Understanding WSDL in a UDDI registry, Part 1

How to publish and find WSDL service descriptions

Peter Brittenham (peterbr@us.ibm.com)
Senior software engineer
IBM Emerging Technologies

01 September 2001

Francisco Cubera (curbera@us.ibm.com)
Research staff member
IBM T.J. Watson Research Center

Dave Ehnebuske (davide@us.ibm.com)
Distinguished engineer
IBM Software Group

Steve Graham
Web services architect
IBM Software Group

The Web Services Description Language has a lot of versatility in its methods of use. In particular, WSDL can work with UDDI registries in several different ways depending upon the application needs. In this first of a three-part series, we will look at these different methods of using WSDL with UDDI registries.

The Web Services Description Language (WSDL) is an XML language for describing Web services as a set of network endpoints that operate on messages. A WSDL service description contains an abstract definition for a set of operations and messages, a concrete protocol binding for these operations and messages, and a network endpoint specification for the binding.

Universal Description Discovery and Integration (UDDI) provides a method for publishing and finding service descriptions. The UDDI data entities provide support for defining both business and service information. The service description information defined in WSDL is complementary to the information found in a UDDI registry. UDDI provides support for many different types of service descriptions. As a result, UDDI has no direct support for WSDL or any other service description mechanism.

The UDDI organization, UDDI.org, has published a best practices document titled *Using WSDL in a UDDI Registry 1.05* (see Resources). This best practices document describes some of the elements on how to publish WSDL service descriptions in a UDDI registry. The purpose of this article is to augment that information. The primary focus is on how to map a complete WSDL service description into a UDDI registry, which is required by existing WSDL tools and runtime environments. The information in this article adheres to the procedures outlined in that best practices document and is consistent with the specifications for WSDL 1.1, UDDI 1.0, and UDDI 2.0 (see Resources).

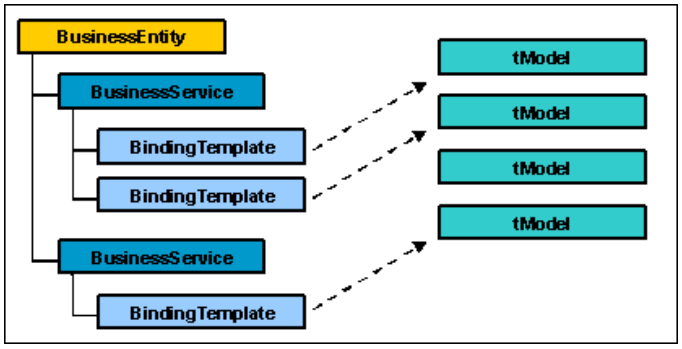
An Overview of UDDI and WSDL

Before describing the process for mapping WSDL service descriptions into a UDDI registry, it is important to understand the UDDI data types and the primary WSDL document types.

UDDI data types

There are four primary data types in a UDDI registry: *businessEntity*, *businessService*, *bindingTemplate*, and *tModel*. Figure 1 shows the relationship between all of these data types.

Figure 1. UDDI data types

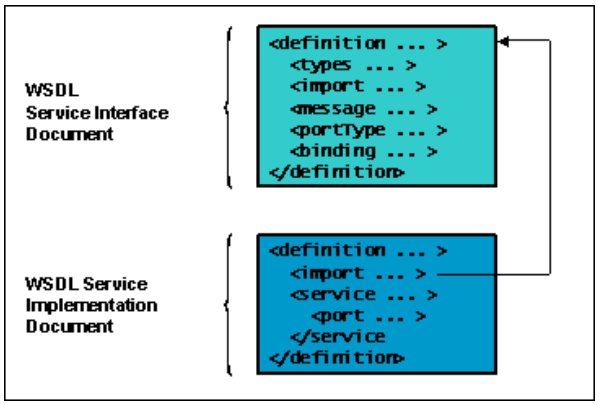


The businessEntity provides information about a business, and can contain one or more businessServices. The business is the service provider. The technical and business descriptions for a Web service are defined in a businessService and its bindingTemplates. Each bindingTemplate contains a reference to one or more tModels. A tModel is used to define the technical specification for a service.

