Web Services Policy Attachment (WS-PolicyAttachment)

Version 1.1
28 May 2003

Authors

Don Box, Microsoft
Francisco Curbera, IBM
Maryann Hondo (editor), IBM
Chris Kaler (editor), Microsoft
Hiroshi Maruyama, IBM
Anthony Nadalin, IBM
David Orchard, BEA
Claus von Riegen, SAP
John Shewchuk, Microsoft

Copyright Notice

Abstract

This document specifies three specific attachment mechanisms for using policy expressions with existing XML Web service technologies. Specifically, we define how to associate policy expressions with WSDL type definitions and UDDI entities. We also define how to associate implementation-specific policy with all or part of a WSDL portType when exposed from a specific implementation.

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 an initial public draft release and is provided for review and evaluation only. BEA, IBM, Microsoft and SAP hope to solicit your contributions and suggestions in the near future. BEA, IBM, Microsoft and SAP make no warrantees or representations regarding the specifications in any manner whatsoever.

Table of Contents

1. Introduction

The WS-Policy specification defines an abstract policy model and an XML policy expression grammar for making policy assertions. This specification defines a general-purpose mechanism for associating policy expressions with subjects. It provides for two approaches to making the associations: the policy assertions may be defined as part of the definition of the subject or the policy assertions 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 [WSDL] and UDDI [UDDIAPI20, UDDIDataStructure20, UDDI30]. Specifically, this specification defines the following:

- How to reference policies from WSDL definitions
- How to associate policies with specific instances of WSDL services
- 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 [WSDL], UDDI service registrations [UDDIAPI20, UDDIDataStructure20, UDDI30] and SOAP message structure and message processing model [SOAP11, SOAP12]. WS-PolicyAttachment should be applicable to any version of SOAP, WSDL, or UDDI.

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/2003/03/addressing

wsdl	http://schemas.xmlsoap.org/wsdl/
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy
wsse	http://schemas.xmlsoap.org/ws/2002/12/secext
wsu	http://schemas.xmlsoap.org/ws/2002/07/utility

A normative copy of the XML Schema [XMLSchema1] for WS-PolicyAttachment constructs may be retrieved by resolving the URI "http://schemas.xmlsoap.org/ws/2002/12/policy".

In this document reference is made to the wsu:Id attribute in a utility schema (http://schemas.xmlsoap.org/ws/2002/07/utility). 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 which are used throughout this document:

Policy Assertion – A *policy assertion* represents an individual preference, requirement, capability or other property.

Policy Expression – A *policy expression* is an XML Infoset representation of one or more policy assertions.

Policy Subject – A *policy subject is* an entity (e.g., an endpoint, object, or resource) to which a policy can be bound.

Policy Attachment – The mechanism for associating policy with one or more subjects is referred to as *policy attachment*.

2.4 Example WS-Policy Expressions

This specification defines several mechanisms for associating WS-Policy [WS-Policy] expressions with various XML Web Service entities. For brevity, we define three sample policy expressions that the remainder of this document reference. Note that these policies include custom (fictitious) assertions using XML namespace extensibility to express domain specific policies. The expressions here indicate different policies. The first one indicates a policy for digitally signing an element. The second is a policy for message visibility. The third is a policy for delivery.

```
<wsp:Policy xmlns:wsp="..."
            xmlns:wsu="..."
            xmlns:x="..."
            xml:base="http://www.fabrikam123.com/policies"
            wsu:Id="P1"
            Name="Q1"
            TargetNamespace="http://www.fabrikam123.com/policies">
     <wsse:Integrity>
        <wsse:Algorithm Type="wsse:AlgSignature"
                 URI="http://www.w3.org/2000/09/xmlenc#aes" />
```

```
      </wsse:Integrity>
</wsp:Policy>
```

```
<wsp:Policy xmlns:wsp="..."
            xmlns:wsu="..."
            xmlns:x="..."
            xml:base="http://www.fabrikam123.com/policies"
            wsu:Id="P2"
            Name="Q2"
            TargetNamespace="http://www.fabrikam123.com/policies">
   <wsse:Visibility wsp:Usage="wsp:Required">
        <MessageParts>
           wsp:GetInfosetForNode(wsp:GetBody(.))
        </MessageParts>
   </wsse:Visibility>
</wsp:Policy>
```

```
<wsp:Policy xmlns:wsp="..."
            xmlns:wsu="..."
            xmlns:x="..."
            xml:base="http://www.fabrikam123.com/policies"
            wsu:Id="P3"
            Name="Q3"
            TargetNamespace="http://www.fabrikam123.com/policies">
  <x:ExpeditedDelivery wsp:Usage="wsp:Required" />
</wsp:Policy>
```

