

THALES	WSDL 2.0 to UDDI mapping WSDL-S/SAWSDL to UDDI mapping		
Type	Technical Note	Date	29/05/06
Author	Pierre Châtel - SC2 Group	Pages	31
Abstract	This document is a technical note that defines a new approach to using WSDL 2.0 and WSDL-S/SAWSDL in a UDDI registry.		
Status	This document is updated periodically on no particular schedule. It has no official standing. Comments on this technical note should be sent to the author at pierre.chatel@fr.thalesgroup.com .		

Recipients		
Name	Mail	Goal
SAWSDL Mailing List Kunal Verma	public-ws-semann@w3.org verma@cs.uga.edu	For Information For Information

Document evolution				
Version	Status	Date	Author	Observations
0	Creation	29/05/06	Pierre Châtel	Document creation
0.1	N/A	30/05/06	Pierre Châtel	WSDL 2.0 mapping added
0.2-0.4	N/A	31/05/06 – 02/06/06	Pierre Châtel	WSDL 2.0 mapping modified
0.5-0.6	N/A	05/06/06 – 06/06/06	Pierre Châtel	WSDL-S mapping modified
0.7	N/A	07/06/06	Pierre Châtel	WSDL-S mapping modified and canonical tModels defined
0.8	N/A	09/06/06	Pierre Châtel	WSDL-S mapping modified and specification of all the canonical tModels
0.9	N/A	11/06/06 – 13/06/06	Pierre Châtel	French to English translation
1.0	Draft	13/06/06	Pierre Châtel	First Public Draft

Table of contents

1.	Introduction.....	4
2.	WSDL 1.1 to UDDI mapping	4
3.	WSDL 2.0 to UDDI mapping	6
3.1.	Correspondences between WSDL 1.1 and 2.0 components	6
3.2.	Mapping	7
3.2.1.	New Canonical tModels	8
3.2.2.	References to WSDL Components from UDDI.....	8
3.2.3.	WSDL 2.0 Import and Include directives.....	9
3.2.4.	wsdl:interface->uddi:tModel	9
3.2.5.	wsdl:binding->uddi:tModel.....	10
3.2.6.	wsdl:service->uddi:businessService.....	11
3.2.7.	wsdl:endpoint->uddi:bindingTemplate	12
3.2.8.	WSDL 2.0 Components not mapped to UDDI.....	13
4.	WSDL-S to UDDI Mapping	14
4.1.	WSDL-S version.....	14
4.2.	Semantic annotations defined by WSDL-S	14
4.3.	Semantic information that will be mapped to UDDI.....	14
4.4.	WSDL 2.0 to UDDI mapping extensions	15
4.5.	Mapping	15
4.5.1.	New Canonical tModels	16
4.5.2.	wsdl-s:category-> uddi:tModel (interface type) [extension].....	16
4.5.3.	wsdl:interface->uddi:tModel [extension]	17
4.5.4.	wsdl:operation->uddi:tModel.....	18
5.	Appendix A: Canonical tModels.....	22
5.1.	WSDL Interface Reference.....	22
5.1.1.	Design Goals	22
5.1.2.	Definition.....	22
5.1.3.	V2 tModel Structure	22
5.1.4.	Valid values	22
5.1.5.	Example of Use	22
5.2.	WSDL Operation Reference	23
5.2.1.	Design Goals	23
5.2.2.	Definition.....	23
5.2.3.	V2 tModel Structure	23
5.2.4.	Valid values	24
5.2.5.	Example of Use	24
5.3.	Functionnal Concept	24
5.3.1.	Design Goals	24
5.3.2.	Definition.....	24
5.3.3.	V2 tModel Structure	24
5.3.4.	Valid values	25
5.3.5.	Example of Use	25
5.4.	Input.....	25
5.4.1.	Design Goals	25
5.4.2.	Definition.....	25
5.4.3.	V2 tModel Structure	25
5.4.4.	Valid values	26
5.4.5.	Example of Use	26

5.5.	Output	26
5.5.1.	Design Goals	26
5.5.2.	Definition.....	26
5.5.3.	V2 tModel Structure	26
5.5.4.	Valid values	27
5.5.5.	Example of Use	27
5.6.	Precondition	27
5.6.1.	Design Goals	27
5.6.2.	Definition.....	27
5.6.3.	V2 tModel Structure	27
5.6.4.	Valid values	28
5.6.5.	Example of Use	28
5.7.	Effect.....	28
5.7.1.	Design Goals	28
5.7.2.	Definition.....	28
5.7.3.	V2 tModel Structure	28
5.7.4.	Valid values	29
5.7.5.	Example of Use	29
6.	Appendix B: Comparison between WSDL 1.1 and WSDL 2.2 component models	30
7.	Appendix C: WSDL 2.0 Components hierarchical view	31

1. Introduction

Three versions of WSDL-S are actually available: each one specifies how and when to annotate a WSDL service definition in order to add *references* to semantic elements (concepts in ontologies):

- One based on the 1.1 version of the WSDL specification which has been created as an interim step of defining the final WSDL-S standard by adding semantics to WSDL 1.1 using the extensibility elements offered by the WSDL 1.1 standard.
- An other based on the 2.0 version of the WSDL specification. It's the version that corresponds to the last W3C proposal of WSDL-S.
- At the time of the writing of this report, the future development of WSDL-S changed hands and is now being maintained by a specific W3C working group¹. It was beforehand developed by the LSDID laboratory. Following this change, WSDL-S has been renamed to SAWSDL (« Semantic Annotations for WSDL »). See [SAWSDL] for more details on the subject.

We will continue to use the WSDL-S denomination in the rest of this technical note.

Although the current version of the SAWSDL specification does not bring many appreciable change to the last specification of WSDL-S, the mapping described by this technical note is based on this version of SAWSDL. Some elements from the last WSDL-S specification that were removed from the current SAWSDL extension-set are still mapped for compatibility reasons, see [4.1. WSDL-S version].

Since WSDL-s has been defined as an extension to the WSDL 2.0 specification, a preliminary step to the WSDL-S to UDDI mapping we will define a WSDL 2.0 to UDDI mapping in [3. WSDL 2.0 to UDDI mapping]. It will keep all the mandatory elements from the previous WSDL 1.1 to UDDI mapping and will describe all the necessary syntactic and semantic aspects of a web service description.

The content of this document is organized as follows:

- A brief recap on the WSDL 1.1 to UDDI mapping.
- A study of the correspondences between WSDL 1.1 and 2.0 structures.
- The development of a brand new WSDL 2.0 to UDDI mapping.
- The development of a WSDL-S to UDDI mapping as an extension to the WSDL 2.0 mapping.

2. WSDL 1.1 to UDDI mapping

There is an OASIS committee technical note [UDDIMAP] that specifies an exact implementation of a WSDL 1.1 to UDDI mapping. We are going to ground our own WSDL 2.0 mapping on this technical note.

