UDDI Specifications TC

Technical Note

Using WSDL in a UDDI Registry, Version 2.0

Document identifier:
uddi-spec-tc-tn-wsdl-20030319-wd

Location:
http://www.oasis-open.org/committees/uddi-spec/doc/draft/uddi-spec-tc-tn-wsdl-20030319-wd.htm

Authors (alphabetically):
John Colgrave, IBM colgrave@uk.ibm.com
Karsten Januszewski, Microsoft karstenj@microsoft.com

Editors:
Anne Thomas Manes, anne@manes.net
Tony Rogers, Computer Associates tony.rogers@ca.com

Abstract:
This document is an OASIS UDDI Technical Note that defines a new approach to using WSDL in a UDDI Registry.

Status:
This document is a working draft.

Committee members should send comments on this document to the *uddi-spec@lists.oasis-open.org* list. Others should subscribe to and send comments to the *uddi-spec-comment@lists.oasis-open.org* list. To subscribe, send an email message to *uddi-spec-comment-request@lists.oasis-open.org* with the word "subscribe" as the body of the message.

Copyright

Table of Contents

1 Introduction

The Universal Description, Discovery & Integration (UDDI) specification provides a platform-independent way of describing and discovering Web services and Web service providers. The UDDI data structures provide a framework for the description of basic service information, and an extensible mechanism to specify detailed service access information using any standard description language. Many such languages exist in specific industry domains and at different levels of the protocol stack. The Web Services Description Language (WSDL) is a general purpose XML language for describing the interface, protocol bindings, and the deployment details of network services. WSDL complements the UDDI standard by providing a uniform way of describing the abstract interface and protocol bindings of arbitrary network services. The purpose of this document is to clarify the relationship between the two and to describe a recommended approach to mapping WSDL descriptions to the UDDI data structures.

Consistent and thorough WSDL mappings are critical to the utility of UDDI.

1.1 Goals and Requirements

The primary goals of this mapping are:

1. To enable the automatic registration of WSDL definitions in UDDI
2. To enable precise and flexible UDDI queries based on specific WSDL artifacts and metadata
3. To maintain compatibility with the mapping described in the *Using WSDL in a UDDI Registry, Version 1.08* [1] Best Practice document
4. To provide a consistent mapping for UDDI Version 2 and UDDI Version 3
5. To support any logical and physical structure of WSDL description

This mapping prescribes a consistent methodology to map WSDL 1.1 artifacts to UDDI structures. It describes an approach that represents reusable, abstract Web service artifacts, (WSDL portTypes and WSDL bindings) and Web service implementations (WSDL services and ports). Tools can use this mapping to generate UDDI registrations automatically from WSDL descriptions.

This mapping captures sufficient information from the WSDL documents to allow precise queries for Web services information without further recourse to the source WSDL documents, and to allow the appropriate WSDL documents to be retrieved once a match has been found. Given that the source WSDL documents can be distributed among the publishers using a UDDI registry, a UDDI registry provides a convenient central point where such queries can be executed.

This mapping enables the following types of queries for both design-time and run-time discovery:

- Given the namespace and/or local name of a wsdl:portType, find the tModel that represents that portType.
- Given the namespace and/or local name of a wsdl:binding, find the tModel that represents that binding.
- Given a tModel representing a portType, find all tModels representing bindings for that portType.
- Given a tModel representing a portType, find all bindingTemplates that represent implementations of that portType.
- Given a tModel representing a binding, find all bindingTemplates that represent implementations of that binding.

Some aspects of the mapping allow information to be retrieved directly without further queries being necessary. For example, given the tModel representing a binding, it is possible to retrieve the key of the tModel representing the portType referred to by the binding. Other aspects of the mapping may require multiple queries to be issued to the UDDI registry.

Although the UDDI V3 data model is slightly different from the UDDI data model, this mapping ensures that the same information is captured in both versions.

1.2 Relationship to Version 1 Best Practice

This document builds on *Using WSDL in a UDDI Registry, Version 1.08*, providing an expanded modeling practice that encompasses the flexibility of WSDL. The primary difference between this mapping and the one described in the existing Best Practice is that this mapping provides a methodology to represent individual Web services artifacts.

As a Technical Note, this document does not replace the Version 1 Best Practice. If the additional flexibility is not required, the existing Best Practice can still be used, particularly when the UDDI artifacts are published manually.

It is anticipated that implementations of the approach described in this Technical Note will be developed, and that once experience with those implementations is obtained this Technical Note will become a Best Practice.

A final goal is to be compatible with the existing Best Practice in that a tModel representing a WSDL binding published using the approach described in this document should be usable by a client that uses the Version 1 Best Practice approach.

1.3 Terminology

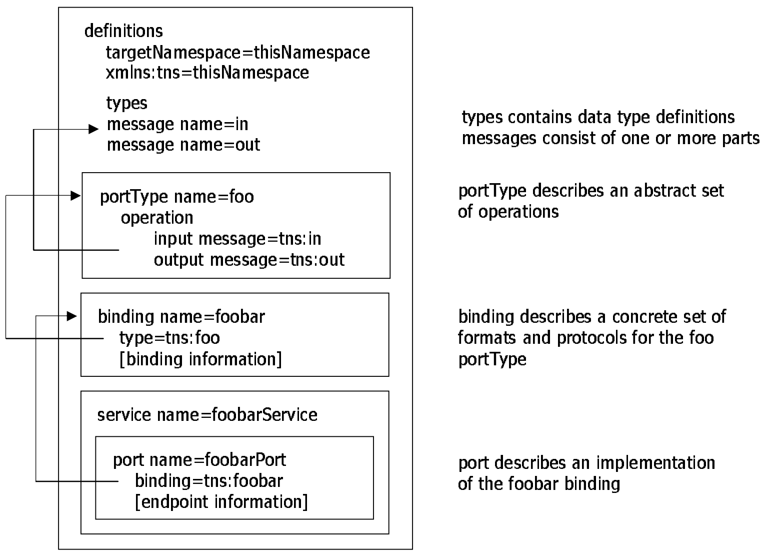
The key words *must, must not, required, shall, shall not, should, should not, recommended, may, and optional* in this document are to be interpreted as described in **[RFC2119]**.

2 Mapping Two Data Models: WSDL & UDDI

A brief discussion of the two respective data models, WSDL and UDDI, follows. For a complete explanation of these specifications, see [2], [3], and [4].

2.1 WSDL Data Model

A review of WSDL in the context of the goals and requirements will help guide a new mapping practice in UDDI.


definitions
targetNamespace=thisNamespace
xmlns:tns=thisNamespace
types
message name=in
message name=out
portType name=foo
operation
input message=tns:in
output message=tns:out
binding name=foobar
type=tns:foo
[binding information]
service name=foobarService
port name=foobarPort
binding=tns:foobar
[endpoint information]
types contains data type definitions
messages consist of one or more parts
portType describes an abstract set of operations
binding describes a concrete set of formats and protocols for the foo portType
port describes an implementation of the foobar binding

2.1.1 portType

The central construct in WSDL is the portType. A portType is an abstract collection of operations that may be supported by one or more Web services. A WSDL portType defines these operations in terms of message definitions, which usually rely on the XML Schema language to describe the representation of each message. A single WSDL file may contain multiple portType entities. Each portType is uniquely identified by the combination of its local name and the target namespace of the definitions element that contains the portType.

WSDL portTypes may be implemented by more than one Web service. Web services that purport to support a given portType must adhere not only to the message formats that are part of the WSDL definition; they must also adhere to the semantic agreement that is implicitly part of the portType. This consistency allows applications to treat two Web services as substitutable if and only if they implement a common portType.

2.1.2 binding

WSDL portTypes are abstract Web service descriptions and do not specify information about the encoding and transport protocols used to transmit the messages. To specify encoding and transport protocol details in WSDL, one must define a second construct, known as a binding. A WSDL binding specifies a specific set of encoding and transport protocols that may be used to communicate with an implementation of a particular WSDL portType. A WSDL binding specifies its portType through a QName reference. The referenced portType may or may not be in the same target namespace as the binding itself. Again, a single WSDL file may contain multiple bindings. For example, a WSDL file may describe multiple protocol bindings for a single portType. Like a portType, a binding is uniquely identified by the combination of its local name and the target namespace of the definitions element that contains the binding.

As with portTypes, WSDL bindings are abstract definitions and do not represent a Web service implementation. Multiple Web services may implement the same WSDL binding.

2.1.3 service and port

Finally, WSDL defines a Web service implementation as a service with a collection of named ports. Each port implements a particular portType using the protocols defined by a named binding. A service may expose multiple ports in order to make a single portType available over multiple protocols. A service may also expose multiple ports in order to expose more than one portType from a single logical entity. A WSDL port specifies the binding it implements through a QName reference. The referenced binding may or may not be in the same target namespace as the port itself. A single WSDL file may contain multiple services. A service is uniquely identified by the combination of its local name and the target namespace of the definitions element that contains the service. Likewise, a port is uniquely identified by the combination of its local name and the target namespace of the definitions element that contains the port.

