopes 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, 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 [[WS-Policy](#)] with one or more Policy Subjects. The first allows XML-based descriptions of resources to associate Policy as part of their intrinsic definition. The second allows Policies to be associated with arbitrary Policy Subjects independently from their definition.

3.3 XML Element Attachment

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 Policy Framework (see section 4 for the specific details of WSDL attachment).

The *Element Policy* is that Policy attached to the Policy Subjects associated with the element information item that contains it. 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 [[WSDL11](#)] and Policy.

This specification defines a global attribute that allows Policy Expressions 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 [[XML-NS](#)] URI for this attribute is `http://schemas.xmlsoap.org/ws/2004/09/policy`.

The `wsp:PolicyURIs` attribute contains a white space-separated list of one or more URIs. When this attribute is used, each of the values identifies a Policy Expression as defined by [[WS-Policy](#)]. If more than one URI is specified, the individual referenced Policies need to be *merged* together to form a single Element Policy Expression. The resultant Policy is then associated with the element information item's Element Policy property.

Note that the Policy Scope of the attachment is specific to the Policy Attachment Mechanism using it; accordingly, any Policy Attachment mechanism using this attribute **MUST** define the Policy Scope of the attachment.

An example of Element Policy through the use of this global attribute is given below using the sample policies stated in [Section 2.4](#).

If the Policies referenced by the following XML element

```
<MyElement wsp:PolicyURIs="http://www.fabrikaml23.com/policies#DSIG
                           http://www.fabrikaml23.com/policies#TOK" />
```

have been processed and *merged*, it would result in an Element Policy whose XML 1.0 representation is:

```

<wsp:Policy>
  <wsse:Integrity>
    <wsse:Algorithm Type="wsse:AlgSignature"
      URI="http://www.w3.org/2000/09/xmlenc#aes" />
  </wsse:Integrity>
  <wsse:SecurityToken>
    <wsse:TokenType>wsse:X509v3</wsse:TokenType>
  </wsse:SecurityToken>
</wsp:Policy>

```

Note that the Element Policy has no meaningful URI.

The presence of the *wsp:PolicyURI*s attribute does not prohibit implementations from using additional mechanisms for associating Policy Expressions 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, 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.fabrikam123.com/policies#DSIG" />
  <wsp:PolicyReference URI="http://www.fabrikam123.com/policies#TOK" />
</MyElement/>

```

3.4 External Policy Attachment

This mechanism allows Policies to be associated with a Policy Subject independent of that subject's definition and/or representation through the use of a *<wsp:PolicyAttachment>* element.

This element has three components: the Policy Scope of the attachment, the Policy Expressions being bound, and optional security information. The Policy Scope of the attachment is defined using one or more extensible domain expressions that identify Policy Subjects, typically using URIs.

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

For the purposes of attaching Policy to a Policy Subject through this mechanism, any Policy Expression contained inside of the *<wsp:AppliesTo>* element MUST NOT be considered in scope. For example, an Endpoint Reference may be used as a domain expression, and it may contain Policy Expressions within it, but this Policy Expressions are not considered in scope with respect to the *<wsp:PolicyAttachment>* element using it.

The following is the pseudo-schema for the *<wsp:PolicyAttachment>* element:

```

<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.

/wsp:PolicyAttachment/wsp:AppliesTo

This required element's children describe the Policy Scope.

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

These child elements MUST specify and/or refine the domain expression(s) that define the Policy Scope. 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 within a Policy Scope. When more than one domain expression is present, the Policy Scope contains the union of the Policy Subjects 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 representing a Policy that is attached to the Policy Subjects within the Policy Scope.

/wsp:PolicyAttachment/wsp:PolicyReference

This element references a Policy Expression to be attached to the Policy Subjects that are in the Policy Scope. Refer to WS-Policy 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 [\[WS-Security\]](#).

/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 WS-Addressing [[WS-Addressing](#)]:

```
<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsa:EndpointReference xmlns:fabrikam="..." >
      <wsa:Address>http://www.fabrikam123.com/acct</wsa:Address>
      <wsa:PortType>fabrikam:InventoryPortType</wsa:PortType>
      <wsa:ServiceName>fabrikam:InventoryService</wsa:ServiceName>
    </wsa:EndpointReference>
  </wsp:AppliesTo>
  <wsp:PolicyReference URI="http://www.fabrikam123.com/policies#DSIG" />
</wsp:PolicyAttachment>
```

In this example, the Policy Expression at <http://www.fabrikam123.com/policies#DSIG> applies to all interactions with the `fabrikam:InventoryService` at the endpoint <http://www.fabrikam123.com/acct>.