A WSDL web service definition can be decomposed in two main parts: one called the “abstract definition” of the service, the other focused on the implementation aspects. In the following table, we present a quick summary of the mapping concerning the two parts that was specified by the OASIS technical note:

¹ <http://www.w3.org/2002/ws/sawSDL/>

<i>WSDL Element</i>	<i>Description</i>	<i>Corresponding UDDI element</i>	<i>Description</i>
Abstract definition			
PortType	A portType is an abstract collection of operations that may be supported by one or more Web services	TModel	tModels represent unique concepts or constructs. They are used to describe compliance with a specification, a concept, or a shared design.
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	tModel	tModels represent unique concepts or constructs. They are used to describe compliance with a specification, a concept, or a shared design.
Service implementation			
Service	WSDL defines a Web service implementation as a service with a collection of named ports	BusinessService	Descriptive information about a particular family of technical services
Port	Each port implements a particular portType using the protocols defined by a named binding	BindingTemplate	Technical information about a service entry point and construction specification

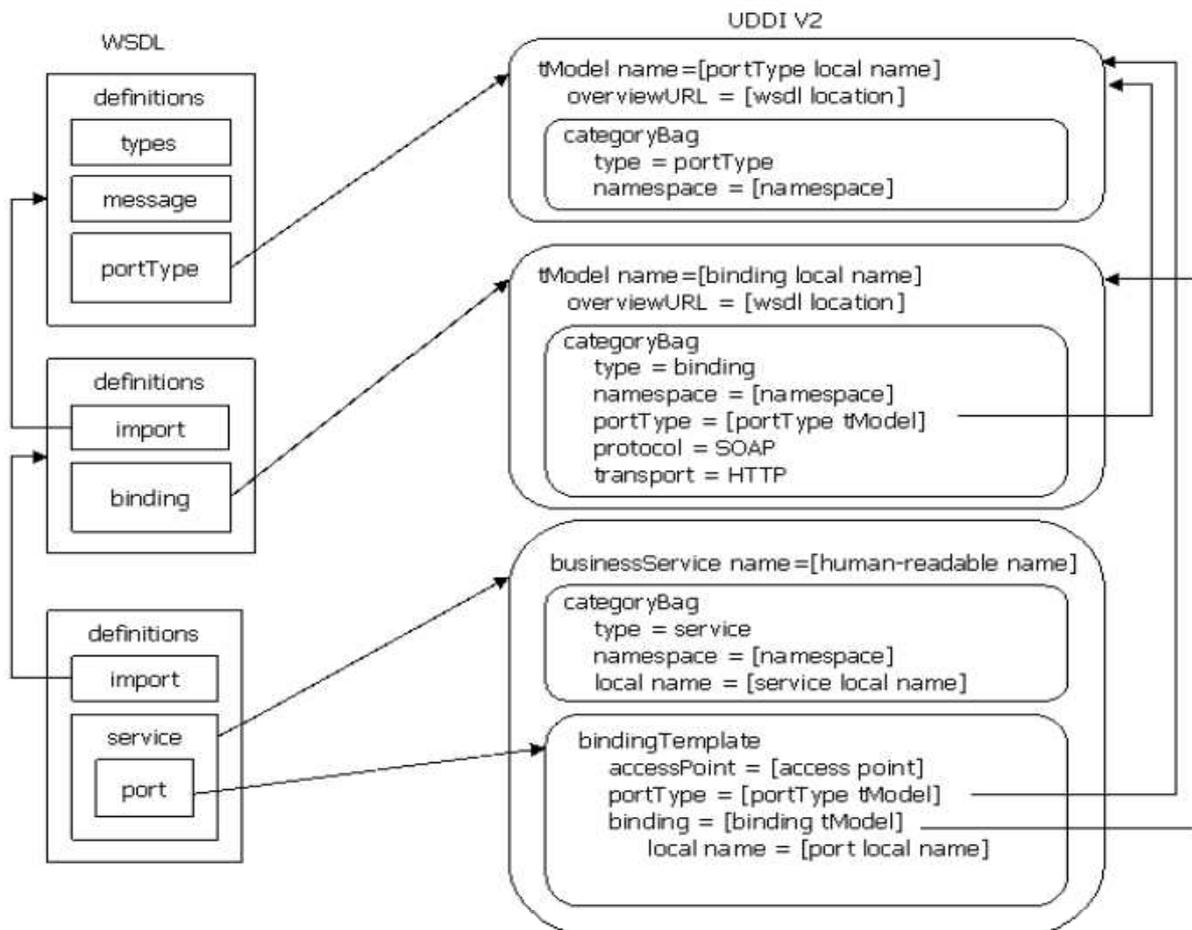


Figure 1 - Recapitulatory of the WSDL 1.1 to UDDI mapping

3.1. Correspondences between WSDL 1.1 and 2.0 components

Refer to [Appendix B: Comparison between WSDL 1.1 and WSDL 2.2 component models] for more details.

WSDL 1.1 component	WSDL 2.0 component	Note
Message	N/A	The <i>message</i> element was removed from the WSDL 2.0 specification. You now have to use xsd type declarations to format operations' inputs and outputs.
PortType	Interface	<i>PortType</i> was renamed to <i>interface</i> in WSDL 2.0. It is now possible to specify a set of declared <i>interface</i> components which a given <i>interface</i> extends.
Operation (from portType)	Operation (from interface)	A WSDL 2.0 <i>operation</i> is semantically equivalent to a WSDL 1.1 <i>operation</i> . But its syntax and implementation are slightly different. It is now required to specify a message exchange pattern and operations

<i>WSDL 1.1 component</i>	<i>WSDL 2.0 component</i>	<i>Note</i>
		are now making direct references to xsd types without using a <i>message</i> construct.
Binding	Binding	The new <i>binding</i> component doesn't necessarily need to be associated to a given <i>interface</i> element (it is now an optional attribute). This new kind of <i>binding</i> can be used on multiple interfaces.
Service	Service	A major difference between the WSDL 1.1 and 2.0 version of the <i>service</i> component is that it is now required to specify the <i>interface</i> that the service is an instance of. That makes it possible to guarantee a logical bond between the various endpoints of a given <i>service</i> . Whereas, in the WSDL 1.1 specification, it was possible for a given <i>service</i> to bring together <i>ports</i> implementing different portTypes/interfaces.
Port	Endpoint	<i>Port</i> was renamed to <i>endpoint</i> in the WSDL 2.0 specification. There are some syntax variations compared to the WSDL 1.1 version.

3.2. Mapping

This mapping is **not compatible** with the following WSDL 1.1 based mappings:

1. The basic mapping which is registered as a "Best Practice" by the OASIS committee. <http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.htm>
2. The second mapping type which is more elaborate but has not been validated yet. <http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm>

You can see in Figure 2 a simplified view of the mapping that is described in this section. It indicates the correspondences between the WSDL 2.0 model and the fixed UDDI structures.

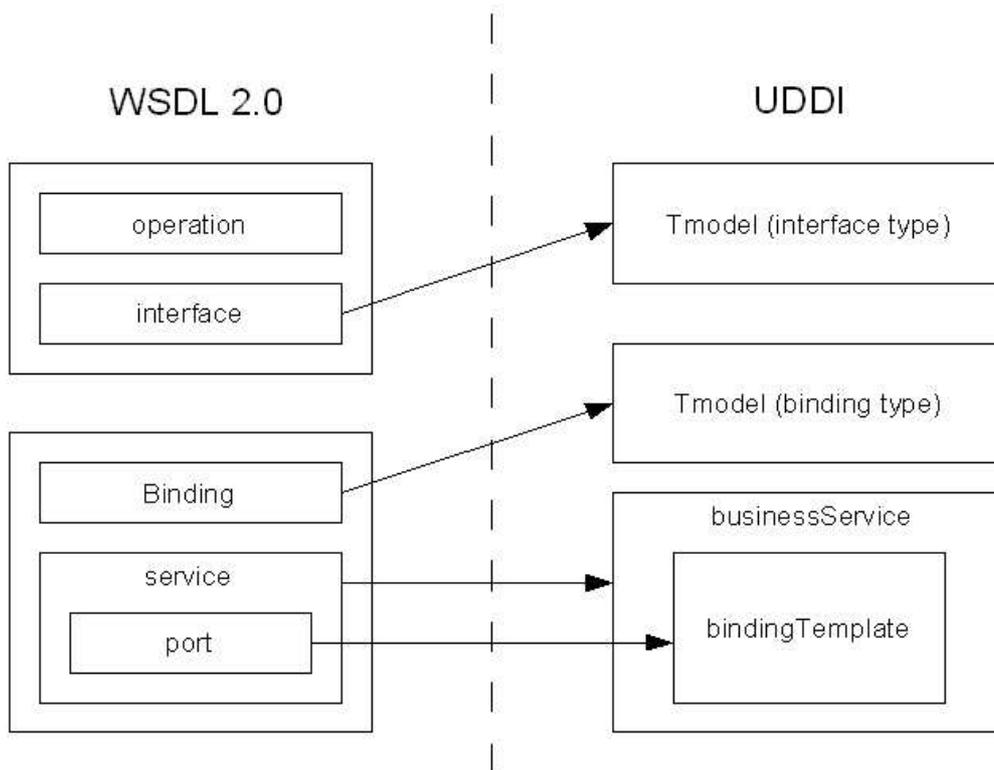


Figure 2 – Simplified representation of the correspondences between WSDL 2.0 and UDDI structures