2.1.4 import

The import directive in WSDL allows the separation of these different entities into multiple files. As such, a WSDL file may be composed of a single portType, multiple portTypes, a single binding that imports its portType definition, multiple bindings, a single service, or multiple services, etc. The WSDL data model provides great flexibility in terms of composition and reusability of WSDL entities.

Given this flexibility, the critical components of a WSDL file in terms of composition and identity are the target namespace of the definitions element and the local names that identify each portType, binding, service, and port within the target namespace.

2.2 UDDI Data Model

As an aid to understanding the sections ahead, we provide here a brief overview of two UDDI data structures that are particularly relevant to the use of WSDL in the context of a UDDI registry: the tModel and the businessService.

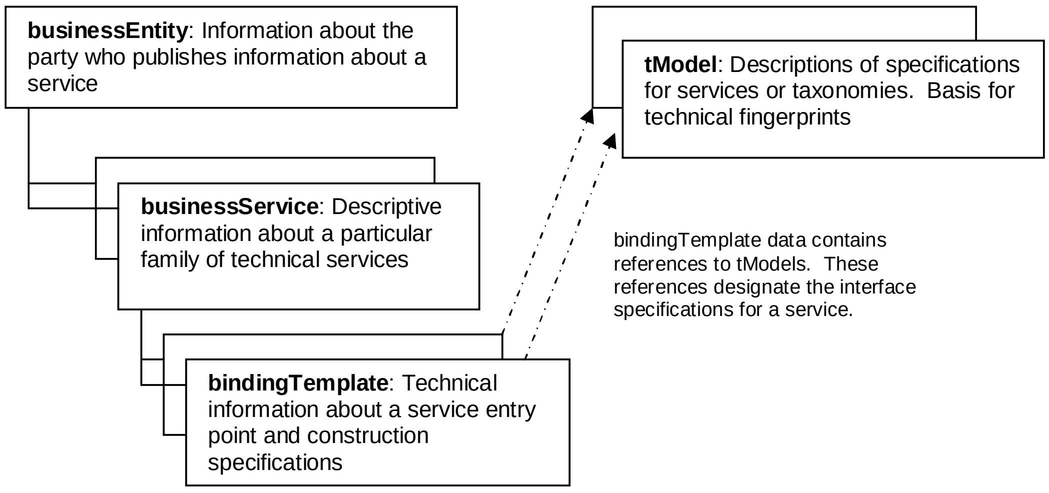
2.2.1 tModels

TModels are often referred to as service type definitions. TModels represent unique concepts or constructs. They are used to describe compliance with a specification, a concept, or a shared design. TModels have various uses in the UDDI registry. In the case of mapping WSDL-described Web services, tModels have two uses. First, tModels are used to represent technical specifications such as service types, bindings, and wire protocols. Second, tModels are used to implement taxonomies that are used to identify or categorize technical specifications and services. This Technical Note defines a set of specification and taxonomy tModels that are used when mapping WSDL entities to UDDI entities. These tModels are defined in Appendix B.

When a particular specification is registered in the UDDI registry as a tModel, it is assigned a unique key, called a tModelKey. This key is used by other UDDI entities to reference the tModel, for example to indicate compliance with the specification.

Each specification tModel contains an overviewURL, which provides the address of the specification itself, for example, a WSDL file.

Additional metadata can be associated with a specification tModel using any number of identifier and categorization taxonomies. Identifiers are grouped in a construct called an identifierBag, and categories are grouped in a construct called a categoryBag. These bags contain a set of keyedReference elements. Each keyedReference specifies the tModelKey of the taxonomy tModel and a name/value pair that specifies the metadata. For example, a keyedReference referencing the namespace taxonomy can be used to specify a WSDL namespace. The metadata values specified in keyedReference elements can be used as selection criteria when searching UDDI.


businessEntity: Information about the party who publishes information about a service
tModel: Descriptions of specifications for services or taxonomies. Basis for technical fingerprints
businessService: Descriptive information about a particular family of technical services
bindingTemplate data contains references to tModels. These references designate the interface specifications for a service.
bindingTemplate: Technical information about a service entry point and construction specifications

2.2.2 businessService & bindingTemplate

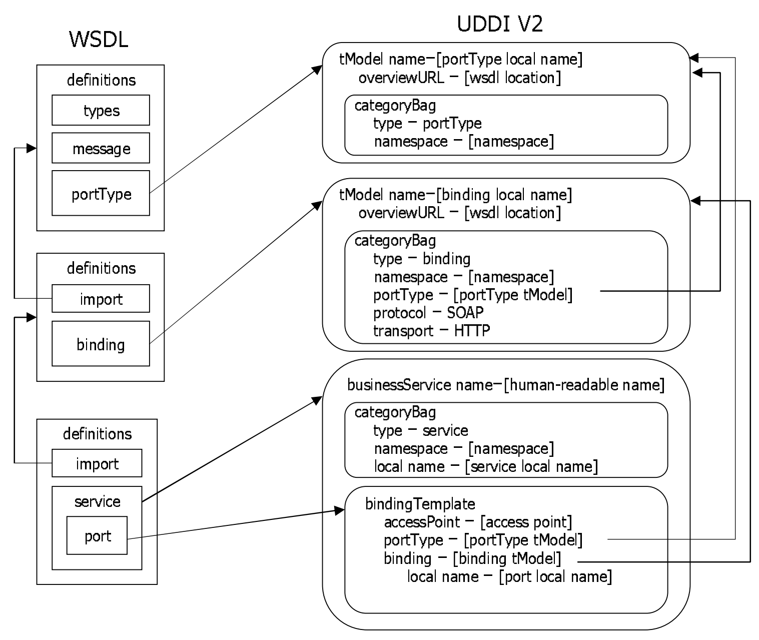
Services are represented in UDDI by the businessService data structure, and the details of how and where the service is accessed are provided by one or more bindingTemplate structures. The businessService might be thought of as a logical container of services. The bindingTemplate structure contains the accessPoint of the service, as well as references to the tModels it is said to implement.

2.3 Mapping WSDL and UDDI

WSDL is designed to support modular and reusable definitions, and each definition artifact has certain relationships with other definition artifacts. As described in Section 1.1, the goals of this technical note and the mapping it defines are to enable the automatic registration of WSDL definitions in UDDI, to enable precise and flexible UDDI queries based on specific WSDL artifacts and metadata, to maintain compatibility with the Version 1 Best Practice methodology, and to ease migration from UDDI V2 to UDDI V3. The mapping itself addresses the first goal. The second goal provides the rationale for the methodology used in this mapping. In order to support queries based on specific WSDL artifacts and metadata, this mapping must be able to represent the individual WSDL artifacts and the relationships between artifacts. This goal also provides the rationale for the amount of information that must be captured in UDDI. Additional information must also be included in some cases to support the third goal. To address the fourth goal, the information captured in the two mappings is as consistent as possible.

2.3.1 Mapping Overview

This mapping describes a methodology for mapping WSDL 1.1 definitions to the UDDI V2 and UDDI V3 data models. The methodology maps each WSDL artifact to a separate UDDI entity, accurately representing the “building block” design of WSDL descriptions. wsdl:portType and wsdl:binding elements map to uddi:tModel entities, wsdl:service elements map to uddi:businessService entities and wsdl:port elements map to uddi:bindingTemplate entities. KeyedReferences provide a mechanism to express additional metadata and to represent a relationship between two UDDI entities.


WSDL
definitions
types
message
portType
definitions
import
binding
definitions
import
service
port
UDDI V2
tModel name=[portType local name]
overviewURL = [wsdl location]
categoryBag
type = portType
namespace = [namespace]
tModel name=[binding local name]
overviewURL = [wsdl location]
categoryBag
type = binding
namespace = [namespace]
portType = [portType tModel]
protocol = SOAP
transport = HTTP
businessService name=[human-readable name]
categoryBag
type = service
namespace = [namespace]
local name = [service local name]
bindingTemplate
accessPoint = [access point]
portType = [portType tModel]
binding = [binding tModel]
local name = [port local name]

2.3.2 Comparison to Version 1 Mapping

One important thing to note about this mapping, especially as compared to the mapping described in the Version 1 Best Practice, is that this approach may map a single WSDL file to multiple tModels. For example, a single WSDL file that contains one portType definition and two binding definitions will map to three distinct tModels in UDDI. This approach differs from the Version 1 Best Practice, which would map the entire WSDL file to a single tModel. The rationale for this new mapping decision is to more effectively represent the modularity and reusability of WSDL artifacts in UDDI. A Web service implementation might implement only one of the bindings described in a WSDL file. By decomposing WSDL into multiple tModels, one can accurately model in UDDI exactly which portTypes and bindings a given Web service implementation supports, as opposed to being constrained to asserting that a Web service always supports the entirety of the WSDL file.