4. Attaching Policies Using WSDL 1.1

The RECOMMENDED means of associating a Policy with a Policy Subject that has a WSDL 1.1 [[WSDL11](#)] description is to attach a reference to the Policy within the WSDL component corresponding to the target Policy Subject.

WSDL/1.1 disallows the use of extensibility elements on certain elements and the use of extensibility attributes on others. Therefore, the Policy reference SHOULD be attached using the `@wsp:policyURIs` attribute or `<wsp:PolicyReference>` as child element accordingly.

If it is necessary to include the actual Policy Expressions within the WSDL description itself, it is RECOMMENDED that their `<wsp:Policy>` elements be included as children of the `<wsdl:definition>` element, and referenced using the mechanisms just described. Alternatively, the Policy Expressions MAY be made available through some other means, such as WS-MetadataExchange [[WS-MetadataExchange](#)].

To ensure that consumers of a Policy-annotated WSDL document are capable of processing such Policy Attachments, this specification defines a single extensibility element, `<wsp:UsingPolicy/>`, that MUST appear as an extensibility element in the containing `<wsdl:definitions/>` element of any WSDL that uses the Policy Attachment mechanism defined in this section and section 3.3. Moreover, if the WSDL should not be processed without the referenced policies being processed, the `<wsp:UsingPolicy>` element SHOULD be marked as a mandatory extension (e.g., with a `wsdl:required="true"` attribute).

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

4.1 Calculating Effective Policy in WSDL 1.1

Policy Attachments in WSDL/1.1 can be used to associate Policies with four different types of Policy Subject, identified as the Service Policy Subject, the Endpoint Policy Subject, the Operation Policy Subject, and the Message Policy Subject. These subjects should be considered as nested, due to the hierarchical nature of WSDL.

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

Figure 1 represents how the Effective Policies, with regard to WSDL, are calculated for each of these Policy Subjects. In the diagram, the dashed boxes represent Policy Scopes implied by WSDL elements. For a particular Policy Subject, the Effective Policy MUST *merge* the Element Policy of each element with a Policy Scope that contains the Policy Subject.

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

Policies that are attached to a deployed resource (e.g., services or ports) are only considered in the Effective Policy of that deployed resource itself.