WSDL document types

To assist with publishing and finding WSDL service descriptions in a UDDI Registry, WSDL documents are divided into two types: *service interfaces* and *service implementations* (see Figure 2).

Figure 2. WSDL document types



A service interface is described by a WSDL document that contains the *types*, *import*, *message*, *portType*, and *binding* elements. A service interface contains the WSDL service definition that will be used to implement one or more services. It is an abstract definition of a Web service, and is used to describe a specific type of service.

A service interface document can reference another service interface document using an import element. For example, a service interface that contains only the message and portType elements can be referenced by another service interface that contains only bindings for the portType.

The WSDL service implementation document will contain the *import* and *service* elements. A service implementation document contains a description of a service that implements a service interface. At least one of the import elements will contain a reference to the WSDL service interface document. A service implementation document can contain references to more than one service interface document.

The import element in a WSDL service implementation document contains two attributes. The *namespace* attribute value is a URL that matches the *targetNamespace* in the service interface document. The *location* attribute is a URL that is used to reference the WSDL document that contains the complete service interface definition. The *binding* attribute on the port element contains a reference to a specific binding in the service interface document.

The service interface document is developed and published by the *service interface provider*. The service implementation document is created and published by the *service provider*. The roles of the service interface provider and service provider are logically separate, but they can be the same business entity.

Publishing and finding WSDL descriptions

This section describes the process for publishing and finding a complete WSDL service description. A complete WSDL service description is a combination of a service interface and a service implementation document.

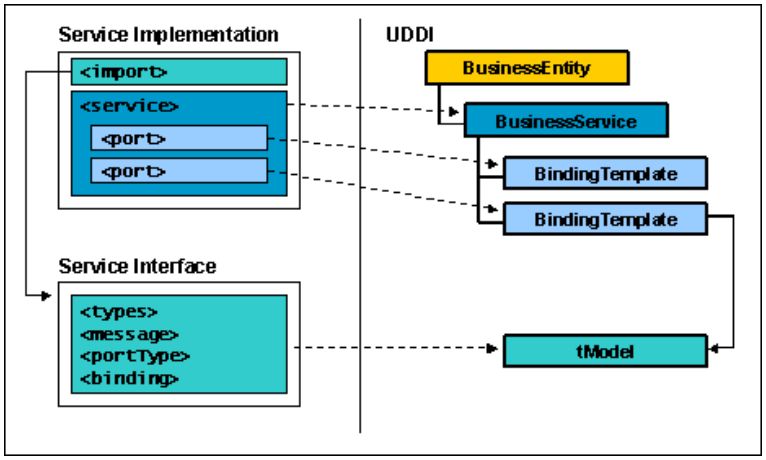
Since the service interface represents a reusable definition of a service, it is published in a UDDI registry as a tModel. The service implementation describes instances of a service. Each instance

is defined using a WSDL service element. Each service element in a service implementation document is used to publish a UDDI businessService.

When publishing a WSDL service description, a service interface must be published as a tModel before a service implementation is published as a businessService.

Figure 3 contains an overview of the mapping from WSDL to UDDI. This mapping will be described in the following sections.

Figure 3. Overview of WSDL to UDDI mapping



Publishing service interfaces

A service interface is published as a tModel in a UDDI registry. The tModel is published by the service interface provider. Some elements in the tModel are constructed using the information from the WSDL service interface description.

UDDI tModel

Table 1 defines the tModel creation steps. A valid tModel reference to a WSDL service interface definition should be named using the targetNamespace and must contain the overviewURL and categoryBag settings.