The URI for these policy expressions are http://www.fabrikam123.com/policies#P1, http://www.fabrikam123.com/policies#P2, and http://www.fabrikam123.com/policies#P3 respectively. The QName identifiers for the policies expressions (assuming xmlns:f123=http://www.fabrikam123.com/policies) are f123:Q1, f123:Q2, and f123:Q3 respectively.

3. Policy Attachment

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

3.1 XML Element Attachment

It is often desirable to associate policy expressions with XML elements. To facilitate this, this specification proposes an XML (Infoset) [Infoset] augmentation for element information items. The name of this augmentation is [Element Policy], and its type is a policy expression as defined in WS-Policy. The precise semantics of how [Element Policy] is to be processed once discovered is schema-specific, however, implementations are likely to follow the precedent specified in the section below on WSDL [WSDL] and policy.

This specification defines two global attributes that allow policy expressions to be attached with an arbitrary XML element. The following is the schema definition for the policy attachment attributes:

```
<xs:schema>
  <xs:attribute name="PolicyURIs" type="wsp:tPolicyURIs" />
  <xs:attribute name="PolicyRefs" type="wsp:tPolicyRefs" />
</xs:schema>
```

The namespace [XML-NS] URI for these attributes is http://schemas.xmlsoap.org/ws/2002/12/policy.

The *wsp:PolicyURIs* attribute contains a list of one or more URIs. Similarly, the *wsp:PolicyRefs* contains a list of one or more QNames. When either of these attributes is used, each of the values identifies a policy expression as defined in WS-Policy. If more than one URI or QName is specified, the individually referenced policy expressions need to be merged together to form a single effective policy expression. The resultant (potentially merged) policy expression is then associated with the element information item's [Element Policy] property. An example of this is given below using the sample policies stated in Section 2.4.

It should be noted that an implementation using this policy reference mechanism is not required to support both URIs and QNames. That is, it MAY choose to support URIs only, QNames only, or support both.

If the following XML element

```
<MyElement xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
           wsp:PolicyURIs="http://www.fabrikam123.com/policies#P1
                           http://www.fabrikam123.com/policies#P3" />
```

has processed the merge, it would result in an [Element Policy] whose XML 1.0 representation is:

```
<wsp:Policy>
    <wsse:Integrity wsp:Usage=”wsp:Required”>
         <wsse:Algorithm Type="wsse:AlgSignature"
                  URI="http://www.w3.org/2000/09/xmlenc#aes" />
    </wsse:Integrity>
    <x:ExpeditedDelivery wsp:Usage="wsp:Required">
      ...
    </x:ExpeditedDelivery>
```

```
</wsp:Policy>
```

Note that the [Element Policy] expression has no meaningful URI.

The existence of either the *wsp:PolicyURIs* or *wsp:PolicyRefs* attributes does not prohibit implementations from using additional mechanisms for associating policy expressions with XML-based constructs.

Using this global attribute, it is possible to augment the elements of a WSDL description.

3.2 Arbitrary Resource Attachment

This mechanism allows a policy expression to be associated with a resource independent of its definition and/or representation using a <wsp:PolicyAttachment> element. This new element has three components: the scope of the attachment, the policy expression being bound, and optional security information. The scope of the attachment is defined using an extensible scope expression that identifies Web-based resources, typically based on URIs.

Scope expressions identify the domain of the association. That is, the set of resources that will be considered for inclusion in the scope.

The domain of a scope expression is identified using an extensible domain expression model. Domain expressions identify resources for consideration. Domain expressions yield an unordered set of resources for consideration.