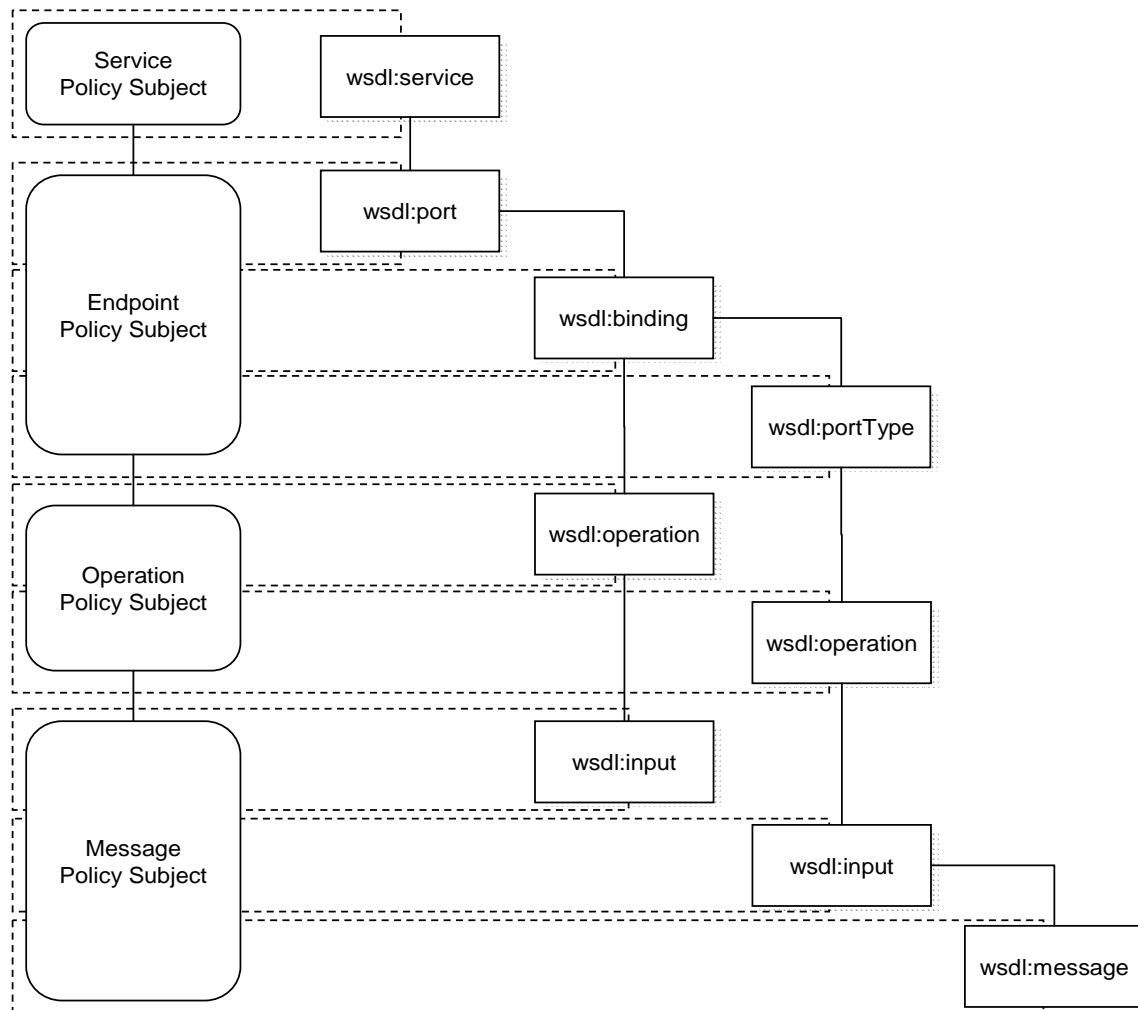


Figure 1. Effective Policy and Policy Scopes in WSDL

When attaching policies at different levels of the WSDL hierarchy, care must be taken. A message exchange with a deployed endpoint MAY contain Effective Policies in all four subject types simultaneously.

For example, in Fig.1, for a particular input message to a deployed endpoint, there are four Policy Subjects involved, each with their own Effective Policy. There is an Effective Policy for the message, as well as an Effective Policy for the parent operation of that message, an Effective Policy for the deployed endpoint, and the Effective Policy for the service as a whole. All four Effective Policies are applicable in relation to that specific input message.

It is RECOMMENDED that, where specific Policy Assertions associated with one Policy Subject are only compatible with specific Policy Assertions on another Policy Subject in the same hierarchical chain, the policies containing these assertions should be attached within a single WSDL binding hierarchy.

For any given port, the Policy Alternatives for each Policy Subject type SHOULD be compatible with each of the Policy Alternatives at each of the Policy Subjects parent and

child Policy Subjects, such that choices between Policy Alternatives at each level are independent of each other.

The rest of this section describes these Policy Subject types, and how the Effective Policy for each Policy Subject is calculated.

4.1.1 Service Policy Subject

The following WSDL/1.1 element is considered as the Service Policy Subject:

- `wsdl:service`

This element MAY have Element Policy as per [Section 3](#) of this specification, and if present MUST be merged into the Effective Policy of the WSDL Service Policy 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 port. This includes – but is not limited to – acting as a consumer or a provider of the service.

4.1.2 Endpoint Policy Subject

The following WSDL/1.1 elements collectively describe an endpoint:

- `wsdl:port`
- `wsdl:portType`
- `wsdl:binding`

These elements MAY have Element Policy as per [Section 3](#) of this specification. The Policy Scope implied by each of these elements contains the Endpoint Policy Subject representing the deployed endpoint.

Since the `wsdl: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.

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

The Effective Policy for a WSDL Endpoint Policy Subject includes the Element Policy of the `wsdl:port` element that defines the endpoint *merged* with the Element Policy of the referenced `wsdl:binding` element and the Element Policy of the referenced `wsdl: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:

- `wsdl:portType/wsdl:operation`
- `wsdl:binding/wsdl:operation`

These elements MAY have Element Policy as per [Section 3](#) of this specification.