While there is an increased amount of data from a WSDL file modeled in UDDI, this new approach is in accord with the Version 1 Best Practice in that it does not attempt to use UDDI as a repository for *all* of the data in a WSDL file. Just as in the Version 1 Best Practice, one still must go outside of the UDDI registry to retrieve the portType and binding information necessary for software applications to work with that Web service.

2.3.3 New Canonical tModels

This mapping introduces a number of canonical tModels that are used to represent WSDL metadata and relationships. These tModels, including the WSDL Entity Type tModel, the XML Namespace tModel, the XML Local Name tModel, the WSDL portType Reference tModel, the WSDL URL Reference tModel, the SOAP Protocol tModel, the HTTP Protocol tModel, the Protocol Categorization tModel, the Transport Categorization tModel, the WSDL URL tModel, and the WSDL Address tModel, are described in Appendix B. These tModels MUST be registered in the UDDI registry to support this mapping. As both V1/V2 and V3 keys are given for these tModels, their keys should be treated as evolved keys.

2.3.4 General Conventions

In this mapping, each WSDL artifact is mapped to its corresponding UDDI entity. A set of keyedReference elements is added to each UDDI entity to capture additional metadata. In order to support the requirements outlined in Section 1.1, the following metadata is captured for each entity:

- The type of WSDL entity being defined (i.e., portType, binding, service, or port)
- The target namespace of the WSDL definitions file that defines the WSDL entity
- The local name of the WSDL entity being defined
- The location of the WSDL file that defines the WSDL entity is captured for portType, binding and, optionally, service entities.

Any relationships and dependencies between entities must also be captured. For example, a tModel that represents a binding provides a reference to the tModel that represents the portType implemented by the binding.

To maintain compatibility with the Version 1 Best Practice mapping, certain UDDI entities are also characterized as being of type “wsdlSpec”.

2.3.5 Support for Multiple UDDI API Versions

The mapping described is designed to appear the same whichever version of the UDDI API is used to access it. There are differences that are mandated by the differences in the API versions, and such differences are noted in the appropriate sections.

The V3 API also introduces some optional features that are not visible to the older APIs, and some guidance is given as to the usage of these optional features.

2.3.6 References to WSDL Components

A UDDI entity normally references technical specifications using the overviewURL element. As noted above, in this mapping a single WSDL file may map to multiple tModels, and each tModel refers to a particular WSDL entity within the file. The particular WSDL entity is uniquely identified by the combination of its local name and the target namespace of the definitions element that contains the WSDL entity. This identity information SHOULD be determined from the metadata contained within the entity’s categoryBag. Alternatively, the overviewURL value MAY contain a fragment identifier that identifies the particular WSDL entity. If the optional fragment identifier is used, then the value of the overviewURL MUST conform to the syntax described in Appendix C.

2.3.7 WSDL Extensibility Elements

WSDL uses extensibility elements to describe technology-specific information within a WSDL definition. Extensibility elements may be included under many of the WSDL elements. The only extensibility elements that are relevant to this mapping are binding and port extensions, specifically the extensibility elements that can be added to the wsdl:binding and wsdl:port elements. The first of these is used to declare particular protocols and message formats; the second is to provide address information.

Information from these extensibility elements is mapped to the tModel for a wsdl:binding and a bindingTemplate. The mappings defined in this document include details on the SOAP 1.1 and HTTP GET/POST bindings defined in the WSDL 1.1 W3C Note. The mappings also describe how other bindings should be incorporated into the UDDI mapping.

2.3.8 Support for WSDL Implementation Documents

In the context of this Technical Note, a WSDL Implementation Document is a WSDL document that contains at least one wsdl:service element and its associated wsdl:port elements. There are two options for how this implementation information is described in UDDI:

1. The information in the UDDI model is the authoritative information and there is no reference to a WSDL Implementation Document.
2. A reference to an external WSDL Implementation Document can be stored in UDDI and the remaining information in UDDI is used to describe the appropriate element in the external WSDL resource.

The mapping described in the body of this document corresponds to the first option above, and that is assumed to be the default mapping. The second option is described in Appendix A.

2.4 Mapping WSDL 1.1 in UDDI V2

This section describes a detailed mapping of WSDL 1.1 artifacts to the UDDI V2 data model.

2.4.1 wsdl:portType → uddi:tModel

A wsdl:portType MUST be modeled as a uddi:tModel.

The minimum information that must be captured about a portType is its entity type, its local name, its namespace, and the location of the WSDL document that defines the portType. Capturing the entity type enables users to search for tModels that represent portType artifacts. Capturing the local name, namespace, and WSDL location enables users to locate the definition of the specified portType artifact.

The wsdl:portType information is captured as follows:

The uddi:name element of the tModel MUST be the value of the name attribute of the wsdl:portType.

The tModel MUST contain a categoryBag, and the categoryBag MUST contain at least the following keyedReference elements:

1. A keyedReference with a tModelKey of the WSDL Entity Type taxonomy and a keyValue of “portType”.
2. A keyedReference with a tModelKey of the XML Namespace taxonomy and a keyValue of the target namespace of the wsdl:definitions element that contains the wsdl:portType.[1]

The tModel MUST contain an overviewDoc with an overviewURL containing the location of the WSDL file that describes the wsdl:portType.

2.4.1.1 Summary of Mapping of wsdl:portType

WSDL	UDDI
portType	tModel (categorized as portType)
Namespace of portType	keyedReference in categoryBag
Local name of portType	tModel name
Location of WSDL file	overviewURL

2.4.2 wsdl:binding → uddi:tModel

A wsdl:binding MUST be modeled as a uddi:tModel.

The minimum information that must be captured about a binding is its entity type, its local name, its namespace, the location of the WSDL document that defines the binding, the portType that it implements, its protocol, and, optionally, the transport information. Capturing the entity type enables users to search for tModels that represent binding artifacts. Capturing the local name, namespace, and WSDL location enables users to locate the definition of the specified binding artifact. The link to the portType enables users to search for bindings that implement a particular portType.

[1] WSDL 1.1 does not require the usage of a targetNamespace, but such a practice is not recommended. In the event that a WSDL file without a targetNamespace is registered in UDDI, it will not have an XML Namespace keyedReference, and queries for these tModels based solely on the tModel name could return multiple results because no namespace can be specified.

A wsdl:binding corresponds to a WSDL service interface definition as defined by the mapping in the Version 1 Best Practice. To maintain compatibility with the previous mapping, the binding must also be characterized as type "wsdlSpec".

The wsdl:binding information is captured as follows:

The uddi:name element of the tModel MUST be the value of the name attribute of the wsdl:binding.

The tModel MUST contain a categoryBag, and the categoryBag MUST contain at least the following keyedReference elements:

1. A keyedReference with a tModelKey of the WSDL Entity Type taxonomy and a keyValue of "binding".
2. A keyedReference with a tModelKey of the XML Namespace taxonomy and a keyValue of the target namespace of the wsdl:definitions element that contains the wsdl:binding.
3. A keyedReference with a tModelKey of the WSDL portType Reference taxonomy and a keyValue of the tModelKey that models the wsdl:portType to which the wsdl:binding relates.
4. A keyedReference with a tModelKey of the UDDI Types taxonomy and a keyValue of "wsdlSpec" for backward compatibility[2].
5. One or two keyedReferences as required to capture the protocol and optionally the transport information – refer to the next section.

The tModel MUST contain an overviewDoc with an overviewURL containing the location of the WSDL file that describes the wsdl:binding.

2.4.2.1 wsdl:binding Extensions

Information about the protocol and transport, if applicable, specified in an extension to the wsdl:binding is used to categorize the binding tModel as described in the following sections. This information is specified using two of the taxonomies defined in this Technical Note:

1. Protocol Categorization
2. Transport Categorization

The valid values for the Protocol Categorization taxonomy are tModelKeys of tModels that are categorized as protocol tModels. Similarly, the valid values for the Transport Categorization taxonomy are tModelKeys of tModels that are categorized as transport tModels.

The reason for having these two categorization schemes that take tModel keys as values is to allow other standard or proprietary protocols and transports to be defined and used in the same way as the standard SOAP and HTTP protocols and transport.

2.4.2.1.1 soap:binding

If the wsdl:binding contains a soap:binding extensibility element from the http://schemas.xmlsoap.org/wsdl/soap/ namespace then the categoryBag MUST include a keyedReference with a tModelKey of the Protocol Categorization taxonomy and a keyValue of the tModelKey of the SOAP Protocol tModel.