Table 1: Steps for tModel creation

	UDDI tModel	WSDL Service Interface	Description	Required
1	name	targetNamespace attribute for definitions element	The tModel name is set using the target namespace for the service interface document. A consistent name is needed to ensure that the tModel can be located using just the information from a service implementation document.	Yes
2	description	documentation element within the definitions element	The tModel description element is restricted to 256 characters. The English	No

			value for this element can be set from the first 256 characters of the documentation element that is associated with the definitions element in the service interface document. If the documentation element does not exist, then the name attribute from the definitions element should be used.	
3	overviewURL	[Service interface document URL and binding specification]	The location for the service interface document must be set in the overviewURL element. If there is more than one binding in the service interface document, then the binding must be encoded in the URL.	Yes
4	categoryBag	[Not applicable]	The categoryBag for the tModel must contain at least one keyed reference. This keyed reference must contain a reference to the `uddi-org:types` tModel and the key name must be `wsdlSpec`. This entry identifies this tModel as a WSDL service interface definition.	Yes

Example of service interface to tModel mapping

Listing 1 contains an example WSDL service interface document. The values that are mapped into a UDDI tModel are shown in the diagram key.

Listing 1

```
<?xml version="1.0"?>
<definitions name="StockQuoteService-interface"
  targetNamespace="http://www.getquote.com/StockQuoteService-interface"
  xmlns:tns="http://www.getquote.com/StockQuoteService-interface"
  xmlns:xsd=" http://www.w3.org/2001/XMLSchema "
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <documentation>
    Standard WSDL service interface definition for a stock quote service.
  </documentation>

  <message name="SingleSymbolRequest">
    <part name="symbol" type="xsd:string"/>
  </message>

  <message name="SingleSymbolQuoteResponse">
    <part name="quote" type="xsd:string"/>
  </message>

  <portType name="SingleSymbolStockQuoteService">
    <operation name="getQuote">
      <input message="tns:SingleSymbolRequest"/>
      <output message="tns:SingleSymbolQuoteResponse"/>
```

```
    </operation>
  </portType>

  <binding name="SingleSymbolBinding"
           type="tns:SingleSymbolStockQuoteService">
    <soap:binding style="rpc"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getQuote">
      <soap:operation soapAction="http://www.getquote.com/GetQuote"/>
      <input>
        <soap:body use="encoded"
            namespace="urn:single-symbol-stock-quotes"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
            namespace="urn:single-symbol-stock-quotes"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>
</definitions>
```

Definitions -The targetNamespace will be used as the name for the tModel. The contents of the documentation element will be used for the description for the tModel.
Binding - The binding name will be used to qualify the overviewURL.

Listing 2 contains the UDDI tModel that is created when the WSDL service interface definition is published. The values are shown in the diagram key.

Listing 2

```
<?xml version="1.0"?>
<tModel tModelKey="">
  <name>http://www.getquote.com/StockQuoteService-interface</name>

  <description xml:lang="en">
    Standard service interface definition for a stock quote service.
  </description>

  <overviewDoc>
    <description xml:lang="en">
      WSDL Service Interface Document
    </description>
    <overviewURL>
      http://www.getquote.com/services/
SQS-interface.wsdl#SingleSymbolBinding
    </overviewURL>
  </overviewDoc>

  <categoryBag>
    <keyedReference tModelKey="UUID:C1ACF26D-9672-4404-9D70-39B756E62AB4"
                    keyName="uddi-org:types" keyValue="wsdlSpec"/>
    <keyedReference tModelKey="UUID:DB77450D-9FA8-45D4-A7BC-04411D14E384"
                    keyName="Stock market trading services"
                    keyValue="84121801"/>
  </categoryBag>
</tModel>
```

tModel - The tModel name is set from the targetNamespace. The description is set from the documentation element that is associated with the definitions element. The

overviewURL is set to the network accessible location for the WSDL sbrvice interface document which is http://www.getquote.com/services/SQS-interface.wsdl. It also contains a direct reference to the binding named SingleSymbolBinding in the service interface document. Since this is the only binding in the service interface document, this reference to the binding is not required. The categoryBag contains the wsdlSpec entry, as well as any other keyedReferences which indicate the intended business use for this service interface description.

Publishing service implementations

A service implementation is published in a UDDI registry as a businessService with one or more bindingTemplates. The businessService is published by the service provider.

UDDI businessService

A new businessService will be created for each service element that is defined in the service implementation document. The following table contains the list of businessService elements that can be created based on the contents of the WSDL service implementation document.