3.2.1. New Canonical tModels

This mapping introduces a number of canonical tModels that are used to represent WSDL 2.0 metadata and relationships. These tModels MUST be registered in the UDDI registry to support this mapping. Only the V1/V2 keys are given for these tModels.

We reuse all the canonical tModels defined by [UDDIMAP] to which we add the following tModels [see Appendix A: Canonical tModels for more details]:

- *WSDL Interface Reference* (based on the WSDL portType Reference tModel)

3.2.2. References to WSDL Components from UDDI

A UDDI entity normally references technical specifications using the *overviewURL* element. As noted above, in this mapping a single WSDL document maps 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 UDDI entity, using the particular mapping for the namespace name and local name applicable to the particular UDDI entity type..

To sum up, 3 values are captured when mapping to a UDDI entity:

- The local name of the originating WSDL element
- The target namespace of this element (a mandatory attribute of the *service description* since WSDL 2.0).
- The location of the service definition itself via the *overviewURL* attribute.

3.2.3. WSDL 2.0 Import and Include directives

The WSDL 2.0 specification is introducing a new dependencies handling mechanism which complements the *import* element already defined by WSDL 1.1:

- The *include* element allows you to assemble the contents of a given WSDL 2.0 namespace from several WSDL 2.0 documents that define components for that namespace. The components defined by a given WSDL 2.0 document consist of those whose definitions are contained in the document and those that are defined by any WSDL 2.0 documents that are included in it via the include element. The effect of the include element is cumulative so that if document A includes document B and document B includes document C, then the components defined by document A consist of those whose definitions are contained in documents A, B, and C.
- In contrast, the *import* element does not define any components. Instead, the import element declares that the components whose definitions are contained in a WSDL 2.0 document for a given WSDL 2.0 namespace refer to components that belong to a different WSDL 2.0 namespace. If a WSDL 2.0 document contains definitions of components that refer to other namespaces, then those namespaces must be declared via an import element. The import element also has an optional location attribute that is a hint to the processor where the definitions of the imported namespace can be found. However, the processor may find the definitions by other means, for example, by using a catalog.

The mapping described in this document does not handle directly the *import* and *include* directives. It's the responsibility of the preprocessor inside the tools based on this specification to assemble the various dependencies prior to publishing service definitions into UDDI. After processing any include elements and locating the components that belong to any imported namespaces, the WSDL 2.0 component model for a WSDL 2.0 document will contain a set of components that belong to the document's WSDL 2.0 namespace and any imported namespaces. These components will refer to each other, usually via QName references. A WSDL 2.0 document is invalid if any component reference cannot be resolved, whether or not the referenced component belongs to the same or a different namespace.

3.2.4. wsdl:interface->uddi:tModel

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

The minimum information that must be captured about an interface is its entity type, its local name, its namespace, and the location of the WSDL document that defines the interface.

Capturing the entity type enables users to search for tModels that represents interface artifacts. Capturing the local name, namespace, and WSDL locations enables users to locate the definition of the specified interface artifact.

IF the wsdl:interface extends one or multiple wsdl:interface (new in WSDL 2.0), the tModel MUST contains references to these parent interfaces.

As in the WSDL 1.1 mapping where the *operation* elements of a specified portType were skipped when mapping to UDDI, the *fault*, *operation*, *feature* and *property* elements of a wsdl :interface won't be mapped in UDDI.

The wsdl:interface information is captured as follows :

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

- The tModel MUST contain an overviewDoc with an overviewURL containing the location of the WSDL document that describes the wsdl:interface.
- The tModel MUST contain a categoryBag:
 - the categoryBag MUST contain a keyedReference with a tModelKey of the WSDL Entity Type category system and a keyValue of “interface”.
 - the categoryBag MUST contain a keyedReference with a tModelKey of the XML Namespace category system and a keyValue of the target namespace of the wsdl:description element that contains the wsdl:interface.
 - IF the wsdl:interface contains an *extends* attribute THEN the categoryBag MUST contain a keyedReference with a tModelKey of the WSDL Interface Reference relationship tModel FOR EACH parent wsdl:interface. And the keyValue MUST be the tModelKey of the specified parent interface.

<i>WSDL 2.0</i>	<i>UDDI</i>
Interface	tModel (categorized as an <i>interface</i>)
Namespace of interface	KeyedReference in categoryBag
Local name of interface	TModel name
Location of WSDL document	OverviewURL
Parent interface(s)	KeyedReference(s) in categoryBag

3.2.5. wsdl:binding->uddi:tModel

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

The minimum information that must be captured about an interface is its entity type, its local name, its namespace, the protocol of the binding, and the location of the WSDL document that defines the interface.

Capturing the entity type enables users to search for tModels that represents binding artifacts. Capturing the local name, namespace, and WSDL locations enables users to locate the definition of the specified binding artifact.

IF the wsdl:binding specify an implemented wsdl:interface THEN the tModel MUST contain a reference to the tModel representative of this interface (it was a mandatory indication in WSDL 1.1).

As in the WSDL 1.1 mapping where the *operation* elements of a specified wsdl:binding where skipped when mapping to UDDI, the *fault*, *operation*, *feature* and *property* elements of a wsdl:binding won't be mapped in UDDI.

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 an overviewDoc with an overviewURL containing the location of the WSDL document that describes the wsdl:binding.
- The tModel MUST contain a categoryBag:
 - the categoryBag MUST contain a keyedReference with a tModelKey of the WSDL Entity Type category system and a keyValue of “binding”.

- the categoryBag MUST contain a keyedReference with a tModelKey of the XML Namespace category system and a keyValue of the target namespace of the wsdl:description element that contains the wsdl:binding.
- the categoryBag MUST include a keyedReference with a tModelKey of the Protocol Categorization category system and a keyValue of the tModelKey of the protocol tModel indicated by the *type* attribute of the wsdl:binding.

As specified by par [[WSDLADJ] – 5.2 Identifying the use of the SOAP binding] and [[WSDLADJ] – 6.1 Identifying the use of the HTTP binding], this indication is sufficient to distinguish between various binding types. The client of a web-service will need to check the full WSDL service definition in order to gather enough information to conduct a dialog with the service.

- IF the value of the *type* attribute of the wsdl:binding is “http://www.w3.org/2006/01/wsdl/soap” THEN we will use the tModelKey of the « SOAP Protocol » tModel.

Also, we need to add a keyedReference in the categoryBag with a tModelKey of the « Transport Categorization » canonical tModel and a keyValue of the tModelKey of the tModel standing for the transport type indicated by the *protocol* attribute defined in the SOAP namespace “http://www.w3.org/2003/05/soap-envelope”.