If the value of the transport attribute of the soap:binding element is http://schemas.xmlsoap.org/soap/http then the categoryBag MUST include a keyedReference with a tModelKey of the Transport Categorization taxonomy and a keyValue of the tModelKey of the HTTP Transport tModel.

If the value of the transport attribute is anything else, then the bindingTemplate MUST include an additional keyedReference with a tModelKey of the Transport Categorization taxonomy and a keyValue of the tModelKey of an appropriate transport tModel.

[2] By categorizing a wsdl:binding tModel according to the Version 1 UDDI/WSDL Best Practice, backward compatibility is maintained. However, wsdl:portType tModels should not be categorized with this designation, as the wsdl:portType tModel will not contain sufficient information to compose a complete WSDL binding.

2.4.2.1.2 http:binding

If the wsdl:binding contains an http:binding extensibility element from the http://schemas.xmlsoap.org/wsdl/http/ namespace then the categoryBag MUST include a keyedReference with a tModelKey of the Protocol Categorization taxonomy and a keyValue of the tModelKey of the HTTP Protocol tModel.

Note that this is a different tModel from the HTTP Transport tModel, and in this case there is no separate transport tModel, and therefore no keyedReference in the categoryBag from the Transport Categorization taxonomy.

2.4.2.1.3 Other wsdl:binding Extensions

Other wsdl:binding extensibility elements are handled in a similar fashion. It is assumed that vendors who provide other bindings will provide the appropriate protocol and transport tModels.

2.4.2.2 Summary of Mapping of wsdl:binding

WSDL	UDDI
binding	tModel (categorized as binding and wsdlSpec)
Namespace of binding	keyedReference in categoryBag
Local name of binding	tModel name
Location of WSDL file	overviewURL
portType binding relates to	keyedReference in categoryBag
Protocol from binding extension	keyedReference in categoryBag
Transport from binding extension (if there is one)	keyedReference in categoryBag

2.4.3 wsdl:service → uddi:businessService

A wsdl:service MUST be modeled as a uddi:businessService. An existing businessService MAY be used or a new businessService MAY be created[3]. Only one wsdl:service can be modeled by an individual uddi:businessService.

The minimum information that must be captured about a service is its entity type, its local name, its namespace, and the list of ports that it supports. Capturing the entity type enables users to search for services that are described by a WSDL definition. The list of ports provides access to the technical information required to consume the service.

The wsdl:service information is captured as follows:

If a new businessService is created, the uddi:name of this businessService SHOULD be a human readable name, although if no human readable name is specified, it MUST be the value of the name attribute of the wsdl:service[4].

[3] WSDL permits any arbitrary group of ports to be collected into a single service, therefore a wsdl:service may not directly correspond to a uddi:businessService. As a best practice for this mapping, a wsdl:service SHOULD contain a collection of associated ports that relate to a single logical business service, for example, a collection of ports that implement alternate bindings for a particular portType. A wsdl:service SHOULD NOT contain multiple ports that do not relate to a single logical business service.

The businessService MUST contain a categoryBag, and the categoryBag MUST contain at least the following keyedReference elements:

1. A keyedReference with a tModelKey of the WSDL Entity Type taxonomy and a keyValue of “service”.
2. A keyedReference with a tModelKey of the XML Namespace taxonomy and a keyValue of the target namespace of the wsdl:definitions element that contains the wsdl:service.
3. A keyedReference with a tModelKey of the XML Local Name taxonomy and a keyValue that is the value of the name attribute of the wsdl:service.

The bindingTemplates element of the businessService MUST include bindingTemplate elements that model the ports of the service, as described in the following sections.

2.4.3.1 Summary of Mapping

WSDL	UDDI
Service	businessService (categorized as service)
Namespace of Service	keyedReference in categoryBag
Local Name of Service	keyedReference in categoryBag; optionally also the name of the service

2.4.4 wsdl:port → uddi:bindingTemplate

A wsdl:port MUST be modeled as a uddi:bindingTemplate.

The minimum information that must be captured about a port is the binding that it implements, the portType that it implements, and its local name[5].

By capturing the binding, users can search for services that implement a specific binding. By capturing the portType, users can search for services that implement a particular portType without necessarily knowing the specific binding implemented by the service.

The wsdl:port information is captured as follows:

The bindingTemplate tModelInstanceDetails element MUST contain at least the following tModelInstanceInfo elements:

1. A tModelInstanceInfo with a tModelKey of the tModel that models the wsdl:binding that this port implements. The instanceParms of this tModelInstanceInfo MUST contain the wsdl:port local name.
2. A tModelInstanceInfo with a tModelKey of the tModel that models the wsdl:portType.

2.4.4.1 Summary of Mapping

WSDL	UDDI
port	bindingTemplate
Namespace	Captured in keyedReference of the containing businessService
Local Name of port	instanceParms of the tModelInstanceInfo relating to the tModel for the binding

[4] Users searching for a wsdl:service MUST NOT assume that the businessService name is the same as the wsdl:service local name. Because an existing businessService could be used, the wsdl:service local name MUST be specified as a keyedReference in the categoryBag.

[5] The namespace is captured in the businessService element.

Binding implemented by port	tModelInstanceInfo with tModelKey of the tModel corresponding to the binding
portType implemented by port	tModelInstanceInfo with tModelKey of the tModel corresponding to the portType

2.4.5 wsdl:port Address Extensions → uddi:bindingTemplate

The uddi:bindingTemplate MUST contain address information for the Web service. This information comes from the wsdl:port address extensibility element.

2.4.5.1 soap:address → uddi:accessPoint

A soap:address MUST be modeled as a uddi:accessPoint in the uddi:bindingTemplate that models the wsdl:port that contains the soap:address.

The soap:address information is captured as follows:

- The accessPoint value MUST be the value of the location attribute of the soap:address element.
- The URLType attribute of the accessPoint MUST correspond to the transport specified by the soap:binding, or “other” if no correspondence exists. In the case of the HTTP transport, for example, the URLType attribute MUST be “http”.

If “other” is used then a tModelInstanceInfo element referencing the appropriate vendor-defined transport tModel MUST be added to the bindingTemplate.

2.4.5.2 http:address → uddi:accessPoint

An http:address MUST be modeled as a uddi:accessPoint in the uddi:bindingTemplate that models the wsdl:port that contains the http:address.

The http:address information is captured as follows:

- The accessPoint value MUST be the value of the location attribute of the http:address element.
- The URLType attribute of the accessPoint MUST be “http”.

2.4.5.3 Other wsdl:port Address Extensions

Any other address extensibility element MUST be modeled as a uddi:accessPoint in the uddi:bindingTemplate that models the wsdl:port that contains the address extensibility element.

The address information is captured as follows:

- The accessPoint value MUST be the value of the location attribute of the address extensibility element. If the value of the location attribute cannot be mapped to the accessPoint value then the WSDL Implementation Document approach must be used. See Appendix A for further information.
- The URLType attribute of the accessPoint MUST correspond to the transport protocol associated with the URL, or “other” if none of the defined values of the attribute are appropriate.

2.5 Differences in mapping WSDL 1.1 in UDDI V3

This section describes the differences in the UDDI V3 view of the model that are a consequence of mandatory items in the UDDI V3 Specification and some optional extensions that can only be used with UDDI V3.

2.5.1 Mandatory Differences

The mandatory differences are:

1. Entities will have V3 keys rather than V2 keys.
2. An accessPoint has a useType attribute rather than a URLType attribute.

2.5.2 Optional Extensions

The optional extensions are:

1. Entities can have publisher-assigned keys.
2. A bindingTemplate can have a categoryBag. If a categoryBag is used, it MUST contain at least the following keyedReferences:
   a. A keyedReference with a tModelKey of the WSDL Entity Type taxonomy and a keyValue of “port”.
   b. A keyedReference with a tModelKey of the XML Namespace taxonomy and a keyValue of the target namespace of the wsdl:definitions element that contains the wsdl:port.
   c. A keyedReference with a tModelKey of the XML Local Name taxonomy and a keyValue of the local name of the wsdl:port.
3. An overviewURL can have an optional useType attribute, and a standard value of “wsdlInterface” has been defined to indicate “an abstract interface document”. This mapping assumes that “wsdlInterface” can be used with tModels that represent both portTypes and bindings.

2.5.3 Comparison to wsdlDeployment in UDDI V3 Specification

The UDDI V3 specification includes support for wsdlDeployment, which appears as both a value for the useType attribute of an accessPoint and as a categorization of a bindingTemplate. Use of wsdlDeployment is not compatible with this Technical Note as it assumes that no modeling of the WSDL is performed, nothing is known about the WSDL other than its URL.

3 A Complete Example

Consider the following WSDL sample based on the WSDL file presented in the WSDL 1.1 specification.[6] This example shows how this one WSDL file is decomposed into two tModels (one for the portType and one for the binding) and one businessService with one bindingTemplate. It then shows the kinds of UDDI API queries that can be used for the purpose of discovery.