	UDDI businessService	Description	Required
1	name	The name element for the businessService is set from the name attribute from the service element in the service implementation document	Yes
2	description	The description element is set from the contents of the documentation element within a service element. The English value for the description element is set from the first 256 characters in the documentation element that is associated with the service element. If a documentation element doesn't exist, then the description element for the businessService is not set.	No

UDDI bindingTemplate

A new bindingTemplate element is created within a businessService for each port element that is defined within a service element.

	UDDI bindingTemplate	Description	Required
1	description	If the port element contains a documentation element, then one description element is set from the first 256 characters of the documentation element.	No
2	accessPoint	For a SOAP or HTTP binding, the accessPoint is set from the location attribute on the extension element that is associated with the port element. The element will contain the URL, and the	Yes

		URLType attribute is set based on the protocol in this URL. For protocol bindings that don't use a URL specification, the URLType attribute should be used to indicate that type of protocol binding and the accessPoint element should contain a value that could be used to locate the Web service using the specified protocol.	
3	tModelInstanceInfo	The bindingTemplate will contain one tModelInstanceInfo element for each tModel that it references. There will be at least one tModelInstanceInfo element which will contain a direct reference to the tModel that represents the service interface document.	Yes
4	overviewURL	The overviewURL element may contain a direct reference to the service implementation document. The reference to this document is used only to provide access to human readable documentation. All of the other information within this document should be accessible through a UDDI data entity. By maintaining a direct reference to the original WSDL document, you are assured that the document that is published is the same one that it is returned during a find operation. If this document contains more than one port, then this element should contain a direct reference to the port name. It is not sufficient to use the binding reference in the tModel, since there can be more than one port that references the same binding. The port name is specified as a fragment identifier on the overviewURL. The fragment identifier is an extension to the URL using a "#" character as a separator.	No

Example of service implementation to businessService mapping

Listing 3 contains an example of a WSDL service implementation document. The highlighted values are required when mapping the WSDL information to a UDDI businessService and bindingTemplates. The values are shown in the diagram key.

Listing 3

```
<?xml version="1.0"?>
<definitions name="StockQuoteService"
  targetNamespace="http://www.getquote.com/StockQuoteService"
  xmlns:interface="http://www.getquote.com/StockQuoteService-interface"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
```

```
  <documentation>
    This service provides an implementation of a standard stock quote service.
    The Web service uses the live stock quote service provided by XMLtoday.com.
    The XMLtoday.com stock quote service uses an HTTP GET interface to request
    a quote, and returns an XML string as a response.

    For additional information on how this service obtains stock quotes, go to
    the XMLtoday.com web site: http://www.xmltoday.com/examples/soap/stock.psp.
  </documentation>

  <import namespace="http://www.getquote.com/StockQuoteService-interface"
     location="http://www.getquote.com/wsdl/SQS-interface.wsdl"/>

  <service name="StockQuoteService">
    <documentation>Stock Quote Service</documentation>

    <port name="SingleSymbolServicePort"
        binding="interface:SingleSymbolBinding">
      <documentation>Single Symbol Stock Quote Service</documentation>
      <soap:address location="http://www.getquote.com/stockquoteservice"/>
    </port>
  </service>
</definitions>
```

Service - The name attribute for the service element is used as the name for the businessService. The documentation element within the service element is used for the description of the businessService.
Import - The port name is appended to the overviewURL, which contains the reference to the service implementation document. The location of the service is used to set the accessPoint in the bindingTemplate.

Listing 4 contains the UDDI businessService definition that is created from the service implementation document. The categoryBag should contain one or more keyedReferences, which are used to categorize the intended business use for the service. The values are shown in the diagram key.

