
[Close Window](#)[Print Story](#)

Registering a Web Service in UDDI

UDDI (Universal Description, Discovery and Integration) is a registry for Web services. It provides a mechanism to advertise and discover Web services. Although you don't need to use UDDI to implement a Web services solution, you'll find that a UDDI registry greatly simplifies the management and administration of your services, particularly once they have reached a certain critical mass.

Once you've developed more than a few services, and once you start giving access to those services to more than a few controlled individuals, management starts to get more challenging. Potential service consumers need a way to search for available services, to determine which services might be applicable to their projects, and to find metadata about those services. In particular, a service consumer must find the WSDL (Web Services Description Language) files that define a service. UDDI was designed to support these requirements.

A business may set up multiple UDDI registries in-house to support intranet and extranet operations, and a business may use UDDI registries set up by its customers and business partners. The UDDI Business Registry (UBR) is a free, public registry operated by IBM, Microsoft, SAP, and NTT, although few businesses feel comfortable advertising their business services in such an insecure public forum.

One key difference between a UDDI registry and other registries and directories is that UDDI provides a mechanism to categorize businesses and services using taxonomies. For example, service providers can use a taxonomy to indicate that a service implements a specific domain standard, or that it provides services to a specific geographic area. These taxonomies make it easier for consumers to find services that match their specific requirements.

Now, of course, in order for consumers to find a service, the service provider must first properly register the service. There are many different ways to register a service, so it's important to define registration conventions so that consumers know what to search for.

In July 2003, the standards group that manages the UDDI specification (the OASIS UDDI-Spec Technical Committee) published a Technical Note called "Using WSDL in a UDDI Registry, Version 2.0." This Technical Note defines conventions for registering a Web service based on information in the service's WSDL definition. (This Technical Note supercedes the previous UDDI Best Practices document, "Using WSDL in a UDDI Registry, Version 1.08.")

The UDDI Data Model

A UDDI registry manages information about service types and service *implementations*. Going back to basic object-oriented concepts, a type is the definition of a thing, and an *implementation* is an

instance of a thing. A service-type registration, called a technical model (tModel), represents a technical specification or some other metadata. For example, a tModel could represent a WSDL document that defines a Web service.

A service implementation registration represents a service offered by a specific service provider. It specifies information about the business entity that offers the service (businessEntity), describes the service (businessService), and captures the binding information (bindingTemplate) required to use the service. The binding Template captures the service endpoint address, and associates the service with the tModels that represent its technical specifications. Figure 1 shows the UDDI data model that captures these registrations.



FIGURE 1 | The UDDI data model

UDDI also uses tModels to represent taxonomies. Service providers categorize UDDI registrations using a construct called a keyedReference. A keyedReference allows you to assign property values to a UDDI entity using a name/value pair. The "name" is the tModelKey of the taxonomy tModel. The "value" is the descriptive information supplied for this categorization. You may define a limited set of valid values (a Value Set) for a taxonomy. For example, you could define a taxonomy to represent the deployment status of a service, and define a Value Set with values of "development", "test", and "production". To indicate that a service is in production, you would add a keyedReference to the businessService entity that represents the service. The keyedReference would specify the tModelKey for the deployment status tModel taxonomy, and a value of "production". It would look something like this:

```
<keyedReference
```

```

tModelKey="uuid:2e444afb-33e5-
3a7b-81b7-1cb8a373f457"
keyName="deployment status"
keyValue="production"/>

```

You'll notice that the `keyedReference` includes three attributes: `tModelKey`, `keyName`, and `keyValue`. The `tModelKey` and `keyValue` attributes are required. The `keyName` attribute is optional and insignificant. The `keyName` attribute allows you to assign a human-readable name to the `keyedReference` attribute, but you cannot search the registry using this name.

The WSDL Data Model

A WSDL file describes a Web service. A WSDL file describes *what* functionality a Web service offers, *how* it communicates, and *where* to find it. Figure 2 shows the WSDL data model, and shows the relationship of the various definition elements.