- IF the value of the *type* attribute of the wsdl:binding is “http://www.w3.org/2006/01/wsdl/http” THEN we will use the tModelKey of the “HTTP Protocol” tModel. In this case, there is no need to specify the transport type.
- Other values are handled in a similar fashion. It is assumed that vendors who provide other protocol or transport types will provide the appropriate tModels.
- IF the wsdl:binding has an interface attribute, THEN the categoryBag MUST contain a keyedReference keyedReference with a tModelKey of the WSDL Interface Reference relationship tModel and a keyValue of the tModelKey of the tModel standing for this interface.

<i>WSDL 2.0</i>	<i>UDDI</i>
Binding	tModel (categorized as a <i>binding</i>)
Namespace of binding	KeyedReference(s) in categoryBag
Local name of binding	TModel name
Location of WSDL document	OverviewURL
Protocol of binding	KeyedReference(s) in categoryBag
Transport implemented by this binding (opt)	KeyedReference(s) in categoryBag
Interface linked to the binding (opt)	KeyedReference(s) in categoryBag

3.2.6. wsdl:service->uddi:businessService

A wsdl:service MUST be modeled as a uddi:businessService.

Since a service MUST implement an interface in WSDL 2.0, we can guarantee that there is a direct link between a `wsdl:service` and a given `uddi:businessService` in the mapping (1:1 relation).

The minimum information that must be captured about a `wsdl:service` is its entity type, its local name, its namespace, its implemented interface and the list of endpoints that it supports.

1. Capturing the entity type enables users to search for services that are described by a WSDL definition.
2. Capturing the interface inside the `businessService` enables users to search for the various implementations of a specified interface in a single instruction.
3. The list of ports provide access to the technical information required to consume the service.

The `wsdl:service` information is captured as follows :

- The `uddi:name` element of the `businessService` MUST be the value of the name attribute of the `wsdl:service`.
- The `businessService` MUST contain a `categoryBag`:
 - The `categoryBag` MUST contain a `keyedReference` with a `tModelKey` of the WSDL Entity Type category system and a `keyValue` of “service”.
 - The `categoryBag` MUST contain a `keyedReference` with a `tModelKey` of the XML Namespace category system and a `keyValue` of the target namespace of the `wsdl:description` element that contains the `wsdl:service`.
 - The `categoryBag` MUST contain a `keyedReference` with a `tModelKey` of the WSDL Interface Reference relationship `tModel` and a `keyValue` of the `tModelKey` of the interface `tModel` implemented by this `wsdl:service`.

Each endpoint defined by this service is automatically mapped into a `uddi:bindingTemplate` inside the corresponding `uddi:businessService`. [see 3.2.7 `wsdl:endpoint->uddi:bindingTemplate`]

<i>WSDL 2.0</i>	<i>UDDI</i>
Service	BusinessService
Namespace of service	KeyedReference(s) in categoryBag
Local name of service	BusinessService name
Interface implemented by this service	KeyedReference(s) in categoryBag
Endpoint(s) defined by this service	<i>bindingTemplate</i> inside the <i>bindingTemplates</i> element of the <code>businessService</code> .

3.2.7. `wsdl:endpoint->uddi:bindingTemplate`

A `wsdl:endpoint` MUST be modeled as a `uddi:bindingTemplate` inside the `uddi:businessService` element defining this particular endpoint.

The minimum information that must be captured about an endpoint is its local name, the *binding* that it implements and the *interface* that it implements.

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

The wsdl:endpoint information is captured as follows :

The bindingTemplate MUST contain a tModelInstanceDetails element:

- This tModelInstanceDetails MUST contain a tModelInstanceInfo with a tModelKey of the tModel that models the wsdl:binding that this endpoint implements. The instanceParms of this tModelInstanceInfo MUST contain the wsdl:endpoint local name.
- This tModelInstanceDetails MUST contain a tModelInstanceInfo with a tModelKey of the tModel that models the wsdl:interface implemented by this endpoint. The tModelKey is the value of the mandatory *interface* attribute of the *service* element defining this endpoint (if specified, it can also be obtained by the *interface* attribute of the wsdl:binding associated to this endpoint)

<i>WSDL 2.0</i>	<i>UDDI</i>
Endpoint	BindingTemplate
Namespace of endpoint	Captured in keyedReference of the containing businessService
Local name of endpoint	InstanceParms of the tModelInstanceInfo related to the tModel for the binding.
Binding implemented by endpoint	tModelInstanceInfo with tModelKey of the tModel corresponding to the binding.
Interface implemented by endpoint	tModelInstanceInfo with tModelKey of the tModel corresponding to the interface.

The uddi:bindingTemplate MUST store address information for the Web service in its accessPoint element. In WSDL 2.0 this information can come from various locations whereas it was only located in the extensibility elements of WSDL 1.1.

- IF the endpoint has an *address* attribute:
 - The value of the accessPoint MUST be the value of this attribute
 - The value of the URLType attribute in the accessPoint element MUST correspond to the transport type specified by the binding linked to this endpoint 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.
- IF the endpoint does not have an address attribute BUT possesses an *Endpoint Reference* [WSA 1.0 Core]: in this case, we can use the value of the *address* element of this particular endpoint reference.
- IF the endpoint does not have an *address* attribute NOR *endpoint reference* THEN this case is not specified by the WSDL 2.0 recommendation [WSDL 2.0 Core Rec], it will not be treated by this specification.

3.2.8. WSDL 2.0 Components not mapped to UDDI

The following components are optional in a WSDL description and don't carry any particular semantic annotations. It was decided not to map them to UDDI elements in the WSDL-S to UDDI mapping

current version of this specification. In order to consult them, one must first access to the full WSDL 2.0 definition of a given service stored in the UDDI registry. A direct consequence is that they cannot be used as criterions in a (semantic) UDDI query.

- The “feature” component describes an abstract piece of functionality typically associated with the exchange of messages between communicating parties. The presence of a Feature component in a WSDL 2.0 description indicates that the service supports the feature and may require that a client that interacts with the service use that feature.
- The “property” component in the Features and Properties architecture represents a named runtime value which affects the behavior of some aspect of a Web service interaction, much like an environment variable.
- The optional “documentation” element information item used by WSDL 2.0 as a container for human readable or machine processable documentation. The content of the element information item is arbitrary character information items and element information items and is allowed inside any WSDL 2.0 element information item.
- All the “fault” handling element of WSDL 2.0.

4. WSDL-S to UDDI Mapping

4.1. *WSDL-S version*

Since SAWSDL is in the early stages of the normalization process, we decided to maintain in our binding some elements that were removed from the WSDL-S specification. Like *precondition* and *effect* [see 4.2 Semantic annotations defined by WSDL-S].

4.2. *Semantic annotations defined by WSDL-S*

The current WSDL standard operates at the syntactic level and lacks the semantic expressivity needed to represent the requirements and capabilities of Web Services. Semantics can improve software reuse and discovery, significantly facilitate composition of Web services and enable integrating legacy applications as part of business process integration.

In WSDL-S a semantic information can be associated to a WSDL element by using an annotation based on the standard service description extension mechanism of WSDL 2.0. It is possible to semantically annotate the following WSDL 2.0 components:

- **XML Schema declarations** (simple type, complex type, element), using *modelReference* and *schemaMapping* attributes.
- **Operations**, using the *modelReference* attribute, *precondition* and *effect* elements.
- **Interfaces**, using the *category* element. This one differs from the other elements since it is used to reference taxonomy concepts not ontologies.

4.3. *Semantic information that will be mapped to UDDI*

Compared to the mapping described by “Adding Semantics to Web Services Standards”, the approach adopted by this specification should allow greater expressibility when composing UDDI request using semantic criterions.

In this early paper by the LSDID Lab, not all the semantic annotations currently available were mapped to UDDI structures. In particular, we add the mapping for the precondition and effect elements as defined by the current WSDL-S specification.