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 schema outlined above:

/PolicyAttachment

This describes a directed policy attachment.

/PolicyAttachment/AppliesTo

This required element indicates the domain of the binding's scope expression.

/PolicyAttachment/AppliesTo/{any}

Other child elements MUST specify and/or refine the domain expression(s) and MUST NOT contradict the semantics of the parent element; if an element is not recognized, it SHOULD be ignored. Domain expressions have unique XML elements that contribute resources to a domain. When more than one domain expression is present, the domain contains the union of the results of each expression. This document defines one domain expression; others may be defined in subsequent specifications.

/PolicyAttachment/Policy

This policy expression contains the set of policy operators and assertions that is being applied to all in-scope resources.

/PolicyAttachment/PolicyReference

This element references the policy expression that is being applied to all in-scope resources. Refer to WS-Policy for additional details.

/PolicyAttachment/wsse:Security

This element allows security information such as signatures to be included. The syntax of this element is described in WS-Security.

/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.

/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 declaration with a domain expression for a deployed WSDL 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/acct-policy.xml" />
  <wsse:Security>
     <ds:Signature> ...
    </ds:Signature>
  </wsse:Security>
</wsp:PolicyAttachment>
```

In this example, the policy expression at http://www.fabrikam123.com/acct-policy.xml applies to all output resources of a service which implement the fabrikam:InventoryPortType on the service at http://www.fabrikam123.com/acct.

4. Attaching Policies to WSDL

The preferred means to attach a policy [WS-Policy] to WSDL [WSDL] is to place it within the WSDL component corresponding to the target. However, because WSDL/1.1 disallows the application of extensibility elements to portTypes and messages, this specification defines the use of *wsp:PolicyURIs* and *wsp:PolicyRefs* attributes.

To ensure that consumers of a policy-annotated WSDL document are capable of processing the 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 portType or message that uses the policy attachment mechanism defined in this section. Moreover, when the <wsp:UsingPolicy> element appears under a <wsdl:definitions> element, it MUST be marked as a mandatory extension (e.g., with a *wsdl:required="true"* attribute).

This section defines how to interpret the policy when it appears within a WSDL portType, a message, a service or a port definition. In all these cases, the hierarchical nature of WSDL is taken into account to determine the *effective policy* for a given aspect of an abstract messaging type.

4.1 Effective Policy and Element Policy

When attaching policies to different levels of the hierarchy [Elements], care must be taken. Policies attached at a higher level are inherited. It is RECOMMENDED that additional policies at lower levels only further qualify inherited policies.

This specification defines an augmentation to the WSDL data model. This augmentation is called the [Effective Policy] and its definition and interpretation is the focus of the remainder of this section.

In general, the [Effective Policy] is inherited from parent nodes (e.g., an operation inherits [Effective Policy] from its parent portType). Assertion types that only apply to the parent's node type are ignored when calculating [Effective Policy].

For WSDL type definitions (portTypes and messages), the [Effective Policy] for a WSDL type definition is considered an intrinsic part of the type definition and applies to all uses of that type, in particular, to every implementation of that WSDL portType.

In the case of policies that apply to deployed resources (services or ports), the [Effective Policy] applies only to the deployed resource itself.

4.2 Policy and WSDL messages

WSDL/1.1 defines two elements that collectively are used to define a WSDL message type:

- wsdl:message
- wsdl:message/wsdl:part

Each of these element types MAY have an [Element Policy] property per Section 3 of this specification. The remainder of this section defines how that [Element Policy] is interpreted to calculate the [Effective Policy].

The [Effective Policy] for a WSDL message type is the [Element Policy] of the wsdl:message element that defines the message type.

The [Effective Policy] for a WSDL message part is the [Element Policy] of the wsdl:part element that defines the message part merged with the effective policy of the part's parent message type.

Consider the following WSDL message type definition which references policies using QNames:

```
<?xml version="1.0"?>
<wsdl:definitions name="Inventory"
                  xmlms:wsdl="http://schemas.xmlsoap.org/wsdl"
```