3.1 WSDL Sample

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions
     name="StockQuote"
     targetNamespace="http://example.com/stockquote/"
     xmlns:tns="http://example.com/stockquote/"
     xmlns:xsd1="http://example.com/stockquote/schema/"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns="http://schemas.xmlsoap.org/wsdl/">
<import
     namespace="http://example.com/stockquote/schema/"
     location="http://location/schema.xsd" />
<message name="GetLastTradePriceInput">
     <part name="body" element="xsd1:TradePriceRequest" />
</message>
<message name="GetLastTradePriceOutput">
     <part name="body" element="xsd1:TradePrice" />
</message>
<portType name="StockQuotePortType">
     <operation name="GetLastTradePrice">
          <input message="tns:GetLastTradePriceInput" />
          <output message="tns:GetLastTradePriceOutput" />
     </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>
               <soap:body use="literal" />
          </input>
          <output>
               <soap:body use="literal" />
          </output>
     </operation>
</binding>

<service name="StockQuoteService">
     <documentation>My first service</documentation>
     <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
          <soap:address location="http://location/sample"/>
     </port>
</service>

</definitions>
```

Note that this WSDL file has one portType, one binding, one service, and one port. As such, this sample represents the simplest WSDL file. Also note that the location of this WSDL is at http://location/sample.wsdl.

[6] The WSDL sample in the WSDL 1.1 spec has an error (the port references the wrong binding QName). This WSDL sample has been corrected.

3.2 UDDI V2 Model

3.2.1 UDDI portType tModel

The WSDL portType entity maps to a tModel. The tModel name is the same as the WSDL portType local name. The tModel contains a categoryBag that specifies the WSDL namespace, and it indicates that the tModel is of type “portType”. The overviewDoc provides a pointer to WSDL file.

```
<tModel tModelKey="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3" >
<name>
     StockQuotePortType
</name>
<categoryBag>
     <keyedReference
          tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496"
          keyName=”portType namespace”
          keyValue=”http://example.com/stockquote/” />
     <keyedReference
          tModelKey="uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0"
          keyName=”WSDL type”
          keyValue="portType" />
</categoryBag>
<overviewDoc>
     <overviewURL>
          http://location/sample.wsdl
     <overviewURL>
<overviewDoc>
</tModel>
```

3.2.2 UDDI binding tModel

The WSDL binding entity maps to a tModel. The tModel name is the same as the WSDL binding local name. The tModel contains a categoryBag that specifies the WSDL namespace, it indicates that the tModel is of type “binding”, it supplies a pointer to the portType tModel, and it indicates what protocols are supported by the binding. The wsdlSpec keyedReference ensures that users can find the tModel using the conventions defined in the Version 1 Best Practice. The overviewDoc provides a pointer to the WSDL file.

```
<tModel tModelKey="uuid:49662926-f4a5-4ba5-b8d0-32ab388dadda">
<name>
     StockQuoteSoapBinding
</name>
<categoryBag>
     <keyedReference
          tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496"
           keyName=”binding namespace”
          keyValue=”http://example.com/stockquote/” />
     <keyedReference
          tModelKey="uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0"
           keyName=”WSDL type”
          keyValue="binding" />
     <keyedReference
          tModelKey="uuid:d3e8ef29-877e-3486-b9e2-46af338d6c85"
           keyName=”portType reference”
          keyValue="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3" />
     <keyedReference
          tModelKey="uuid:ee733f78-b289-3637-8ff5-1623ea4672dd"
           keyName=”SOAP protocol”
          keyValue= “uuid:057916d3-6ec1-3755-b847-013f0f514586” />
     <keyedReference
          tModelKey="uuid:4eeccd58-d3b0-3a6f-a466-9cce01cb1273"
           keyName=”HTTP transport”
          keyValue=" uuid:68DE9E80-AD09-469D-8A37-088422BFBC36" />
     <keyedReference
          tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
           keyName=”uddi-org:types”
          keyValue="wsdlSpec" />
</categoryBag>
<overviewDoc>
     <overviewURL>
          http://location/sample.wsdl
     </overviewURL>
</overviewDoc>
</tModel>
```

3.2.3 UDDI businessService and bindingTemplate

The WSDL service entity maps to a businessService, and the WSDL port entity maps to a bindingTemplate. Any information from the WSDL binding extensibility elements is also captured in the bindingTemplate. The businessService name should be a human-readable name. The businessService contains a categoryBag that indicates that this service represents a WSDL service, and it specifies the WSDL namespace and WSDL service local name. The bindingTemplate specifies the endpoint of the service, and it contains a set of tModelInstanceDetails. The first tModelInstanceInfo indicates that the service implements the StockQuoteSoapBinding and provides the WSDL port local name. The second tModelInstanceInfo indicates that the service implements the StockQuotePortType.

```
<businessService
     serviceKey="102b114a-52e0-4af4-a292-02700da543d4"
     businessKey="1e65ea29-4e0f-4807-8098-d352d7b10368">
<name>Stock Quote Service</name>
<bindingTemplates>
     <bindingTemplate
               bindingKey="f793c521-0daf-434c-8700-0e32da232e74”
               serviceKey="102b114a-52e0-4af4-a292-02700da543d4">
          <accessPoint URLType="http">
               http://location/sample
          </accessPoint>
          <tModelInstanceDetails>
               <tModelInstanceInfo
                    tModelKey="uuid:49662926-f4a5-4ba5-b8d0-32ab388dadda">
                    <description xml:lang="en">
                         The wsdl:binding that this wsdl:port implements.
                         The instanceParms specifies the port local name.
                    </description>
                    <instanceDetails>
                         <instanceParms>StockQuotePort</instanceParms>
                    </instanceDetails>
               </tModelInstanceInfo>
               <tModelInstanceInfo
                    tModelKey="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3">
                    <description xml:lang="en">
                         The wsdl:portType that this wsdl:port implements.
                    </description>
               </tModelInstanceInfo>
          </tModelInstanceDetails>
     </bindingTemplate>
</bindingTemplates>
<categoryBag>
    <keyedReference
          tModelKey=” uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0”
           keyName=”WSDL type”
          keyValue=”service” />
    <keyedReference
          tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496"
           keyName=”service namespace”
          keyValue=”http://example.com/stockquote/” />
    <keyedReference
          tModelKey=" uuid:451515ac-db54-3785-8937-114029f1d37b"
           keyName=”service local name”
          keyValue="StockQuoteService" />
</categoryBag>
</businessService>
```

3.3 Sample V2 Queries

This section shows how to perform various UDDI V2 queries given the model of the example.

3.3.1 Find tModel for portType name

Find the portType tModel for StockQuotePortType in the namespace http://example.com/stockquote/.

```
<find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
    <name>StockQuotePortType</name>
    <categoryBag>
        <keyedReference
          tModelKey="uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0"
          keyName="WSDL type"
          keyValue="portType"/>
        <keyedReference
          tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496"
          keyName="portType namespace"
          keyValue="http://example.com/stockquote/"/>
    </categoryBag>
</find_tModel>
```

This should return the tModelKey uuid:e8cf1163-8234-4b35-865f-94a7322e40c3.

3.3.2 Find bindings for portType

Find all bindings for StockQuotePortType.

```
<find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
    <categoryBag>
        <keyedReference
          tModelKey=" uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0"
          keyName="WSDL type"
          keyValue="binding"/>
        <keyedReference
          tModelKey="uuid:d3e8ef29-877e-3486-b9e2-46af338d6c85"
          keyName="portType reference"
          keyValue="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3"/>
    </categoryBag>
</find_tModel>
```

This should return the tModelKey uuid:49662926-f4a5-4ba5-b8d0-32ab388dadda.

3.3.3 Find Implementations of portType

Find all implementations of StockQuotePortType.

```
<find_binding generic="2.0" xmlns="urn:uddi-org:api_v2">
    <tModelBag>
        <tModelKey>uuid:e8cf1163-8234-4b35-865f-94a7322e40c3</tModelKey>
    </tModelBag>
</find_binding>
```

This should return the bindingKey f793c521-0daf-434c-8700-0e32da232e74.

3.3.4 Find implementations of binding

Find all implementations of StockQuoteSoapBinding.