The semantic information attached to an operation input and output will also be mapped into UDDI. This will allow richer queries to be made and should significantly improve the performance when searching for a particular web-service in a UDDI registry (i.e. it will not be necessary to gather the full wsdl description of a service stored in UDDI in order to match and rank it based on the semantic similarity with the query being made).

The following semantic elements will be mapped to UDDI :

- Operation inputs
- Operation outputs
- Operation preconditions
- Operation effects
- Operation functional concept
- Interface categorization information

4.4. WSDL 2.0 to UDDI mapping extensions

Semantic annotations of WSDL-S are limited to the “abstract” part of a service declaration. As such, extensions will only be made on the operation and interface mapping specifications. Concerning the service implementation part, we will use the previously defined mapping [see 3. WSDL 2.0 to UDDI mapping].

Since WSDL-S stores input/output related semantic information in the XML Schema types declarations we will also need to extract the needed ontological references from this structures in order to map them into UDDI.

<i>WSDL 2.0 Element</i>	<i>Extension</i>
Operation	Was not mapped in the WSDL 2.0 to UDDI mapping. In WSDL-S they are annotated with semantic elements so we need to map them to UDDI. This mapping will convert them to tModels.
Interface	The mapping that was specified in 3.2.4 need to be modified in order to create a link between a given interface and the operations that it defines.
Type	The “message” construct was removed from WSDL 2.0. As such, the semantic information that was added to the message elements is now directly appended to the xsd types themselves. We need to build a mapping to store this semantic elements inside the operation tModels.

4.5. Mapping

Figure 3 shows a synthetic view of the extensions that were above-mentioned. The following sections will explain in details the implementation of these mappings.

The semantic additions made by WSDL-S/SAWSDL to the WSDL 2.0 model and their mapping to UDDI are shown in blue.

The logical links between interfaces and operations, operations and types, are figured in red.

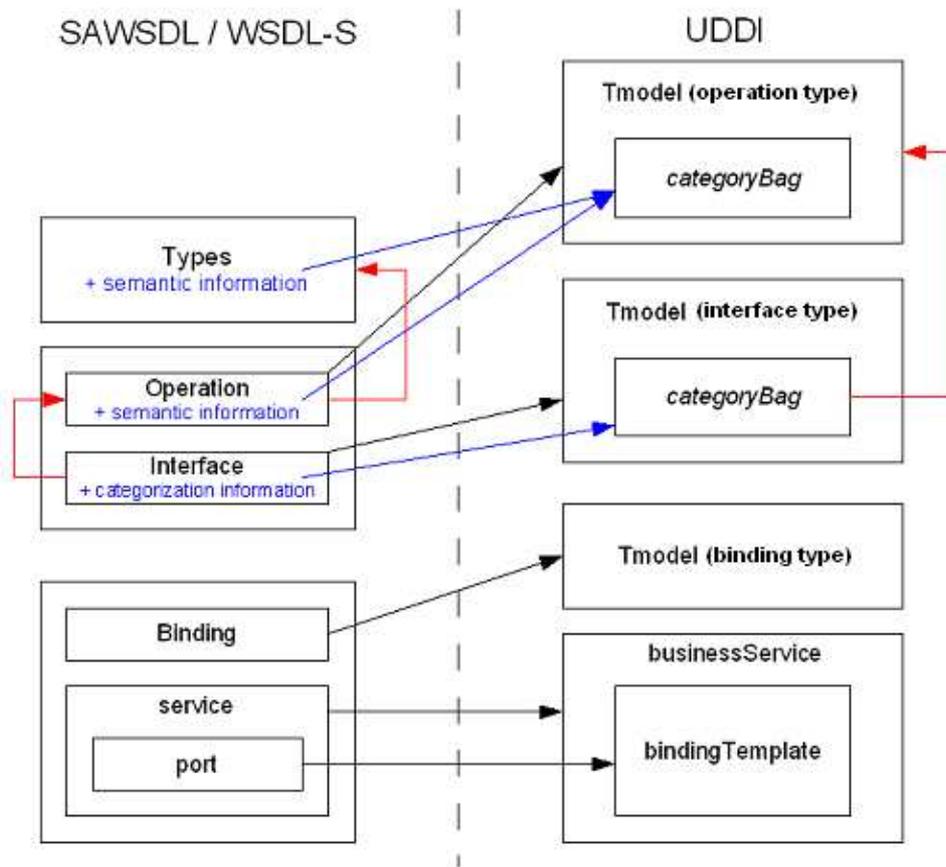


Figure 3 - WSDL 2.0 to UDDI mapping extensions

4.5.1. New Canonical tModels

This mapping introduces a number of canonical tModels that are used to represent WSDL-S/SAWSDL 2.0 semantic metadata and relationships. These tModels MUST be registered in the UDDI registry to support this mapping. Only the V1/V2 keys are given for these tModels.

We reuse all the canonical tModels that were defined by [3.2.1 New Canonical tModels] to which we add all the following tModels [see Appendix A: Canonical tModels for more details] :

- WSDL Operation Reference
- Functional Concept
- Input
- Output
- Precondition
- Effect

4.5.2. wsdl-s:category-> uddi:tModel (interface type) [extension]

We base this mapping on the one specified by [3.2.4 wsdl:interface->uddi:tModel] to which we add the categorization information specified by the *category* extension element. This information consist of references to taxonomies elements, Users can choose any

WSDL-S to UDDI mapping

Copyright © Thales Group 2006. All Rights Reserved

categorization of their choice such as NAICS, UNSPSC and GICS. This aids in service discovery by narrowing the range of candidate services.

Multiple *category* elements can be used to specify that the service falls into multiple categories. A *category* element specifies one categorization. As such we need to map each and every *category* element individually.

The information associated to a given `wsdl-s:category` element is added to the `categoryBag` of the associated interface `tModel` as such:

- For each `wsdl-s:category` element defined in a `wsdl:interface`:
 - IF the value of the *taxonomyURI* attribute of the `wsdl-s:category` element is an URI associated in the UDDI registry to a valid taxonomy `tModel` (NAICS and UNSPSC `tModels` are bundled with `juddi`)

THEN the `categoryBag` MUST contain a `keyedReference` with a `tModelKey` of the corresponding taxonomy `tModel` and the value of the *taxonomyCode* attribute of the `wsdl-s:category` element as the `keyValue`
 - ELSE IF the value of the value of the *taxonomyCode* attribute of the `wsdl-s:category` element is unknown in the UDDI registry

THEN we must register a new taxonomy `tModel` to the UDDI registry PRIOR TO the mapping of this `wsdl-s:category` element.

<i>WSDL 2.0</i>	<i>UDDI</i>
Interface	Tmodel (categorized as an <i>interface</i>)
Namespace of interface	<code>KeyedReference</code> in <code>categoryBag</code>
Local name of interface	<code>TModel</code> name
Location of WSDL document	<code>OverviewURL</code>
Parent interface(s)	<code>KeyedReference(s)</code> in <code>categoryBag</code>
Interface category(s)	<code>KeyedReference(s)</code> in <code>categoryBag</code>

4.5.3. `wsdl:interface->uddi:tModel` [extension]

We base this mapping on the one specified by [4.5.2 `wsdl-s:category-> uddi:tModel` (interface type) [extension]] to which we add the ability to link an interface to the operation it defines.

We do not add to the mapping of a given interface the operations that may have been defined by extended interfaces (as allowed by the new WSDL 2.0 specification), they will be mapped when these interfaces will be analyzed. It is possible to indirectly access these operations using the “WSDL Interface Reference” `keyedReferences` defined by [3.2.4 `wsdl:interface->uddi:tModel`].