```
                  targetNamespace="http://example.com/inventory.wsdl"
                  ...
                  xmlns:ps="http://example.com/policies">
  <wsp:UsingPolicy wsdl:Required="true" />
  <wsdl:message name="LookupResponse"
                wsp:PolicyRefs="ps:Q1">
    <wsdl:part name="key"
               type="xs:string"
               wsp:PolicyRefs="ps:Q2" />
    <wsdl:part name="value"
               type="xs:string"
               wsp:PolicyRefs="ps:Q3" />
    <wsdl:part name="hint" type="xs:string" />
  </wsdl:message>
</wsdl:definitions>
```

The LookupResponse message type has an [Effective 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>
</wsp:Policy>
```

The “key” part of the LookupResponse message type has an [Effective 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:Visibility wsp:Usage="wsp:Required">
        <MessageParts>
           wsp:GetInfosetForNode(wsp:GetBody(.))
        </MessageParts>
  </wsse:Visibility>
</wsp:Policy>
```

The “value” part of the LookupResponse message type has an [Effective 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>
  <x:ExpeditedDelivery wsp:Usage="wsp:Required" />
</wsp:Policy>
```

Because the “hint” part of the LookupResponse message type has no [Element Policy], the [Effective Policy] of the “hint” message part is:

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

4.3 Policy and WSDL portTypes

WSDL/1.1 defines five elements that collectively are used to define a portType:

- wsdl:portType
- wsdl:portType/wsdl:operation
- wsdl:portType/wsdl:operation/wsdl:input
- wsdl:portType/wsdl:operation/wsdl:output
- wsdl:portType/wsdl:operation/wsdl:fault

Each of these element types MAY have an [Element Policy] property per Section 3 of this specification. Care should be taken when attaching policies to outbound operations to make sure the result is what is expected.

The remainder of this section defines how that [Element Policy] is interpreted to calculate the *effective* policy.

The [Effective Policy] for a WSDL portType is the [Element Policy] of the `wsdl:portType` element that defines the portType.

The [Effective Policy] for a WSDL operation is the [Element Policy] of the `wsdl:operation` element that defines the operation merged with the [Effective Policy] of the operation's parent portType.

The [Effective Policy] for a WSDL input message is the [Element Policy] of the `wsdl:input` element that defines the input message merged with the [Effective Policy] of the message's parent operation and the [Effective Policy] of the referenced WSDL message type.

The [Effective Policy] for a WSDL output message is the [Element Policy] of the `wsdl:output` element that defines the output message merged with the [Effective

Policy] of the message's parent operation and the [Effective Policy] of the referenced WSDL message type.

The [Effective Policy] for a WSDL fault message is the [Element Policy] of the `wsdl:fault` element that defines the fault message merged with the [Effective Policy] of the message's parent operation and the [Effective Policy] of the referenced WSDL message type.

4.4 Policy and WSDL bindings

WSDL/1.1 defines five elements that collectively are used to define a binding:

- wsdl:binding
- wsdl:binding/wsdl:operation
- wsdl:binding/wsdl:operation/wsdl:input
- wsdl:binding/wsdl:operation/wsdl:output
- wsdl:binding/wsdl:operation/wsdl:fault

Each of these element types MAY have an [Element Policy] property per Section 3 of this specification. Care should be taken when attaching policies to outbound operations to make sure the result is what is expected.

The remainder of this section defines how that [Element Policy] is interpreted to calculate the *effective* policy.

The [Effective Policy] for a WSDL binding is the [Element Policy] of the `wsdl:binding` element that defines the binding merged with the [Effective Policy] of the referenced `wsdl:portType` element.

The [Effective Policy] for a WSDL operation binding is the [Element Policy] of the `wsdl:binding/wsdl:operation` element that defines the operation binding merged with the [Effective Policy] of the operation's parent binding and the [Effective Policy] of the (implicitly) referenced `wsdl:portType/wsdl:operation` element.

The [Effective Policy] for a WSDL input message binding is the [Element Policy] of the `wsdl:binding/wsdl:operation/wsdl:input` element that defines the input message binding merged with the [Effective Policy] of the parent operation binding and the [Effective Policy] of the (implicitly) referenced `wsdl:portType/wsdl:operation/wsdl:input` element.