```
<find_binding generic="2.0" xmlns="urn:uddi-org:api_v2">
    <tModelBag>
        <tModelKey>uuid:49662926-f4a5-4ba5-b8d0-32ab388dadda</tModelKey>
    </tModelBag>
</find_binding>
```

This should return the bindingKey f793c521-0daf-434c-8700-0e32da232e74.

3.3.5 Find SOAP Implementations of portType

Find all implementations of StockQuotePortType that support SOAP.

At least two queries are needed. The first query returns all the binding tModels that are categorized with SOAP. The second phase depends on whether or not other criteria are also required in the query.

3.3.5.1 No Other Criteria

In this case, only one other query is required. This query must specify “orAllKeys” and a tModelBag must be supplied which contains all the binding tModel keys returned by the first query.

3.3.5.2 Other Criteria

In this case, a query per binding tModel key is required and the default of “andAllKeys” must be used.

3.3.6 Find SOAP/HTTP Implementations of portType

This is similar to the previous case except that the first query must also include a category for the HTTP transport in addition to the SOAP protocol.

3.3.7 Find the portType of a binding

The portType of a binding is contained in the categoryBag of the binding tModel. No query is required once the tModel of the binding has been obtained. The keyValue of the keyedReference with tModelKey=”uuid:d3e8ef29-877e-3486-b9e2-46af338d6c85” contains the portType tModelKey.

4 References

4.1 Normative

[RFC2119]	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF RFC 2119, March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt.
[1]	Using WSDL in a UDDI Registry 1.08. Available at http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.pdf
[2]	Web Services Description Language (WSDL) 1.1, March 15, 2000. Available at http://www.w3.org/TR/wsdl
[3]	UDDI Version 2.03 Data Structure Reference, July 7, 2002. Available at http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.pdf.
[4]	UDDI Version 3.0 Published Specification, 19 July 2002. Available at http://www.uddi.org/pubs/uddi-v3.00-published-20020719.pdf.
[5]	XPointer xpointer() Scheme, W3C Working Draft, 10 July 2002. Available at http://www.w3.org/TR/2002/WD-xptr-xpointer-20020710/

A External WSDL Implementation Documents

There are multiple reasons why it may be desirable to support an external WSDL Implementation Document, among which are the following:

1. There are extensibility elements defined for the wsdl:service.
2. There is a wsdl:documentation element for a wsdl:port.
3. The address of a port may not be representable as a uddi:accessPoint value.
4. The authoritative source of the address is desired to be the WSDL document rather than UDDI.

The approach described here assumes that if any one of these reasons leads to the use of an external WSDL Deployment Document then the entire mapping described in this section is used.

There are two additional necessary pieces of information that must be captured to use external WSDL Implementation Documents:

1. The URL of the WSDL Implementation Document.
2. An indication that the port address must be obtained from the WSDL Implementation Document.

A.1 Capturing The URL

If an external WSDL Implementation Document is being used then the URL of this document must be used as the accessPoint value of each and every port of each and every service.

A.2 Obtaining the Port Address from WSDL

If a WSDL Implementation Document is being used then the bindingTemplate MUST contain sufficient information to identify the port address in the WSDL Implementation Document. The mapping described here MUST be used instead of the mapping defined in section 2.4.5.

In all cases where a WSDL Implementation Document is used, the URLType attribute of the accessPoint corresponding to each port MUST be “other”, and the value of the accessPoint MUST be the URL of the WSDL Implementation Document.

The bindingTemplate MUST contain a tModelInstanceInfo element with a tModelKey of the WSDL Address tModel. This tModelInstanceInfo element, in combination with the protocol and transport information from the binding tModel, provides the necessary information to locate and interpret the endpoint address.

A.3 Querying Services that use a WSDL Implementation Document

It is possible to query the services that have a WSDL Implementation Document by querying specifying the tModelKey of the WSDL Address tModel.

B Canonical tModels

This mapping introduces a number of canonical tModels that are used to represent WSDL metadata and relationships. These tModels are defined here.

B.1 WSDL Entity Type tModel

B.1.1 Design Goals

This mapping uses a number of UDDI entities to represent the various entities within a WSDL file. A mechanism is required to indicate what type of WSDL entity is being described by each UDDI entity. The WSDL Entity Type tModel provides a typing system for this purpose. This taxonomy is used to indicate that a UDDI entity represents a particular type of WSDL entity.

B.1.2 Definition

Name:	uddi.org:wsdl:types
Description:	WSDL Type Category System
V3 format key:	uddi:uddi.org:wsdl:types
V1,V2 format key:	uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0
Categorization:	categorization
Checked:	no

B.1.2.1 V2 tModel Structure

```
<tModel tModelKey="uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0" >
<name>uddi.org:wsdl:types</name>
<overviewDoc>
     <overviewURL>
     http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-
wsdl-20030319-wd.htm#wsdlTypes
     </overviewURL>
</overviewDoc>
<categoryBag>
     <keyedReference
          tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
          keyValue="unchecked"
     />
     <keyedReference
          tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
          keyValue="categorization"
     />
</categoryBag>
</tModel>
```

B.1.3 Valid Values

While this is an unchecked taxonomy, there are only four values that should be used with this taxonomy:

keyValue	Description	UDDI Entity
portType	Represents a UDDI entity categorized as a wsdl:portType	tModel
binding	Represents a UDDI entity categorized as a wsdl:binding	tModel

service	Represents a UDDI entity categorized as a wsdl:service	businessService
port	Represents a UDDI entity categorized as a wsdl:port	bindingTemplate (v3 only)

B.1.4 Example of Use

A V2 tModel representing a portType tModel would have a categoryBag representing its type:

```
<categoryBag>
     <keyedReference
          tModelKey="uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0"
          keyName="WSDL Entity type"
          keyValue="portType"
     />
…
</categoryBag>
```

B.2 XML Namespace tModel

B.2.1 Design Goals

A namespace provides necessary qualifying information about a technical concept or model. The XML Namespace tModel provides a mechanism to associate a namespace with a UDDI entity. This taxonomy describes a UDDI entity by specifying the target namespace of the description file (i.e., a WSDL file or XML Schema file) that describes the entity. *More than one tModel might be categorized with the same namespace* – in fact, this mapping would be quite common, as many WSDL files use a common target namespace for <wsdl:portType>, <wsdl:binding>, and <wsdl:service> elements.

B.2.2 Definition

Name:	uddi.org:xml:namespace
Description:	A category system used to indicate namespaces
V3 format key:	uddi:uddi.org:xml:namespace
V1, V2 format key:	uuid:fb5fb934-9a3d-39dc-9871-271f64780496
Categorization:	categorization
Checked:	no

B.2.2.1 V2 tModel Structure

```
<tModel tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496">
     <name>uddi.org:xml:namespace</name>
     <overviewDoc>
          <overviewURL>
          http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-
tn-wsdl-20030319-wd.htm #xmlNamespace
          </overviewURL>
     </overviewDoc>
     <categoryBag>
          <keyedReference
               tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
               keyValue="unchecked"
          />
         <keyedReference
               tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
               keyValue="categorization"
          />
     </categoryBag>
</tModel>
```

B.2.3 Valid Values

The values used in this taxonomy are namespaces of type “anyURI”. The content of keyValue in a keyedReference that refers to this tModel is the target namespace of the WSDL file that describes the WSDL entity described by the UDDI entity.

B.2.4 Example of Use

A namespace keyedReference would be as follows:

```
<categoryBag>
     <keyedReference
          tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496"
          keyName="namespace"
          keyValue="urn:foo"
     />
…
</categoryBag>
```

B.3 XML Local Name tModel

B.3.1 Design Goals

Each WSDL entity is identified by its name attribute, and this identification information needs to be captured in the mapped UDDI entities. In the case of wsdl:portType and wsdl:binding, the name attribute is mapped to the tModel name element. However, it isn’t appropriate to map the wsdl:service name attribute to the name element of the businessService, and, in the case of wsdl:port, the bindingTemplate entity does not have a name element. The XML Local Name tModel provides a mechanism to indicate the name attribute for these two constructs.

B.3.2 Definition

Name:	uddi.org:xml:localName
Description:	A category system used to indicate XML local names
V3 format key:	uddi:uddi.org:xml:localName
V1,V2 format key:	uuid:451515ac-db54-3785-8937-114029f1d37b
Categorization:	categorization
Checked:	no

B.3.2.1 V2 tModel Structure

```
<tModel tModelKey="uuid:451515ac-db54-3785-8937-114029f1d37b" >
     <name>uddi.org:xml:localName</name>
     <overviewDoc>
          <overviewURL>
          http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-
tn-wsdl-20030319-wd.htm#xmlLocalName
          </overviewURL>
     </overviewDoc>
     <categoryBag>
          <keyedReference
               tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
               keyValue="unchecked"
          />
          <keyedReference
```

```
              tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
              keyValue="categorization"
          />
     </categoryBag>
</tModel>
```

B.3.3 Valid Values

The values used in this taxonomy are XML local names. The content of keyValue in a keyedReference that refers to this tModel is equal to the name attribute of the WSDL entity described by the UDDI entity.

B.3.4 Example of Use

A local name keyedReference would be as follows:

```
<categoryBag>
     <keyedReference
          tModelKey="uuid:451515ac-db54-3785-8937-114029f1d37b"
         keyName="Local service name"
          keyValue="StockQuoteService"
          />
…
</categoryBag>
```

B.4 WSDL portType Reference tModel

B.4.1 Design Goals

WSDL entities exhibit many relationships. Specifically, a wsdl:port describes an implementation of a wsdl:binding, and a wsdl:binding describes a binding of a particular wsdl:portType. These same relationships must be expressed in the UDDI mapping. UDDI provides a built-in mechanism, via the tModelInstanceInfo structure, to associate a bindingTemplate with a tModel. But UDDI does not provide a built-in mechanism to describe a relationship between two tModels. The WSDL portType Reference category system provides a mechanism to indicate that a wsdl:binding tModel is a binding of a specific wsdl:portType tModel.

B.4.2 Definition

Name:	uddi.org:wsdl:portTypeReference
Description:	A category system used to reference a wsdl:portType tModel
V3 format key:	uddi:uddi.org:wsdl:portTypeReference
V1,V2 format key:	uuid:d3e8ef29-877e-3486-b9e2-46af338d6c85
Categorization:	categorization
Checked:	no

B.4.2.1 V2 tModel Structure

```
<tModel tModelKey="uuid:d3e8ef29-877e-3486-b9e2-46af338d6c85" >
     <name>uddi.org:wsdl:portTypeReference</name>
     <description xml:lang="en">
This tModel is a taxonomy tModel that can be used to identify a relationship
to a portType tModel.
     </description>
     <overviewDoc>
          <overviewURL>
     http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-
wsdl-20030319-wd.htm#portTypeReference
          </overviewURL>
     </overviewDoc>
     <categoryBag>
          <keyedReference
              tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
```

```
               keyValue="categorization"
          />
         <keyedReference
               tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
               keyValue="unchecked"
          />
     </categoryBag>
</tModel>
```

B.4.3 Valid Values

Valid values for this taxonomy are tModelKeys. The content of keyValue in a keyedReference that refers to this tModel is the tModelKey of the wsdl:portType tModel being referenced.

B.4.4 Example of Use

One would add the following keyedReference to signify that a wsdl:binding implements a specific portType:

```
<categoryBag>
     <keyedReference
          tModelKey="uuid:d3e8ef29-877e-3486-b9e2-46af338d6c85"
          keyName="wsdl:portType Reference"
          keyValue="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3"
     />
…
</categoryBag>
```

Note that the keyValue is a tModelKey, which, if queried for using get_tModelDetail, would return the tModel that represents the portType.

B.5 SOAP Protocol tModel

B.5.1 Design Goals

Web services can support a wide variety of protocols. Users looking for Web services may want to search for Web services that support a specific protocol. The SOAP Protocol tModel can be used to indicate that a Web service supports the SOAP 1.1 protocol. This tModel correlates to the http://schemas.xmlsoap.org/wsdl/soap/ namespace identified in the WSDL Specification.

B.5.2 Definition

Name:	uddi.org:protocol:soap
Description:	A tModel that represents the SOAP 1.1 protocol
V3 format key:	uddi:uddi.org:protocol:soap
V1,V2 format key:	uuid:057916d3-6ec1-3755-b847-013f0f514586
Categorization:	protocol

B.5.2.1 tModel Structure

```
<tModel tModelKey="uuid:057916d3-6ec1-3755-b847-013f0f514586">
     <name>uddi.org:protocol:soap</name>
     <overviewDoc>
          <overviewURL>
          http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-
tn-wsdl-20030319-wd.htm#soap
          </overviewURL>
     </overviewDoc>
     <categoryBag>
          <keyedReference
               tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
               keyValue="protocol"
          />
     </categoryBag>
```

```
</tModel>
```

B.5.3 Example of Use

The SOAP Protocol tModel is used to categorise a binding tModel that corresponds to a wsdl:binding that supports the SOAP 1.1 protocol.

```
<tModel tModelKey="uuid:49662926-f4a5-4ba5-b8d0-32ab388dadda">
<name>…</name>
<categoryBag>
     <keyedReference
          tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496"
           keyName=”binding namespace”
          keyValue=”http://example.com/stockquote/” />
     <keyedReference
          tModelKey="uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0"
           keyName=”WSDL type”
          keyValue="binding" />
     <keyedReference
          tModelKey="uuid:d3e8ef29-877e-3486-b9e2-46af338d6c85"
           keyName=”portType reference”
          keyValue="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3" />
     <keyedReference
          tModelKey="uuid:ee733f78-b289-3637-8ff5-1623ea4672dd"
           keyName=”SOAP protocol”
          keyValue= “uuid:057916d3-6ec1-3755-b847-013f0f514586” />
     <keyedReference
          tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
           keyName=”types”
          keyValue="wsdlSpec" />
</categoryBag>
<overviewDoc>
     <overviewURL>http://location/sample.wsdl</overviewURL>
</overviewDoc>
</tModel>
```

B.6 HTTP Protocol tModel

B.6.1 Design Goals

Web services can support a wide variety of protocols. Users looking for Web services may want to search for Web services that support a specific protocol. The HTTP Protocol tModel can be used to indicate that a Web service supports the HTTP protocol. Note that this tModel is different from the HTTP Transport tModel. This tModel represents a protocol; for example, it represents the http://schemas.xmlsoap.org/wsdl/http/ namespace in the WSDL specification. The HTTP Transport tModel represents a transport.

B.6.2 Definition

Name:	uddi.org:protocol:http
Description:	A tModel that represents the HTTP protocol
V3 format key:	uddi:uddi.org:protocol:http
V1,V2 format key:	uuid:e01a4d7f-b7d6-337d-b47c-2cf3e84edd75
Categorization:	protocol

B.6.2.1 V2 tModel Structure

```
<tModel tModelKey="uuid:e01a4d7f-b7d6-337d-b47c-2cf3e84edd75">
     <name>uddi.org:protocol:http</name>
     <overviewDoc>
          <overviewURL>
               http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-
spec-tc-tn-wsdl-20030319-wd.htm#http
          </overviewURL>
     </overviewDoc>
     <categoryBag>
```

```
          <keyedReference
               tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
               keyValue="protocol"
          />
     </categoryBag>
</tModel>
```

B.6.3 Example of Use

The HTTP Protocol tModel is used to categorise a binding tModel that corresponds to a wsdl:binding that supports the HTTP protocol.

```
<tModel tModelKey="uuid:49662926-f4a5-4ba5-b8d0-32ab388dadda">
<name>
     StockQuoteSoapBinding
</name>
<categoryBag>
     <keyedReference
          tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496"
           keyName="binding namespace"
          keyValue="http://example.com/stockquote/" />
     <keyedReference
          tModelKey="uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0"
           keyName="WSDL type"
          keyValue="binding" />
     <keyedReference
          tModelKey="uuid:d3e8ef29-877e-3486-b9e2-46af338d6c85"
           keyName="portType reference"
          keyValue="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3" />
     <keyedReference
          tModelKey="uuid:ee733f78-b289-3637-8ff5-1623ea4672dd"
           keyName="HTTP protocol"
          keyValue= "uuid:e01a4d7f-b7d6-337d-b47c-2cf3e84edd75" />
     <keyedReference
          tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
           keyName="types"
          keyValue="wsdlSpec" />
</categoryBag>
<overviewDoc>
     <overviewURL>
          http://location/sample.wsdl
     </overviewURL>
</overviewDoc>
</tModel>
```

B.7 Protocol Categorization

B.7.1 Design Goals

A Web service may communicate using a variety of protocols. A WSDL binding binds a portType to a specific protocol. A user may wish to search for bindings that implement a specific protocol. The Protocol Categorization tModel provides a mechanism to capture this protocol information in the UDDI binding tModel.

B.7.2 Definition

Name: uddi-org:wsdl:categorization:protocol

Description: Category system used to describe the protocol supported by a wsdl:binding.

V3 format key: uddi:uddi.org:wsdl:categorization:protocol

V1,V2 format key: uuid:ee733f78-b289-3637-8ff5-1623ea4672dd

Categorization: categorization

Checked: no

B.7.2.1 V2 tModel Structure

```
<tModel tModelKey="uuid:uddi.org:wsdl:categorization:protocol">
    <name>uddi-org:wsdl:categorization:protocol</name>
    <overviewDoc>
        <overviewURL>
            http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-
tc-tn-wsdl-20030319-wd.htm#protocol
        </overviewURL>
    </overviewDoc>
    <categoryBag>
        <keyedReference keyName=”types”
                        keyValue="categorization"
                        tModelKey=" uuid:c1acf26d-9672-4404-9d70-
39b756e62ab4”/>
        <keyedReference keyName=”types”
                        keyValue="unchecked"
                        tModelKey=" uuid:c1acf26d-9672-4404-9d70-
39b756e62ab4”/>
     </categoryBag>
</tModel>
```

B.7.3 Example of Use

The Protocol category scheme is used to indicate the protocol that a binding supports.