Listing 4

```
  <businessService businessKey="..." serviceKey="...">
  <name>StockQuoteService</name>

  <description xml:lang="en">
    Stock Quote Service
  </description>

  <bindingTemplates>
    <bindingTemplate bindingKey="..." serviceKey="...">
      <description>
        Single Symbol Stock Quote Service
      </description>
      <accesssPoint URLType="http">
        http://www.getquote.com/singlestockquote
      </accessPoint>
      <tModelInstanceDetails>
        <tModelInstanceInfo tModelKey="[tModel Key for Service Interface]">
          <instanceDetails>
            <overviewURL>
              http://www.getquote.com/services/SQS.wsdl
            </overviewURL>
          </instanceDetails>
```

```
        </tModelInstanceInfo>
      </tModelInstanceDetails>
    </bindingTemplate>
  </bindingTemplates>

  <categoryBag>
    <keyedReference tModelKey="UUID:DB77450D-9FA8-45D4-A7BC-04411D14E384"
                    keyName="Stock market trading services"
                    keyValue="84121801"/>
  </categoryBag>
</businessService>
```

BusinessService - The businessService name is set from the service name in the WSDL service implementation document. The description is set from the documentation element within the service element.
BindingTemplate - The description is set from the documentation element within the port element. The accessPoint is set from the soap:address extension element. The tModelKey is set to the UUID for the tModel that is associated with the service interface document. This tModel can be located using the namespace attribute on the import element. The overviewURL is the location of the service implementation document. It does not contain a reference to the SingleServiceQuote port, since it is the only one in this document.

Finding WSDL service interface descriptions

All WSDL service interfaces are published in a UDDI registry as a tModel. Each of these tModels are categorized to identify them as a WSDL service description. The UDDI find_tModel message can be used to find tModels that have been categorized. Using the UDDI V1.0 API, the find_tModel message listed in Listing 5 can be used to locate all WSDL service interface descriptions.

Listing 5

```
<?xml version="1.0"?>
<find_tModel generic="1.0" xmlns="urn:uddi-org:api">
  <categoryBag>
    <keyedReference tModelKey="UUID:C1ACF26D-9672-4404-9D70-39B756E62AB4"
                    keyName="uddi-org:types" keyValue="wsdlSpec"/>
  </categoryBag>
</find_tModel>
```

The find_tModel message will return a list of tModel keys. A specific service interface description is retrieved using the `get_tModelDetail` message. The get_tModelDetail message will return a tModel such as the one listed in Listing 2.

After a tModel has been retrieved, the overviewURL can be used to retrieve the contents of the WSDL service interface document.

Additional keyedReferences can be added to the categoryBag to limit the set of tModels that are returned in the response to this find request. The find_tModel message in Listing 6 can be used to locate all of the stock quote services that are defined using WSDL.

Listing 6

```
<?xml version="1.0"?>
<find_tModel generic="1.0" xmlns="urn:uddi-org:api">
  <categoryBag>
    <keyedReference tModelKey="UUID:C1ACF26D-9672-4404-9D70-39B756E62AB4"
                    keyName="uddi-org:types" keyValue="wsdlSpec"/>
    <keyedReference tModelKey="UUID:DB77450D-9FA8-45D4-A7BC-04411D14E384"
                    keyName="Stock market trading services"
                    keyValue="84121801"/>
  </categoryBag>
</find_tModel>
```

Finding WSDL service implementation descriptions

A WSDL service implementation is published in a UDDI registry as a businessService. The businessService will contain one or more bindingTemplates. The businessService is classified to identify it as a WSDL based service description. Using the UDDI V1.0 API, a businessEntity or set of businessEntities must be found before the find API for a businessService can be used. Similarly, a businessService or set of businessServices must be found before using the find API to locate a bindingTemplate.

Finding a UDDI businessService

There are two basic methods that can be used to find a service description. The UDDI `find_service` message can be used to find service descriptions with a specific classification, or it can be used to find service descriptions that implement a specific service interface.

Using the UDDI V1.0 API, the message in Listing 7 can be issued for a specific businessEntity to locate the businessServices that are implementations of a stock quote service. Additional keyedReferences can be added to the categoryBag to narrow the scope of the service descriptions that are returned in the response to this message.