The [Effective Policy] for a WSDL output message binding is the [Element Policy] of the `wsdl:binding/wsdl:operation/wsdl:output` element that defines the output message binding merged with the [Effective Policy] of the parent operation binding and the [Effective Policy] of the (implicitly) referenced `wsdl:portType/wsdl:operation/wsdl:output` element.

The [Effective Policy] for a WSDL fault message binding is the [Element Policy] of the `wsdl:binding/wsdl:operation/wsdl:fault` element that defines the fault message binding merged with the [Effective Policy] of the parent operation binding and the [Effective Policy] of the (implicitly) referenced `wsdl:portType/wsdl:operation/wsdl:fault` element.

4.5 Policy and WSDL services and ports

In this section we show how a WSDL service or port definition can indicate policies that reference a list of policies that apply to them using the *wsp:PolicyURIs* and *wsp:PolicyRefs* attributes.

Two elements are used in WSDL/1.1 to define a WSDL service:

- wsdl:service
- wsdl:service/port

Additional extensibility elements from different namespaces are allowed under wsdl:service/port, to indicate transport and address information. We will not concern ourselves with these.

The [Effective Policy] for a WSDL service is the [Element Policy] of the `wsdl:service` element that defines the service.

The [Effective Policy] for a WSDL port is the [Element Policy] of the `wsdl:port` element that defines the port merged with the [Effective Policy] of the port's parent service and the [Effective Policy] of the referenced WSDL binding.

4.6 Referencing Deployed Endpoints

Attaching policies to WSDL allows the annotation of the Web Service description. However, WSDL currently does not allow a reference to the instance of a deployed service. This section illustrates a new *domain expression* based on WS-Addressing [WS-Addressing] allows the use of the `<wsp:PolicyAttachment>` mechanism to reference a specific endpoint on a deployed Web service.

The following schema outline illustrates this extension:

```
<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsa:EndpointReference>... </wsa:EndpointRefernce>
  </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 AppliesTo element.

5. Registering Policies in UDDI

This section defines a minimum level of support for associating WS-Policy - based policy expressions with entities in a UDDI registry. While the general concept that is specified in sections 5.1 and 5.2 is based on UDDI Version 2 [UDDIAPI20, UDDIDataStructure20], the necessary changes with respect to UDDI Version 3 [UDDI30] are explained in section 5.3.

There are essentially two approaches for registering policies in UDDI. The one 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 5.1) is expected to be used for policy expressions that are mainly unique for a given

Web service, the latter approach (see section 5.2) is expected to be used for more modular and reusable policy expressions.

5.1 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 expression with an entity in a UDDI registry.

This new tModel is defined as follows:

```
<tModel tModelKey="uuid:0b1b5a47-bebf-3b7d-9802-f2dd80a91adebd3966a8-faa5-
416e-9772-128554343571">

<name>http://schemas.xmlsoap.org/ws/20023/03/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:C1ACF26D-9672-4404-9D70-39B756E62AB4" />
  </categoryBag>
</tModel>
```

UDDI registries MUST use the tModelKey uuid:0b1b5a47-bebf-3b7d-9802-f2dd80a91ade to uniquely identify this tModel so that UDDI registry users can expect the same behavior across different UDDI registries.

The keyedReference specifies that the tModel is a category system. The valid values of this category system are those URIs that identify external policy expressions, that is, when referencing this category system in a category bag, the corresponding keyValue of the keyedReference is the URI of the policy expression.

Given the tModel defined above, 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 at 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 servives"
        keyValue="http://www.example.com/myservice/policy"
        tModelKey="uuid:0b1b5a47-bebf-3b7d-9802-f2dd80a91ade" />
  </categoryBag>
</tModel>
```

The tModelKey of the keyedReference must match the fixed tModelKey from the remote policy reference category system as defined above. The keyValue MUST be the URI of 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:0b1b5a47-bebf-3b7d-9802-f2dd80a91ade" >
      <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 of the policy expression.