We add the following elements to the `categoryBag` of the `tModel` modeling a given interface:

- For each `wsdl:operation` defined by this `wsdl:interface`:
 - The `categoryBag` MUST contain a `keyedReference` with a `tModelKey` of the WSDL Operation Reference relationship `tModel` and a `keyValue` of the

tModelKey of the operation tModel.

<i>WSDL 2.0</i>	<i>UDDI</i>
Interface	tModel (categorized as an <i>interface</i>)
Namespace of interface	KeyedReference in categoryBag
Local name of interface	tModel name
Location of WSDL document	OverviewURL
Parent interface(s)	KeyedReference(s) in categoryBag
Interface category(s)	KeyedReference(s) in categoryBag
Interface defined operation(s)	KeyedReference(s) in categoryBag

4.5.4. wsdl:operation->uddi:tModel

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

The minimum information that must be captured about an operation is its entity type, its local name, its namespace, the location of the WSDL document that defines the interface and the interface that defines this operation. To which we add the following semantic information:

- If specified, the functional concept of the operation(also known as “action”).
- If specified, the bottom-level ontological concepts (see [SAWSDL] section 2.2.1 for more details) associated with the types of this operation’s inputs.
- If specified, the bottom-level ontological concepts associated with the types of this operation’s outputs.
- If a precondition is specified for this operation, the ontological concept associated to this precondition.
- If an effect is specified for this operation, the ontological concept associated to this effect. If a functional concept is specified for this operation then it must specify an effect.

Capturing the entity type enables users to search for tModels that represents operation artifacts. Capturing the local name, namespace, and WSDL locations enables users to locate the definition of the specified binding artifact. Capturing the semantic information enables users to search for a service based on ontological concepts associated to its operations.

The wsdl:operation information is captured as follows :

- The uddi:name element of the tModel MUST be the value of the name attribute of the wsdl:operation.
- The tModel MUST contain an overviewDoc with an overviewURL containing the location of the WSDL document that describes the wsdl:operation.
- The tModel MUST contain a categoryBag:
 - The categoryBag MUST contain a keyedReference with a tModelKey of the WSDL Entity Type category system and a keyValue of “operation”.
 - The categoryBag MUST contain a keyedReference with a tModelKey of the XML Namespace category system and a keyValue of the target

namespace of the wsdl:description element that contains the wsdl:operation.

- The categoryBag MUST contain a keyedReference with a tModelKey of the WSDL Interface Reference relationship tModel and a keyValue of the tModelKey of the interface tModel implementing this operation.
- IF the wsdl:operation has a wsdl-s:modelReference attribute THEN the categoryBag MUST contain a keyedReference with a tModelKey of the Functional Concept category system and the value of the wsdl-s:modelReference attribute as a keyValue. The SAWSDL technical note specify that this value is an URI.
- IF the wsdl:operation has a wsdl-s:precondition element THEN the categoryBag MUST contain a keyedReference with a tModelKey of the Precondition category system and, as keyValue, the value of the *modelReference* attribute of the wsdl-s:precondition element. The WSDL-S last proposal specify that this value is an URI.
- FOR EACH wsdl-s:effect element of the wsdl:operation the categoryBag MUST contain a keyedReference with a tModelKey of the Effect category system and, as keyValue, the value of the *modelReference* attribute of the wsdl-s:effect element. The WSDL-S last proposal specify that this value is an URI.
- FOR EACH wsdl:input element of the wsdl:operation :
 - IF this input has an *element* attribute,
 - AND IF this attribute is referencing a XML Schema type,
 - AND IF this type has a bottom-level wsdl-s:modelReference attribute,
 - THEN the categoryBag MUST contain a keyedReference with a tModelKey of the Input category system and, as keyValue, the value of the *modelReference* attribute of the type element. The WSDL-S last proposal specify that this value is an URI.
- FOR EACH wsdl:output element of the wsdl:operation :
 - IF this output has an *element* attribute,
 - AND IF this attribute is referencing a XML Schema type,
 - AND IF this type has a bottom-level wsdl-s:modelReference attribute,
 - THEN the categoryBag MUST contain a keyedReference with a tModelKey of the Output category system and, as keyValue, the value of the *modelReference* attribute of the type element. The WSDL-S last proposal specify that this value is an URI.

<i>WSDL 2.0</i>	<i>UDDI</i>
Operation	tModel (categorized as an <i>operation</i>)
Namespace of operation	KeyedReference in categoryBag
Local name of operation	TModel name

<i>WSDL 2.0</i>	<i>UDDI</i>
Location of WSDL document	OverviewURL
Functional Concept	KeyedReference in categoryBag
Input(s)	KeyedReference(s) in categoryBag
Output(s)	KeyedReference(s) in categoryBag
Precondition	KeyedReference in categoryBag
Effect(s)	KeyedReference(s) in categoryBag

References

- [UDDIMAP] - OASIS - Using WSDL in a UDDI Registry, Version 2.0.2 - Technical Note - <http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v202-20040631.htm>
- [WSDLADJ] - W3C - Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts - <http://www.w3.org/TR/2006/CR-wsdl20-adjuncts-20060327>
- [WSA 1.0 Core] - W3C - Web Services Addressing 1.0 - Core - <http://www.w3.org/TR/2005/CR-ws-addr-core-20050817>
- [WSDL 2.0 Core Rec] - W3C - Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language - <http://www.w3.org/TR/2006/CR-wsdl20-20060327>
- [SAWSDL] - W3C – Semantic Annotations for WSDL - <http://www.w3.org/2002/ws/sawSDL/spec/>

5. Appendix A: Canonical tModels

5.1. WSDL Interface Reference

5.1.1. 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:interface. 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 Interface Reference relationship tModel provides a mechanism to indicate that a UDDI entity has a relationship with a certain wsdl:interface tModel. This can be applied, for example, to indicate that a wsdl:binding tModel is a binding of a specific wsdl:interface tModel.

5.1.2. Definition

Name: thalesgroup-com:sc2:wsdl:interfaceReference
Description: A relationship tModel used to reference a wsdl:interface tModel
V1,V2 format key: uuid:9FC8E760-F7C6-11DA-A760-E48CBDDDD8628
Categorization: relationship
Checked: no

5.1.3. V2 tModel Structure

```
<tModel tModelKey="uuid:9FC8E760-F7C6-11DA-A760-E48CBDDDD8628">
  <name> thalesgroup-com:sc2:wsdl:interfaceReference </name>
  <description xml:lang="en">A relationship tModel used to reference
a wsdl:interface tModel</description>
  <overviewDoc>
    <overviewURL>
      http://www.thalesgroup.com/sc2/wsdl-
s_mapping.htm#interfaceReference
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="uuid:c1acf26d-9672-4404-9d70-
39b756e62ab4"
keyName="uddi-org:types" keyValue="relationship"/>
    <keyedReference
tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
keyName="uddi-org:types" keyValue="unchecked"/>
  </categoryBag>
</tModel>
```

5.1.4. Valid values

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

5.1.5. Example of Use

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

```

<categoryBag>
  <keyedReference
    tModelKey="uuid:9FC8E760-F7C6-11DA-A760-E48CBDDD8628"
    keyName="wsdl:interface 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.

5.2. WSDL Operation Reference

5.2.1. Design Goals

WSDL Entities exhibit many relationships. Specifically, a wsdl:interface describes multiple wsdl:operation. This same relationship 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 Operation Reference relationship tModel provides a mechanism to indicate that a UDDI entity has a relationship with a certain wsdl:operation tModel.

5.2.2. Definition

Name: thalesgroup-com:sc2:wsdl:operationReference
Description: A relationship tModel used to reference a wsdl:operation tModel
V1,V2 format key: uuid:F7893D50-F7C7-11DA-BD50-F1F106FBFEDE
Categorization: relationship
Checked: no

5.2.3. V2 tModel Structure