Listing 7

```
<?xml version="1.0"?>
<find_service businessKey="..." generic="1.0" xmlns="urn:uddi-org:api">
  <categoryBag>
    <keyedReference tModelKey="UUID:DB77450D-9FA8-45D4-A7BC-04411D14E384"
                    keyName="Stock market trading services"
                    keyValue="84121801"/>
  </categoryBag>
</find_service>
```

The find_service message can also be used to locate businessServices that are implementations of a specific service interface. The message in Listing 8 contains an example of a find_service message that will find all businessServices, within a businessEntity, that implement the service interface for a stock quote service. Since a service interface is represented by a tModel, a tModelBag is used to specify the tModel key for the WSDL service interface associate with the stock quote service.

Listing 8

```
<?xml version="1.0"?>
<find_service businessKey="..." generic="1.0" xmlns="urn:uddi-org:api">
  <tModelBag>
    <tModelKey>[tModel key for WSDL service interface]</tModelKey>
  </tModelBag>
</find_service>
```

The find_service message will return a list of service keys. The businessService description can be retrieved using the `get_serviceDetail` message. The get_serviceDetail message will return a businessService such as the one in Listing 4.

After a businessService has been retrieved, a specific bindingTemplate can be selected to invoke the Web service. The accessPoint within the bindingTemplate is the endpoint for the service. The overviewURL can be used to retrieve the contents of the WSDL service implementation document, which may contain additional details about the implemented service.

Finding a UDDI bindingTemplate

If the businessService contains more than one bindingTemplate it might be difficult to determine which bindingTemplate to use. The `find_binding` message can be used to locate the bindingTemplate that should be used.

Listing 9 contains a find_binding message that can be used to retrieve the bindingTemplates that reference a specific tModel. This tModel can be associated with a specific binding within a WSDL service interface description.

Listing 9

```
<?xml version="1.0"?>
<find_binding serviceKey="..." generic="1.0" xmlns="urn:uddi-org:api">
  <tModelBag>
    <tModelKey>[tModelKey for WSDL service interface]<tModelKey/>
  </tModelBag>
</find_service>
```

This message will return one or more bindingTemplates, such as the one in Listing 10. After accessing the bindingTemplate, the endpoint for the Web service is listed in the accessPoint. If the bindingTemplate was created from an existing WSDL service implementation document, then the overviewURL may contain a reference to this document. This document can be accessed to obtain additional, human readable information about the Web service that can not be found in the UDDI registry.

Listing 10

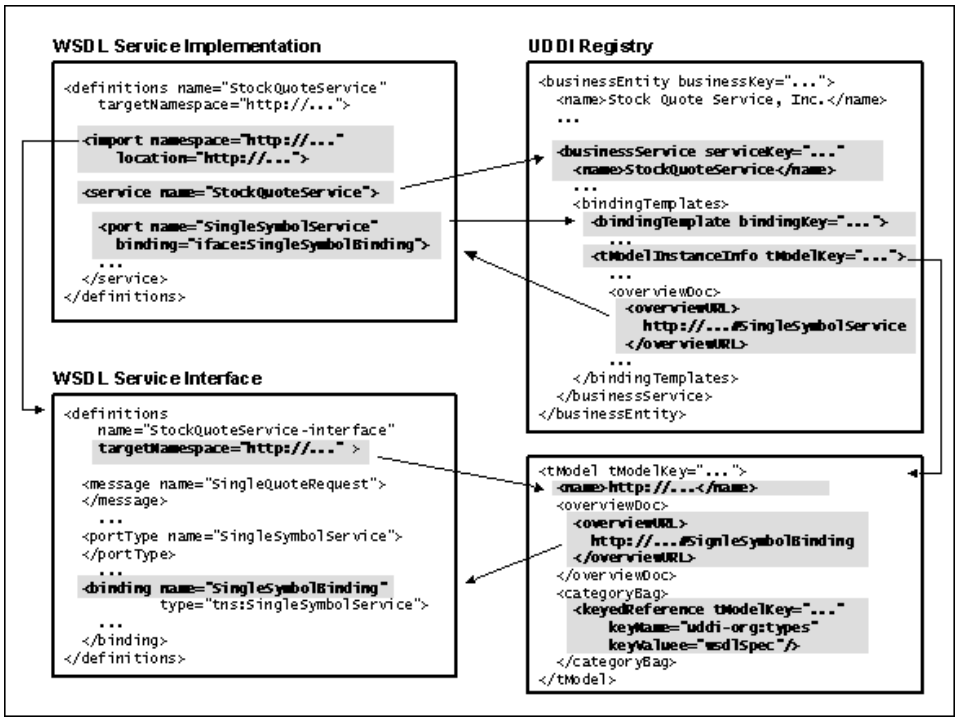
```
<?xml version="1.0"?>
<bindingTemplate bindingKey="" serviceKey="">
  <accesssPoint URLType="http">
    http://www.getquote.com/singlestockquote
  </accessPoint>
  <tModelInstanceDetails>
    <tModelInstanceInfo tModelKey="[tModel Key for Service Interface]">
      <instanceDetails>
        <overviewURL>
          http://www.getquote.com/services/SQS.wsdl
        </overviewURL>
      </instanceDetails>
    </tModelInstanceInfo>
  </tModelInstanceDetails>
</bindingTemplate>
```