5.2 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. The following illustrates a tModel for the policy expression at 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="reusable"
        tModelKey="uuid:0b1b5a47-bebf-3b7d-9802-f2dd80a91ade" />
    <keyedReference
        keyName="Policy expression for example’s Web services"
        keyValue="http://www.example.com/myservice/policy"
        tModelKey="uuid:0b1b5a47-bebf-3b7d-9802-f2dd80a91ade" />
  </categoryBag>
</tModel>
```

The first keyedReference specifies that the tModel is in fact a policy expression – rather than only being associated with one - by using the remote policy reference category system’s built-in category “reusable”. 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 reference to the policy expression is also specified in the tModel’s overviewURL for those UDDI users that want to use the overviewURL in order to programmatically access the tModel’s documentation.

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

This new tModel is defined as follows:

```
<tModel tModelKey="uuid:0c41b09e-07c7-321f-93a6-3ab354fa84f8" >
  <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:C1ACF26D-9672-4404-9D70-39B756E62AB4" />
  </categoryBag>
```

```
</tModel>
```

UDDI registries MUST use the tModelKey uuid:0c41b09e-07c7-321f-93a6-3ab354fa84f8 to uniquely identify this tModel so that UDDI registry users can expect the same behavior across different UDDI registries.

The keyedReference specifies that the tModel is a category system. The valid values of this category system are those tModelKeys identifying tModels that

- exist in the same UDDI registry
- and are categorized as being “reusable” using the remote policy reference category system.

That is, when referencing this category system in a category bag, the corresponding keyValue of the keyedReference is the tModelKey of the policy expression.

Given this new tModel, 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:0c41b09e-07c7-321f-93a6-3ab354fa84f8" />
  </categoryBag>
</businessService>
```

The tModelKey of the keyedReference MUST match the fixed tModelKey from the local policy reference category system as defined above. 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:0c41b09e-07c7-321f-93a6-3ab354fa84f8" >
      <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 as defined above. The instanceParms MUST be the tModelKey of the policy expression that is registered with the UDDI registry.

5.3 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 specified Version 2 keys were already programmatically derived from the Version 3 keys given below.

The tModelKey for the remote policy reference tModel changes from "uuid:0b1b5a47-bebf-3b7d-9802-f2dd80a91ade" to "uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03".

The tModelKey for the local policy reference tModel changes from "uuid:0c41b09e-07c7-321f-93a6-3ab354fa84f8" to "uddi:schemas.xmlsoap.org:localpolicyreference:2003_03".

Second, since the local policy reference tModel's valid values are the tModelKeys of policy reference tModels in UDDI Version 2, and entity key formats change in UDDI Version 3 (at least, the tModelKey prefix "uuid:" is changed to "uddi:"), a separate local policy reference tModel is needed in order to build tModel references based on Version 3 format keys. This tModel is defined as follows:

```
<tModel
  tModelKey="uddi:schemas.xmlsoap.org:localpolicyreference_v3:2003_03" >

<name>http://schemas.xmlsoap.org/ws/2003/03/localpolicyreference_v3</name>
  <description xml:lang="en">Category system used for UDDI entities to
point to a WS-Policy policy expression tModel that describes their
characteristics using Version 3 format keys. See WS-PolicyAttachment
specification for further details.</description>
  <categoryBag>
    <keyedReference
      keyName="uddi-org:types:categorization"
      keyValue="categorization"
      tModelKey="uddi:uddi.org:categorization:types" />
  </categoryBag>
</tModel>
```

Corresponding keyedReferences in UDDI entity categoryBags are constructed accordingly. For example, associating the policy expression tModel with the tModelKey "uuid:04cfa..." with a businessService is done as follows (assuming that the tModel's tModelKey has a Version 3 format of "uddi:04cfa…"):

```
<businessService serviceKey="..." >
  <name>...</name>
  <description>...</description>
  <bindingTemplates>...</bindingTemplates>
  <categoryBag>
    <keyedReference
      keyName="Policy expression for example's Web services"
      keyValue="uddi:04cfa..."
      tModelKey="uddi:schemas.xmlsoap.org:localpolicyreference_v3:2003_03"
/>
  </categoryBag>
</businessService>
```

Third, 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 businessServices. 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>
```

Fourth, inquiries for reusable policy expression tModels and UDDI entities that are associated with remote policy expressions is enhanced by the wildcard mechanism for keyValues in keyedReferences. For example, searching for all policy expressions 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>
</find_tModel>
```