The Policy Scope implied by each of these elements contains the Operation Policy Subject representing the specific operation of the Endpoint Policy Subject.

Since the `wsdl:portType/wsdl: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 attached to an Operation Policy Subject affect behaviours associated with a sequence of message exchanges, as defined by a WSDL operation. This includes – but is not limited to – initiating the sequence and ending the sequence.

The Effective Policy for a WSDL Operation Policy Subject is calculated in relation to a specific port, and includes the Element Policy of the `wsdl:portType/wsdl:operation` element that defines the operation *merged* with that of the corresponding `wsdl:binding/wsdl:operation` element.

4.1.4 Message Policy Subject

The following WSDL/1.1 elements are used to describe messages:

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

These elements MAY have Element Policy as per [Section 3](#) of this specification.

The Policy Scope implied by these elements contains the Message Policy Subject representing the specific input, output, or fault message in relation to the Operation Policy Subject.

Policy attached to a Message Policy Subject pertains to behaviours associated with a particular message. This includes – but is not limited to – sending and receiving the message.

The Effective Policy for a specific WSDL message (i.e., input, output, or fault message) is calculated in relation to a specific port, and includes the Element Policy of the `wsdl:message` element that defines the message's type *merged* with the Element Policy of the `wsdl:binding` and `wsdl:portType` message definitions that describe that message.

For example, the Effective Policy of a specific input message for a specific port would be the *merge* of the `wsdl:message` element defining the message type, the `wsdl:portType/wsdl:operation/wsdl:input` element, and the corresponding `wsdl:binding/wsdl:operation/wsdl:input` element for that message.

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

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

Care should be taken when attaching policies to outbound messages as the result may not be what is expected. For example, expressing a choice on a service's outbound message without a mechanism for a requester of that service to communicate its choice to the service before the outbound message is sent may not result in the desired behaviours. It is therefore RECOMMENDED that Policy Alternatives on outbound

messages SHOULD be avoided without the use of some form of mutual Policy exchange between the parties involved.

4.1.5 Example

As an example of the combination of these Policy Scopes and Effective Policy calculation, consider the following WSDL type definition that references policies:

```
<?xml version="1.0"?>
<definitions
  name="StockQuote"
  targetNamespace="http://example.com/stockquote.wsdl"
  xmlns:tns="http://example.com/stockquote.wsdl"
  xmlns:xsd="http://example.com/stockquote.xsd"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:ps="http://fabrikaml23.com/policies" >

  <wsp:UsingPolicy wsdl:Required="true" />
  ...
  <message name="GetLastTradePriceRequest" >
    <part name="body" element="xsd:TradePriceRequest" />
  </message>

  <message name="GetLastTradePriceResponse" >
    <part name="body" element="xsd:TradePrice" />
  </message>

  <portType name="StockQuotePortType"
    wsp:PolicyURIs="http://www.fabrikaml23.com/policies#RM" >
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceRequest"
        wsp:PolicyURIs="http://www.fabrikaml23.com/policies#DSIG" />
      <output message="tns:GetLastTradePriceResponse" />
    </operation>
  </portType>

  <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType" >
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http" />
```

```

<operation name="GetLastTradePrice" >
  <soap:operation soapAction="http://example.com/GetLastTradePrice" />
  <input>
    <wsp:PolicyReference
      URI="http://www.fabrikam123.com/policies#TOK" />
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
</operation>
</binding>

<service name="StockQuoteService" >
  <port name="StockQuotePort" binding="tns:StockQuoteBinding" />
  <wsp:PolicyReference
    URI="http://www.fabrikam123.com/policies#AUDIT" />
  <soap:address location="http://example.com/stockquote" />
</port>
</service>

</definitions>

```

For the deployed endpoint, StockQuotePort, the Effective Policy of the endpoint can be expressed in XML 1.0 representation as:

```

<wsp:Policy xmlns:wsm="..."
  xmlns:x="..."
  xml:base="http://www.fabrikam123.com/policies" >
  <wsm:SequenceCreation />
  <wsxx:Audit />
</wsp:Policy>

```

and, for the *GetLastTradePrice* operation, an additional message-level Effective Policy is in effect for the input message, whose XML 1.0 representation is:

```

<wsp:Policy xmlns:x="..."
  xml:base="http://www.fabrikam123.com/policies" >
  <wsse:Integrity>
    <wsse:Algorithm Type="wsse:AlgSignature"