Summary

This article has described the process for publishing and finding complete WSDL service descriptions in a UDDI registry. A WSDL service interface description is published as UDDI tModel, and a WSDL service implementation description is published as a UDDI businessService. Figure 4 provides a summary of the mapping from WSDL service interface and service implementation documents to UDDI registry entries.

In the next part of this series, we will take a look at the exact code needed for using WSDL with UDDI registries in five different scenarios that developers may commonly encounter.

Figure 4: Summary of WSDL to UDDI mapping



In the next part in this series, we will apply the process descibed in this article to different WSDL usage scenarios. Each usage scenario will illustrace how a different type of WSDL service description should be published in a UDDI registry.

Resources

- Web Services Description Language (WSDL) 1.1 , March 2001, World Wide Web Consortium.
- UDDI Programmer's API Specification Version 1.0, June 28, 2002, UDDI.org
- UDDI Data Structure Reference Version 1.0, June 28, 2002, UDDI.org
- UDDI Programmer's API Specification Version 2.0, July 19, 2002, UDDI.org
- UDDI Data Structure Reference Version 2.03, July 19, 2002, UDDI.org
- Using WSDL in a UDDI Registry 1.05, June 25, 2001, UDDI.org

About the authors

Peter Brittenham

Peter Brittenham is currently the lead architect for the IBM Web Services Toolkit. The Web Services Toolkit contains the tools and runtime support that are required to build Web services using SOAP and WSDL, as well as the runtime support to publish and find service definitions in a UDDI registry. He can be contacted at peterbr@us.ibm.com.

Francisco Cubera

Francisco Curbera is a Research Staff Member at IBM TJ Watson Research Center, which he joined in 1993. His work is centered around component software development and the design and implementation of standards-based distributed computing application frameworks. He is a coauthor of the Web Services Description Language (WSDL) and has contributed to the definition and implementation of several other specifications in the Web services area. He received his MS in Physics from the Universidad Complutense de Madrid (Spain), and his PhD in Computer Science from Columbia University. He can be contacted at curbera@us.ibm.com.

Dave Ehnebuske

David Ehnebuske is a Distinguished engineer with IBM's Application Integration and Middleware Division in Austin, TX. He has held a variety of technical and management positions at IBM development laboratories across the US. His long-term interest is in the practical exploitation of leading-edge software development technology for large-scale applications. For the last two years he has focused on the architecture and design of middleware aimed at enabling businesses to realize the promise of eBusiness.
David is Vice President, Americas of the IBM Academy of Technology. He can be reached at davide@us.ibm.com

---

Steve Graham

Steve Graham is an architect in the Emerging Technologies division of IBM Software Group. He has spent the last several years working on service-oriented architectures, most recently as part of the IBM Web Services Initiative. Prior to this, Steve worked

as a technologist and consultant working with various emerging technologies such as Java and XML, and before that he was an architect and consultant with the IBM Smalltalk consulting organization.
Before joining IBM, Steve was a developer with Sybase, a consultant, and a faculty member in the Department of Computer Science at the University of Waterloo. Steve holds a BMath and MMAth in computer science from The University of Waterloo.You can reach him at sggraham@us.ibm.com.