Fifth, all UDDI entities may be digitally signed using XML digital signatures [XMLSignature]. Publishers who want to digitally sign their policy reference tModels or policy 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:

Martijn de Boer, SAP
Erik Christensen, Microsoft
Giovanni Della-Libera, Microsoft
Martin Gudgin, Microsoft
Andrew Hately, IBM
Scott Konersmann, Microsoft
Frank Leymann, IBM
Steve Lucco, Microsoft
Steve Millet , Microsoft
Nataraj Nagaratnam, IBM
Henrik Frystyk Nielsen, Microsoft
Keith Stobie, Microsoft
Tony Storey, IBM
Sanjiva Weerawarana, IBM

8. References

[Infoset]
"XML Information Set," W3C Recommendation, John Cowan and Richard Tobin (editors), 24 October 2001.

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

[RFC 2396]
"Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, T. Berners-Lee, R. Fielding, and L. Masinter (editors), August 1998.

[SCC14N]
"Schema Centric XML Canonicalization Version 1.0," OASIS Committee Specification, Selim Aissi, Bob Atkinson, and Maryann Hondo, 10 July 2002.

[SOAP11]
"SOAP: Simple Object Access Protocol 1.1," W3C Note, Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer, 08 May 2000.

[SOAP12]
"SOAP Version 1.2 Part 1: Messaging Framework," W3C Candidate Recommendation, Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen (editors), 19 December 2002.

[UDDIAPI20]
"UDDI Version 2.04 API," OASIS Committee Specification, Dave Ehnebuske, Barbara McKee, and Dan Rogers (editors), 19 July 2002.

[UDDIDataStructure20]
"UDDI Version 2.03 Data Structure Reference," OASIS Committee Specification, David Ehnebuske, Dan Rogers, Claus von Riegen (editors), 19 July 2002.

[UDDI30]
"UDDI Version 3.0," OASIS Committee Specification, Tom Bellwood, Luc Clément, David Ehnebuske, Andrew Hately, Maryann Hondo, Yin Leng Husband, Karsten Januszewski, Sam Lee, Barbara McKee, Joel Munter, and Claus von Riegen (editors), 19 July 2002.

[WS-Addressing]
"Web Services Addressing (WS-Addressing)," Adam Bosworth, Don Box (Editor), Erik Christensen, Francisco Curbera (Editor), Donald Ferguson, Jeffrey Frey, Chris Kaler, David Langworthy, Frank Leymann, Steve Lucco, Steve Millet, Nirmal Mukhi, Mark Nottingham, David Orchard, John Shewchuk, Tony Storey, and Sanjiva Weerawarana, 13 March 2003.

[WS-Policy]
"Web Services Policy Framework (WS-Policy)," Don Box, Francisco Curbera, Maryann Hondo (Editor), Chris Kaler (Editor), Dave Langworthy, Anthony Nadalin, Nataraj Nagaratnam, Mark Nottingham, Claus von Riegen, and John Shewchuk, March 2003.

[WS-Security]
"Web Services Security (WS-Security)," Bob Atkinson, Giovanni Della-Libera, Satoshi Hada, Maryann Hondo, Phillip Hallam-Baker, Chris Kaler (Editor), Johannes Klein, Brian LaMacchia, Paul Leach, John Manferdelli, Hiroshi Maruyama, Anthony Nadalin, Nataraj Nagaratnam, Hemma Prafullchandra, John Shewchuk, and Dan Simon, 5 April 2002.

[WSDL]
"Web Services Description Language (WSDL) 1.1," W3C Note, Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana, 15 March 2001.

[XML-NS]
"Namespaces in XML," W3C Recommendation, Tim Bray, Dave Hollander, and Andrew Layman (editors), 14 January 1999.

[XML-Signature]
"XML-Signature Syntax and Processing," W3C Recommendation, Donald Eastlake, Joseph Reagle, and David Solo (editors), 12 February 2002.

[XMLSchema1]
"XML Schema Part 1: Structures," W3C Recommendation, Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn (editors), 2 May 2001.