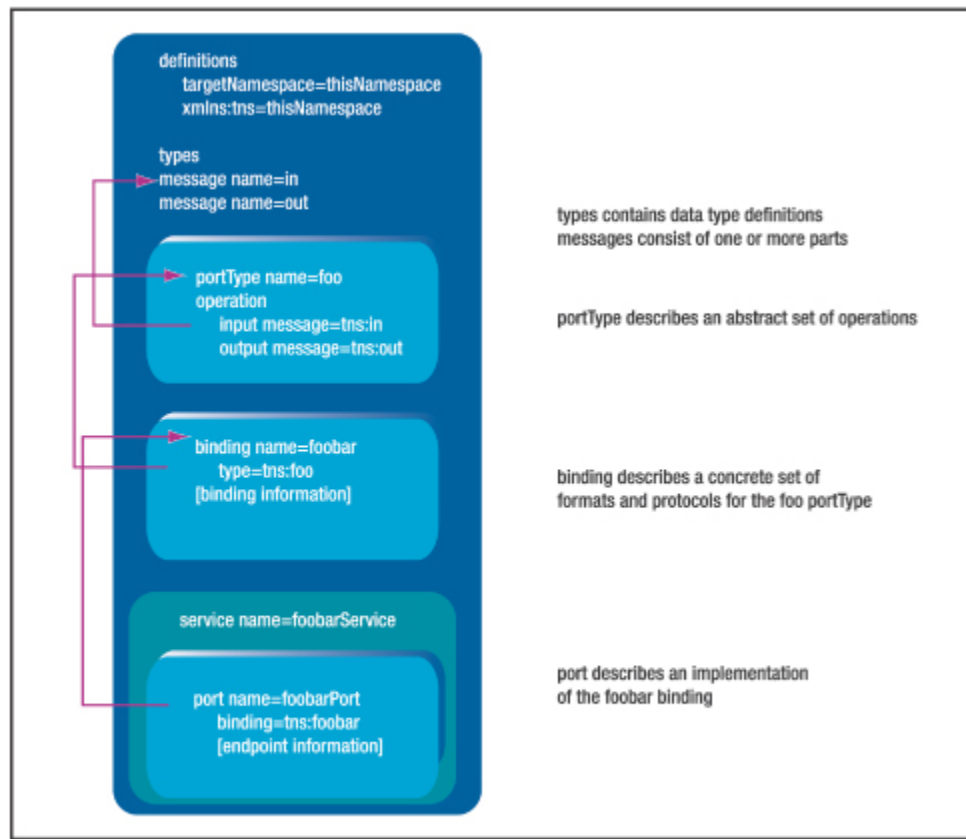


FIGURE 2 | The WSDL data model

The *what* part of a WSDL document (a portType) defines the abstract interface of a Web service. A portType is a reusable definition component. Multiple providers can offer Web services that implement the same portType. A portType specifies what operations the abstract service supports, and it specifies the input and output messages associated with each operation. Each message is defined in a separate *message* definition, which in turn references element or type definitions in the *types* section.

The *how* part of a WSDL document (a *binding*) maps a portType to a concrete set of formats and protocols. A binding indicates how the input and output messages defined in the referenced portType should be packaged into a message, and it specifies what communication protocols can be used to convey the messages. As with portTypes, a WSDL binding is a reusable definition component. Multiple service providers can support the same WSDL binding. A service can also implement multiple bindings of the same portType, thereby supporting multiple protocols.

The *where* part of a WSDL document (a *service*) describes a specific Web service implementation. A Web service implementation contains one or more *ports*. A port implements the referenced binding and specifies the endpoint of that Web service binding. A business might offer multiple endpoints to a particular service, each implementing a different binding.

Mapping WSDL to UDDI

The "Using WSDL in a UDDI Registry" Technical Note defines a mapping model between WSDL and UDDI that supports the reusability characteristics of the WSDL data model.

Figure 3 shows a Web service description that consists of three different WSDL files for the portType, binding, and service definitions. The service definition imports the binding definition, and the binding definition imports the portType definition. By breaking the descriptions into multiple files, the portType and binding definitions support reusability.