```

```

        URI="http://www.w3.org/2000/09/xmlenc#aes" />
    </wsse:Integrity>
    <wsse:SecurityToken>
        <wsse:TokenType>wsse:X509v3</wsse:TokenType>
    </wsse:SecurityToken>
</wsp:Policy>

```

4.2 External Attachment to Deployed Endpoints

This section defines a *domain expression* based on WS-Addressing [[WS-Addressing](#)] 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 3.2 to illustrate the extensibility of the <wsp:AppliesTo> element.

Use of this domain expression is equivalent to Policy Attachment to a deployed endpoint in WSDL, using the wsdl:port element, i.e., the Effective Policy resulting from the combination of Policies 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 through the use of UDDI. It defines a minimum level of support for associating Policy Expressions with entities in a UDDI registry. The calculation of Effective Policy for UDDI entities is described in section 5.1. While the general concept for associating Policy Expressions with UDDI entities, which is specified in sections 5.2 and 5.3, is based on UDDI Version 2 [[UDDI API 20](#), [UDDI Data Structure 20](#)], the necessary changes with respect to UDDI Version 3 [[UDDI 30](#)] are explained in section 5.4.

There are essentially two approaches for registering policies in UDDI. One approach is to directly reference remotely accessible Policy Expressions in UDDI entities, the other is to register Policy Expressions as distinct tModels and then reference these tModels in each UDDI entity that is using the Policy Expression. While the former approach (see section 5.2) is expected to be used for Policy Expressions that are mainly unique for a given Web service, the latter approach (see section 5.3) is expected to be used for more modular and reusable Policy Expressions.

5.1 Calculating Effective Policy and Element Policy in UDDI

When attaching a Policy to a UDDI entity a Policy Scope is implied for that attachment. The Policy Scope only contains the Policy Subjects associated with that entity, and not those associated with the children of that entity. This Policy is the entity's Element Policy.

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

For UDDI tModels that represent Web service types, the [Element Policy] is considered an intrinsic part of the tModel and applies to all uses of that tModel. In particular, it *MUST* be *merged* into the Effective Policy of every bindingTemplate that references that tModel.

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

Each of these entities *MAY* have an Element Policy per [Section 3](#) of this specification. The remainder of this section defines how that Element Policy is interpreted to calculate the Effective Policy.

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 as per [Section 3](#) of this specification, and if present *MUST* be merged into the Effective Policy of the UDDI businessEntity Subject.

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

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 as per [Section 3](#) of this specification, and if present *MUST* be merged into the Effective Policy of the UDDI businessService Subject.

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

5.1.3 Endpoint Policy Subject

The following UDDI elements collectively describe an endpoint:

- uddi:bindingTemplate
- uddi:tModel

These elements *MAY* have Element Policy as per [Section 3](#) of this specification. The Policy Scope implied by each of these elements contains the Endpoint Policy Subject representing the deployed endpoint.

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

The Effective Policy for a UDDI endpoint includes the Element Policy of the `uddi:bindingTemplate` element that defines the endpoint *merged* with the Element Policy of those `uddi:tModel` elements that are referenced in contained `uddi:tModelInstanceInfo` elements.

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 WS-Policy into the UDDI model, WS-PolicyAttachment pre-defines one tModel that is used to associate a remotely accessible Policy with an entity in a UDDI registry.

This new tModel is called the Remote Policy Reference category system and is defined in [Appendix A.1](#).

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; that is, when referencing this category system in a categoryBag, the corresponding keyValue of the keyedReference is the URI of the Policy Expression.

Using the Remote Policy Reference category system, one can then associate a Policy Expression with a businessEntity, a businessService, and a tModel using the entity's categoryBag. For example, associating the Policy Expression 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.

A different approach has to be taken to associate a Policy Expression 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.

5.3 Registering Reusable Policy Expressions

In addition to using the approach outlined in the section above, publishers may register a specific Policy Expression in a UDDI registry as a distinct tModel. To properly categorize tModels as Policy Expressions, WS-PolicyAttachment pre-defines the WS-Policy Types category system as a tModel. This tModel is defined in [Appendix A.2](#).