```

<tModel tModelKey="uuid:F7893D50-F7C7-11DA-BD50-F1F106FBFEDE">
  <name> thalesgroup-com:sc2:wsdl:operationReference </name>
  <description xml:lang="en">A relationship tModel used to reference
a wsdl:operation tModel</description>
  <overviewDoc>
    <overviewURL>
      http://www.thalesgroup.com/sc2/wsdl-
s_mapping.htm#operationReference
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="uuid:c1acf26d-9672-4404-9d70-
39b756e62ab4"
    keyName="uddi-org:types" keyValue="relationship"/>
    <keyedReference
tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
    keyName="uddi-org:types" keyValue="unchecked"/>
  </categoryBag>
</tModel>

```

5.2.4. Valid values

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

5.2.5. Example of Use

```
<categoryBag>
  <keyedReference
    tModelKey="uuid:F7893D50-F7C7-11DA-BD50-F1F106FBFEDE"
    keyName="wsdl:operation Reference"
    keyValue="uuid:e8cf1163-8234-4b35-865f-94a7322e40d4"/>
  ...
</categoryBag>
```

5.3. *Functionnal Concept*

5.3.1. Design Goals

As needed by 4.5.4, this tModel is used to express a relation between an operation and a functional concept from a specific ontology. It is to be used with operation tModels.

5.3.2. Definition

Name: thalesgroup-com:sc2:wsdl:functionnalConcept
Description: A category system used to associate a concept from a specific semantic domain to an operation tModel.
V1,V2 format key: uuid:15D03F20-F7C8-11DA-BF20-C3F48481A023
Categorization: categorization
Checked: no

5.3.3. V2 tModel Structure

```
<tModel tModelKey="uuid:15D03F20-F7C8-11DA-BF20-C3F48481A023">
  <name> thalesgroup-com:sc2:wsdl:functionnalConcept </name>
  <description xml:lang="en"> A category system used to associate a
  concept from a specific semantic domain to an operation tModel
</description>
  <overviewDoc>
    <overviewURL>
      http://www.thalesgroup.com/sc2/wsdl-
      s_mapping.htm#functionnalConcept
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="uuid:c1acf26d-9672-4404-9d70-
    39b756e62ab4"
    keyName="uddi-org:types" keyValue="categorization"/>
    <keyedReference
      tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
      keyName="uddi-org:types" keyValue="unchecked"/>
  </categoryBag>
</tModel>
```

5.3.4. Valid values

Valid values for this category system are URIs. The content of the `keyValue` attribute in a `keyedReference` that refers to this `tModel` is the value of the `modelReference` attribute of the `wsdl-s:modelReference` element of this specific operation.

5.3.5. Example of Use

```
<categoryBag>
  keyedReference
    tModelKey="uuid:15D03F20-F7C8-11DA-BF20-C3F48481A023"
    keyName="Functional Concept"
    keyValue="http://example.org/rosetta#RequestPurchaseOrder"
  ...
</categoryBag>
```

5.4. Input

5.4.1. Design Goals

As needed by 4.5.4, this `tModel` is used to express a relation between an operation's input element and a concept from a specific ontology. It is to be used with operation `tModels`.

5.4.2. Definition

Name: thalesgroup-com:sc2:wsdl:input
Description: A category system used to associate an input related concept from a specific semantic domain to an operation `tModel`.
V1,V2 format key: uuid:349CC4A0-F7C8-11DA-84A0-90AE920025E6
Categorization: categorization
Checked: no

5.4.3. V2 tModel Structure

```
<tModel tModelKey="uuid:349CC4A0-F7C8-11DA-84A0-90AE920025E6">
  <name> thalesgroup-com:sc2:wsdl:input </name>
  <description xml:lang="en">A category system used to associate an
input related concept from a specific semantic domain to an operation
tModel</description>
  <overviewDoc>
    <overviewURL>
      http://www.thalesgroup.com/sc2/wsdl-s_mapping.htm#input
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="uuid:c1acf26d-9672-4404-9d70-
39b756e62ab4"
    keyName="uddi-org:types" keyValue="categorization"/>
    <keyedReference
      tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
      keyName="uddi-org:types" keyValue="unchecked"/>
  </categoryBag>
</tModel>
```

5.4.4. Valid values

Valid values for this category system are URIs. The content of the keyName attribute in a keyedReference that refers to this tModel is the value of the *modelReference* attribute of the type system element associated to this specific input.

5.4.5. Example of Use

```
<categoryBag>
  keyedReference
    tModelKey="uuid:349CC4A0-F7C8-11DA-84A0-90AE920025E6"
    keyName="Input"
    keyValue="http://example.org/rosetta#PurchaseOrderRequest"
  ...
</categoryBag>
```

5.5. Output

5.5.1. Design Goals

As needed by 4.5.4, this tModel is used to express a relation between an operation's output element and a concept from a specific ontology. It is to be used with operation tModels.

5.5.2. Definition

Name: thalesgroup-com:sc2:wSDL:output
Description: A category system used to associate an output related concept from a specific semantic domain to an operation tModel.
V1,V2 format key: uuid:72CBF520-F7C8-11DA-B520-E08563B732CC
Categorization: categorization
Checked: no

5.5.3. V2 tModel Structure

```
<tModel tModelKey="uuid:72CBF520-F7C8-11DA-B520-E08563B732CC">
  <name> thalesgroup-com:sc2:wSDL:output </name>
  <description xml:lang="en">A category system used to associate an
  output related concept from a specific semantic domain to an operation
  tModel</description>
  <overviewDoc>
    <overviewURL>
      http://www.thalesgroup.com/sc2/wSDL-s_mapping.htm#output
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="uuid:c1acf26d-9672-4404-9d70-
    39b756e62ab4"
    keyName="uddi-org:types" keyValue="categorization"/>
    <keyedReference
    tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
    keyName="uddi-org:types" keyValue="unchecked"/>
  </categoryBag>
</tModel>
```

5.5.4. Valid values

Valid values for this category system are URIs. The content of the keyName attribute in a keyedReference that refers to this tModel is the value of the *modelReference* attribute of the type system element associated to this specific output.

5.5.5. Example of Use

```
<categoryBag>
  keyedReference
    tModelKey="uuid:72CBF520-F7C8-11DA-B520-E08563B732CC"
    keyName="Output"
    keyValue="http://example.org/rosetta#PurchaseConfirmation"
  ...
</categoryBag>
```

5.6. Precondition

5.6.1. Design Goals

As needed by 4.5.4, this tModel is used to express a relation between an operation's precondition element and a concept from a specific ontology. It is to be used with operation tModels.

5.6.2. Definition

Name: thalesgroup-com:sc2:wSDL:precondition
Description: A category system used to associate a precondition related concept from a specific semantic domain to an operation tModel.
V1,V2 format key: uuid:A05DC270-F7C8-11DA-8270-ABAD25871E16
Categorization: categorization
Checked: no

5.6.3. V2 tModel Structure

```
<tModel tModelKey="uuid:A05DC270-F7C8-11DA-8270-ABAD25871E16">
  <name> thalesgroup-com:sc2:wSDL:precondition </name>
  <description xml:lang="en">A category system used to associate a
  precondition related concept from a specific semantic domain to an
  operation tModel</description>
  <overviewDoc>
    <overviewURL>
      http://www.thalesgroup.com/sc2/wSDL-
      s_mapping.htm#precondition
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="uuid:c1acf26d-9672-4404-9d70-
    39b756e62ab4"
    keyName="uddi-org:types" keyValue="categorization"/>
    <keyedReference
    tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
    keyName="uddi-org:types" keyValue="unchecked"/>
  </categoryBag>
</tModel>
```

5.6.4. Valid values

Valid values for this category system are URIs. The content of the keyName attribute in a keyedReference that refers to this tModel is the value of the *modelReference* attribute of the precondition element defined by a specific operation.

5.6.5. Example of Use