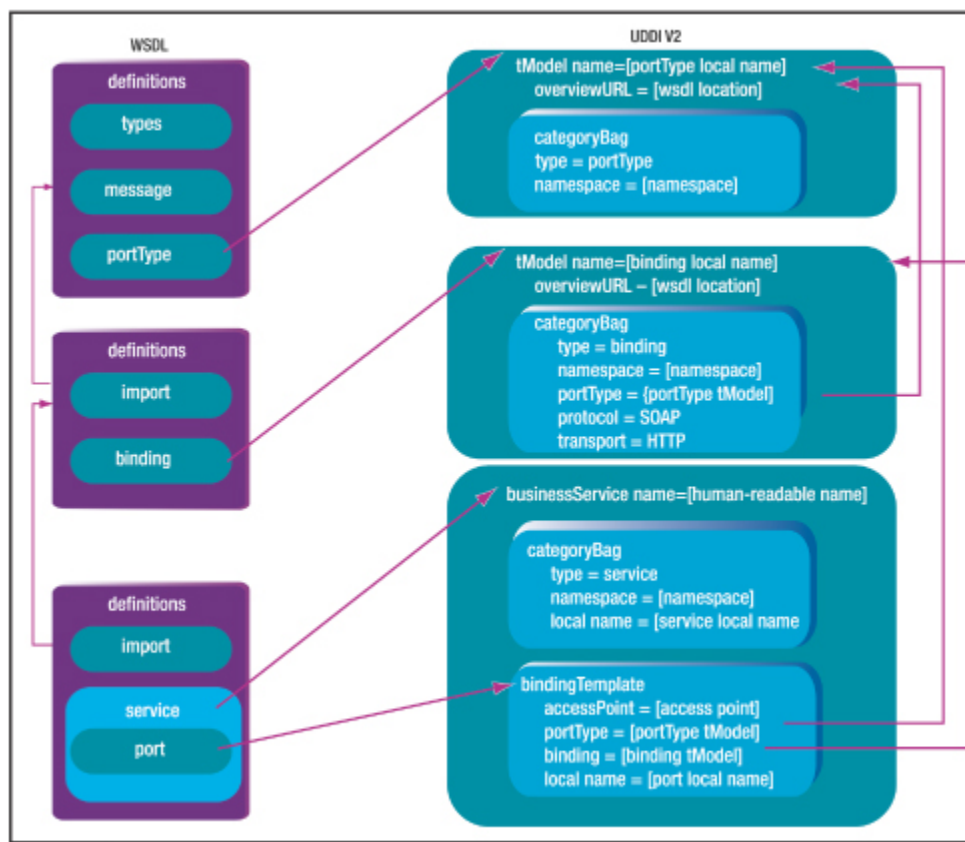


FIGURE 3 Mapping WSDL to UDDI

This is the approach people should take when defining corporate standards or industry-specific domain standards. A standards group should define the abstract service portTypes and bindings, but it should not define specific service implementations. A service provider that implements a standard service description should not redefine the standard definitions. It should import the standard definitions.

The WSDL-to-UDDI mapping model is designed to help users find services that implement standard definitions. It also helps users find services that support a specific set of protocols, such as SOAP over HTTP. The Technical Note defines a new set of canonical tModels to help capture the necessary information. These tModels include categorization systems to capture the WSDL element type (portType, binding, or service) and the element's fully qualified name (its local name and namespace name), along with the protocols supported by a WSDL binding.

The mapping model works as follows:

- **Register each WSDL portType as a tModel.** Give the tModel the same name as the portType local name. Capture the URL of the WSDL file in the tModel overviewURL. Use keyedReferences to indicate that the tModel represents a WSDL portType and to capture the namespace.
- **Register each WSDL binding as a tModel.** Give the tModel the same name as the binding local name. Capture the URL of the WSDL file in the tModel overviewURL. Use keyedReferences to indicate that the tModel represents a WSDL binding, to capture the namespace, to reference the tModel that describes the portType for this binding, and to indicate protocols supported by this binding.
- **Register each WSDL service as a businessService.** Give the businessService a human-readable name. Use keyedReferences to indicate that this service is described by WSDL and to capture the WSDL service local and namespace names.
- **Register each WSDL service port as a bindingTemplate.** Capture the port endpoint address in the bindingTemplate accessPoint. Using the tModelInstanceDetails structure, associate the service with all the tModels that describe the service. At a minimum, the bindingTemplate should point to the tModels that represent the WSDL portType and WSDL binding definitions. Capture the port's local name in the InstanceParms field of the WSDL binding link.

Conclusion

The new mapping gives users some powerful options when searching for services. It permits users to perform the following types of queries:

- Find specifications by namespace name
- Find specifications by WSDL portType or binding name
- Find all bindings for a portType
- Find all SOAP bindings for a portType
- Find implementations by namespace name
- Find all implementations of a WSDL portType
- Find all SOAP implementations of a WSDL portType
- Find all implementations of a WSDL binding

As your investment in Web services grows, you'll no doubt find these capabilities valuable.

© 2008 SYS-CON Media Inc.