The following illustrates a tModel for the Policy Expression 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">WS-Policy Expression</description>
    <overviewURL>http://www.example.com/myservice/policy</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
      keyName="Reusable policy Expression"
      keyValue="policy"
      tModelKey="uuid:fald77dc-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 – rather than only being associated with one - by using the WS-Policy Types category system's built-in category "policy", which is its single valid value. This is necessary in order to enable UDDI inquiries for Policy Expressions in general. The second keyedReference designates the Policy Expression the tModel represents by using the approach from the section above. This is necessary in order to enable UDDI inquiries for particular Policy Expressions based on their URI.

Note that the Policy Expression URI is also specified in the tModel's overview URL to indicate that it is a resolvable URL to actually retrieve the Policy Expression.

WS-PolicyAttachment pre-defines another tModel that is used to associate such a pre-registered, locally available Policy Expressions with an entity in a UDDI registry

This new tModel is called the Local Policy Reference category system and is defined in [Appendix A.3](#).

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 WS-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.

Given the Local Policy Reference category system, one can then associate a Policy Expression tModel with a businessEntity, a businessService, and a tModel using the entity's categoryBag. For example, associating the Policy Expression 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>

```

```
</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 that is registered with the UDDI registry.

A different approach has to be taken to associate a Policy Expression 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 that is registered with the UDDI registry.

5.4 Registering Policies in UDDI Version 3

UDDI Version 3 [[UDDI30](#)] 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 [[UDDI30](#)]. 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 WS-Policy Types tModel changes from "uuid:fa1d77dc-edf0-3a84-a99a-5972e434e993" to "uddi:schemas.xmlsoap.org:policytypes:2003_03".

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

Second, rather than putting Policy Expression references in a bindingTemplate's tModelInstanceInfo, they are added to the bindingTemplate's categoryBag, analogous to

the mechanism described for other UDDI entities. For example, the example bindingTemplate from section 5.1 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 tModels and UDDI entities that are associated with remote Policy Expression is enhanced by the wildcard mechanism for keyValues in keyedReferences. For example, searching for all Policy Expression tModels whose 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 [[XMLSignature](#)]. Publishers who want to digitally sign their Policy Expression tModels or Policy Expression references in UDDI MUST use the Schema-centric canonicalization algorithm [[SCC14N](#)].

6. Security Considerations

It is strongly RECOMMENDED that Policy Attachments be signed to prevent tampering. This also provides a mechanism for authenticating Policy Attachments by determining if the signer has the right to "speak for" the scope of the Policy Attachment.

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.

7. Acknowledgements

We would like to thank the following people for their contributions towards this specification: Dimitar Angelov (SAP), Martijn de Boer (SAP), Erik Christensen (Microsoft), Giovanni Della-Libera (Microsoft), Christopher Ferris (IBM), Martin Gudgin (Microsoft), Andrew Hately (IBM), Yigal Hoffner (IBM), Brian Hulse (IBM), Andrew Jones (IBM), Todd Karakashian (BEA Systems), Scott Konersmann (Microsoft), Al Lee (Microsoft), David Levin (Microsoft), Frank Leymann (IBM), Steve Lucco (Microsoft), Steve Millet (Microsoft), Nataraj Nagaratnam (IBM), Henrik Frystyk Nielsen (Microsoft), Paul Nolan (IBM), Keith Stobie (Microsoft), Tony Storey (IBM), Sanjiva Weerawarana (IBM), Volker Wiechers (SAP).

8. References

[InfoSet]

J. Cowan, et al, "[XML Information Set](#)," October 2001.

[RFC 2119]

S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," [RFC 2119](#), March 1997.

[RFC 2396]

T. Berners-Lee, et al, "Uniform Resource Identifiers (URI): Generic Syntax," [RFC 2396](#), August 1998.

[SCC14N]

S. Aissi, et al "[Schema Centric XML Canonicalization Version 1.0](#)," July 2002.

[SOAP11]

Don Box, et al, "[SOAP: Simple Object Access Protocol 1.1](#)," May 2000.