```
<categoryBag>
  keyedReference
    tModelKey="uuid:A05DC270-F7C8-11DA-8270-ABAD25871E16"
    keyName="Precondition"
    keyValue="http://example.org/rosetta#ValidCreditCard"
  ...
</categoryBag>
```

5.7. Effect

5.7.1. Design Goals

As needed by 4.5.4, this tModel is used to express a relation between an operation's effect element and a concept from a specific ontology. It is to be used with operation tModels.

5.7.2. Definition

Name: thalesgroup-com:sc2:wSDL:effect
Description: A category system used to associate an effect related concept from a specific semantic domain to an operation tModel.
V1,V2 format key: uuid:D87CD330-F7C8-11DA-9330-E5D46D2020A1
Categorization: categorization
Checked: no

5.7.3. V2 tModel Structure

```
<tModel tModelKey="uuid:D87CD330-F7C8-11DA-9330-E5D46D2020A1">
  <name> thalesgroup-com:sc2:wSDL:effect </name>
  <description xml:lang="en">A category system used to associate an
effect related concept from a specific semantic domain to an operation
tModel</description>
  <overviewDoc>
    <overviewURL>
      http://www.thalesgroup.com/sc2/wSDL-s_mapping.htm#effect
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="uuid:c1acf26d-9672-4404-9d70-
39b756e62ab4"
    keyName="uddi-org:types" keyValue="categorization"/>
    <keyedReference
tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
keyName="uddi-org:types" keyValue="unchecked"/>
  </categoryBag>
</tModel>
```

5.7.4. Valid values

Valid values for this category system are URIs. The content of the keyValue attribute in a keyedReference that refers to this tModel is the value of the *modelReference* attribute of the effect element defined by a specific operation.

5.7.5. Example of Use

```
<categoryBag>
  keyedReference
    tModelKey="uuid:D87CD330-F7C8-11DA-9330-E5D46D2020A1"
    keyName="Effect "
    keyValue="http://example.org/rosetta#AccountDebited"
  ...
</categoryBag>
```

6. Appendix B: Comparison between WSDL 1.1 and WSDL 2.2 component models

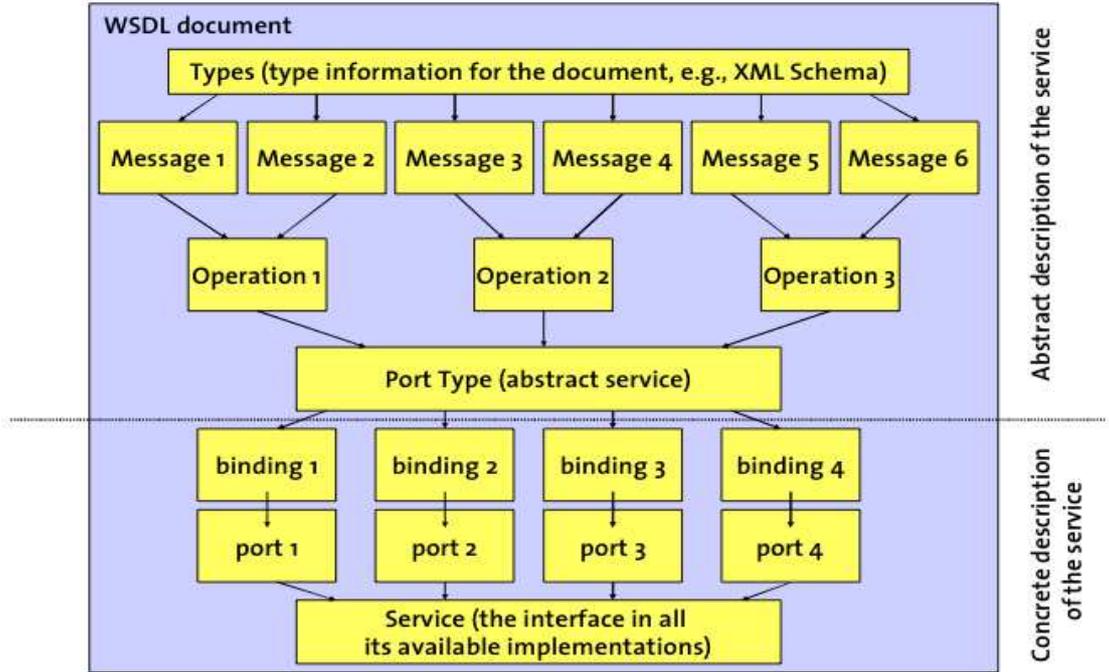


Figure 4 - WSDL 1.1

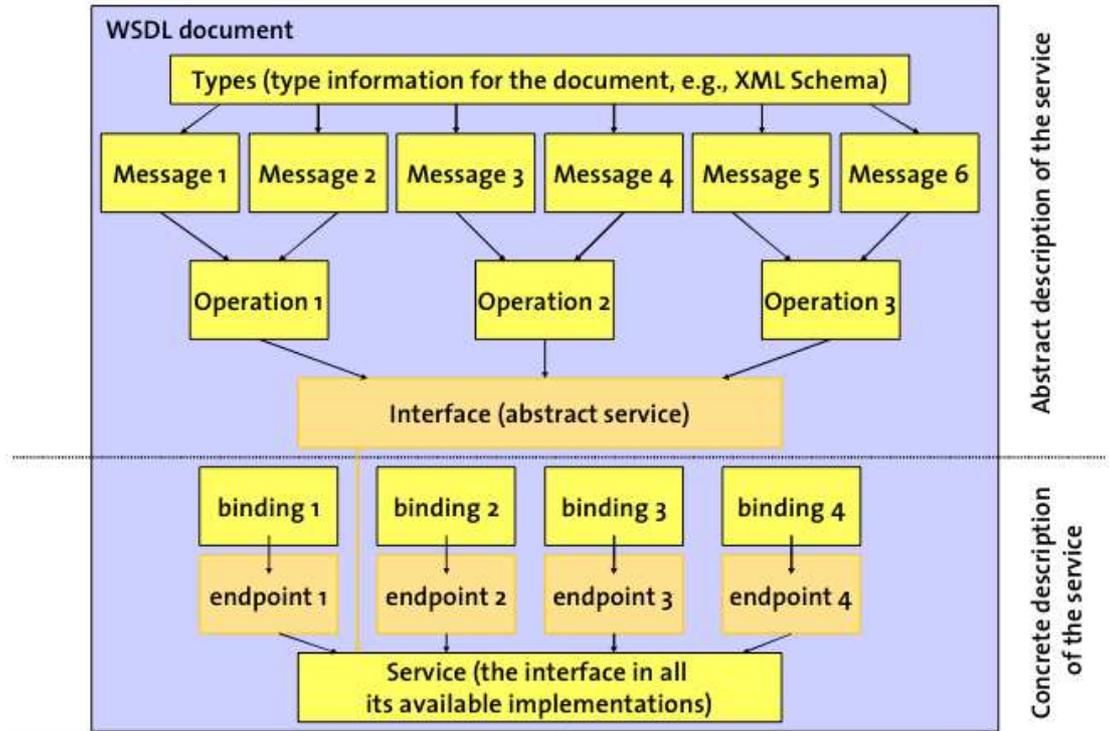


Figure 5 - WSDL 2.0

Errata: this diagram comes from an old WSDL 2.0 specification. The message component was removed from the last specification.

7. Appendix C: WSDL 2.0 Components hierarchical view