```
<tModel tModelKey="uuid:49662926-f4a5-4ba5-b8d0-32ab388dadda">
    <name>StockQuoteSoapBinding</name>
    <categoryBag>
        <keyedReference
            tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496"
            keyName=”binding namespace”
            keyValue=”http://example.com/stockquote/”/>
         <keyedReference
            tModelKey="uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0"
            keyName=”WSDL type”
            keyValue="binding"/>
        <keyedReference
            tModelKey="uuid:d3e8ef29-877e-3486-b9e2-46af338d6c85"
            keyName=”portType reference”
            keyValue="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3"/>
        <keyedReference
            tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
            keyName=”types”
            keyValue="wsdlSpec"/>
        <keyedReference
            tModelKey="uuid:ee733f78-b289-3637-8ff5-1623ea4672dd"
            keyName=”WSDL binding supports the SOAP protocol”
            keyValue="uddi:057916d3-6ec1-3755-b847-013f0f514586"/>
    </categoryBag>
    <overviewDoc>
        <overviewURL>http://location/sample.wsdl<overviewURL>
    <overviewDoc>
</tModel>
```

B.8 Transport Categorization

B.8.1 Design Goals

A Web service may communicate using a variety of transports. A WSDL binding binds a portType to a specific transport protocol. A user may wish to search for bindings that implement a specific transport protocol. The Transport Categorization tModel provides a mechanism to capture this transport information in the UDDI binding tModel.

B.8.2 Definition

Name: uddi-org:wsdl:categorization:transport

Description: Category system used to describe the transport supported by a wsdl:binding.

V3 format key:	uddi:uddi.org:wsdl:categorization:transport
V1,V2 format key:	uuid:4eeccd58-d3b0-3a6f-a466-9cce01cb1273
Categorization:	categorization
Checked:	no

B.8.2.1 V2 tModel Structure

```
<tModel tModelKey="uuid:uddi.org:wsdl:categorization:transport">
    <name>uddi-org:wsdl:categorization:transport</name>
    <overviewDoc>
        <overviewURL>
            http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-
tc-tn-wsdl-20030319-wd.htm#transport
        </overviewURL>
    </overviewDoc>
    <categoryBag>
        <keyedReference keyName="types"
                        keyValue="categorization"
                        tModelKey="uuid:c1acf26d-9672-4404-9d70-
39b756e62ab4"/>
        <keyedReference keyName="types"
                        keyValue="unchecked"
                        tModelKey="uuid:c1acf26d-9672-4404-9d70-
39b756e62ab4"/>
     </categoryBag>
</tModel>
```

B.8.3 Example of Use

The Transport category scheme is used to indicate the transport that a binding supports.

```
<tModel tModelKey="uuid:49662926-f4a5-4ba5-b8d0-32ab388dadda">
    <name>StockQuoteSoapBinding</name>
    <categoryBag>
        <keyedReference
            tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496"
            keyName="binding namespace"
            keyValue="http://example.com/stockquote/"/>
         <keyedReference
            tModelKey="uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0"
            keyName="WSDL type"
            keyValue="binding"/>
        <keyedReference
            tModelKey="uuid:d3e8ef29-877e-3486-b9e2-46af338d6c85"
            keyName="portType reference"
            keyValue="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3"/>
        <keyedReference
            tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
            keyName="types"
            keyValue="wsdlSpec"/>
        <keyedReference
            tModelKey="uuid:hashed key"
            keyName="WSDL binding protocol"
            keyValue="uddi:057916d3-6ec1-3755-b847-013f0f514586"/>
        <keyedReference
            tModelKey="uuid:4eeccd58-d3b0-3a6f-a466-9cce01cb1273"
            keyName="WSDL transport protocol"
            keyValue="uuid:68DE9E80-AD09-469D-8A37-088422BFBC36"/>
    </categoryBag>
    <overviewDoc>
        <overviewURL>http://location/sample.wsdl<overviewURL>
    <overviewDoc>
</tModel>
```

B.9 WSDL Address tModel

B.9.1 Design Goals

A service provider may not want to specify the address of a service port in the uddi:accessPoint element and instead require the user to retrieve a WSDL file to obtain the service address. UDDI V2 does not provide a built-in mechanism to indicate that the endpoint address should be obtained from a WSDL file. This document describes an approach to provide a mechanism using existing UDDI V2 features. This approach requires that the bindingTemplate indicate that the WSDL file must be retrieved to obtain the address information. The WSDL Address tModel provides such a mechanism. A V2 bindingTemplate includes a tModelInstanceInfo element that references this tModel to indicate that the address information must be retrieved from the WSDL file.

B.9.2 Definition

Name:	uddi-org:wsdl:address
Description:	A tModel used to indicate the WSDL address option
V3 format key:	uddi:uddi.org:wsdl:address
V1,V2 format key:	uuid:2646df99-ec31-3c67-80e2-5743d0c0e829
Categorization:	none

B.9.2.1 V2 tModel Structure

```
<tModel tModelKey="uuid:" >
     <name>uddi-org:wsdl:address</name>
     <description xml:lang="en">
This tModel is used to specify the URL fact that the address must be obtained
from the WSDL deployment file.
     </description>
     <overviewDoc>
          <overviewURL>
          http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-
tn-wsdl-20030319-wd.htm#Address
          </overviewURL>
     </overviewDoc>
</tModel>
```

B.9.3 Valid Values

There are no valid values associated with this tModel, it is simply a marker.

B.9.4 Example of Use

If a service provider requires the user to retrieve the service endpoint from a WSDL file rather than from the UDDI bindingTemplate, the accessPoint element must have a value of “WSDL” and a URLType attribute value of “other”:

```
<bindingTemplate
          bindingKey="f793c521-0daf-434c-8700-0e32da232e74”
          serviceKey="102b114a-52e0-4af4-a292-02700da543d4">
     <accessPoint URLType="other">WSDL</accessPoint>
<tModelInstanceDetails>
     <tModelInstanceInfo
               tModelKey="uuid:2646df99-ec31-3c67-80e2-5743d0c0e829">
     <tModelInstanceInfo>
     ...
     </tModelInstanceDetails>
</bindingTemplate
```

C Using XPointer in overviewURL

C.1 XPointer Syntax

In this mapping of WSDL to UDDI, a UDDI entity describes a particular element within a WSDL file. The particular WSDL element described SHOULD be determined by using the metadata contained within the entity's categoryBag, and either the UDDI entity's name or the instanceParms value specified in the tModelInstanceInfo that relates to the binding that a port implements. Alternatively, the overviewURL value MAY contain a fragment identifier that identifies the particular WSDL element.

As the WSDL 1.1 schema does not allow for id attributes on WSDL elements, we cannot simply use a fragment identifier of the form #foo.

If the optional fragment identifier is used, the syntax defined by XPointer [5] MUST be used for the fragment identifier. It should be noted that at the time of writing this Technical Note, XPointer is a set of Working Draft documents and is therefore subject to change.

C.1.1 Example of Use

Referring to the WSDL Sample in Section 3.1, the StockQuotePortType tModel may reference the wsdl:portType element directly from the overviewURL using XPointer syntax.

```
<tModel tModelKey="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3" >
<name>
     StockQuotePortType
</name>
<categoryBag>
     <keyedReference
          tModelKey="uuid:fb5fb934-9a3d-39dc-9871-271f64780496"
           keyName=”portType target namespace”
          keyValue=”http://example.com/stockquote/”
     />
     <keyedReference
          tModelKey="uuid:5b67c4b8-fbb8-3681-9c63-bf6b0c838dd0"
           keyName=”WSDL Entity Type”
          keyValue="portType"
     />
</categoryBag>
<overviewDoc>
     <overviewURL>
     http://location/sample.wsdl#xmlns(wsdl=http://schemas.xmlsoap.org/wsdl/)
xpointer(/wsdl:definitions/wsdl:portType[@name=”StockQuotePortType”]).
     <overviewURL>
<overviewDoc>
</tModel>
```

D Acknowledgments

The following individuals were members of the committee during the development of this technical note:

Andrew Hately, IBM
Sam Lee, Oracle
Alok Srivastava, Oracle
Claus von Riegen, SAP

E Revision History

Rev	Date	By Whom	What
20021022	22 Oct 2002	John Colgrave and Karsten Januszewski	First draft of V2.0 TN
20021114	14 Nov 2002	Tony Rogers and Anne Thomas Manes	Second draft of V2.0 TN for TC discussion
20030319	19 Mar 2003	John Colgrave, Anne Thomas Manes and Tony Rogers	Final draft of V2.0 TN for TC review

F Notices