[SOAP12]

M. Gudgin, et al, "[SOAP Version 1.2 Part 1: Messaging Framework](#)," December 2002.

[UDDI API 20]

D. Ehnebuske, et al, "[UDDI Version 2.04 API](#)," July 2002.

[UDDI DataStructure20]

D. Ehnebuske, et al, "[UDDI Version 2.03 Data Structure Reference](#)," July 2002.

[UDDI 30]

T. Bellwood, et al, "[UDDI Version 3.0](#)," July 2002.

[WS-Addressing]

D. Box, et al, "[Web Services Addressing \(WS-Addressing\)](#)," August 2004.

[WS-Policy]

D. Box, et al, "[Web Services Policy Framework \(WS-Policy\)](#)," September 2004.

[WS-Security]

A. Nadalin, et al, "[Web Services Security: SOAP Message Security 1.0 \(WS-Security 2004\)](#)," March 2004.

[WSDL 1.1]

E. Christensen, et al, "[Web Services Description Language \(WSDL\) 1.1](#)," March 2001.

[WS-MetadataExchange]

K. Ballinger, et al, "[Web Services Metadata Exchange \(WS-MetadataExchange\)](#)," September 2004.

[XML-NS]

T. Bray, et al, "[Namespaces in XML](#)," January 1999.

[XML-Signature]

D. Eastlake, et al, "[XML-Signature Syntax and Processing](#)," February 2002.

[XMLSchema]

D. Beech, et al, "[XML Schema Part 1: Structures](#)," May 2001.

Appendix A: UDDI tModel Definitions

This section contains the UDDI tModel definitions for the canonical tModels used in [Section 5](#). 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.

A.1 Remote Policy Reference Category System

A1.1 Design Goals

This tModel is used to attach a Policy to a UDDI entity by referencing the Policy's URI.

A1.2 tModel Definition

Name: <http://schemas.xmlsoap.org/ws/2003/03/remotepolicyreference>

Description: Category system used for UDDI entities to point to an external WS-PolicyAttachment Policy that describes their characteristics. See WS-PolicyAttachment 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

A1.3 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 WS-PolicyAttachment Policy Expression that describes
their characteristics. See WS-PolicyAttachment specification for further
details.</description>
  <categoryBag>
    <keyedReference
      keyName="uddi-org:types:categorization"
      keyValue="categorization"
      tModelKey="uuid:clacf26d-9672-4404-9d70-39b756e62ab4" />
  </categoryBag>
</tModel>
```

```
</categoryBag>
</tModel>
```

A.2 WS-Policy Types Category System

A2.1 Design Goals

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

A2.2 tModel Definition

Name:	http://schemas.xmlsoap.org/ws/2003/03/policytypes
Description:	WS-Policy Types category system used for UDDI tModels to characterize them as WS-Policy – based Policy Expressions.
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

A2.3 tModel Structure

```
<tModel tModelKey="uuid:fa1d77dc-edf0-3a84-a99a-5972e434e993" >
  <name>http://schemas.xmlsoap.org/ws/2003/03/policytypes</name>
  <description xml:lang="EN">WS-Policy Types category system used for UDDI
tModels to characterize them as WS-Policy – based Policy
Expressions.</description>
  <categoryBag>
    <keyedReference
      keyName="uddi-org:types:categorization"
      keyValue="categorization"
      tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4" />
    </categoryBag>
</tModel>
```

A.3 Local Policy Reference Category System

A3.1 Design Goals

This tModel is used to attach a Policy Expression to a UDDI entity by referencing the UDDI entity that represents this Policy Expression. The Local Policy Reference category system is based on tModelKeys. It is expected that referenced tModels are registered

with the same UDDI registry and are categorized as representing Policy Expressions using the WS-Policy Types category system.

A3.2 tModel Definition

Name: <http://schemas.xmlsoap.org/ws/2003/03/localpolicyreference>

Description: Category system used for UDDI entities to point to a WS-Policy Policy Expression tModel that describes their characteristics. See WS-PolicyAttachment 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

A3.3 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 WS-Policy Policy Expression tModel that describes their
characteristics. See WS-PolicyAttachment specification for further
details.</description>
  <categoryBag>
    <keyedReference
      keyName="uddi-org:types:categorization"
      keyValue="categorization"
      tModelKey="uuid:clacf26d-9672-4404-9d70-39b756e62aB4" />
    <keyedReference
      keyName="uddi-org:entityKeyValues"
      keyValue="tModelKey"
      tModelKey="uuid:916b87bf-0756-3919-8eae-97dfa325e5a4" />
  </categoryBag>
</tModel>
```