UDDI Spec TC

# Technical Note

## Using BPEL4WS in a UDDI registry

**Document identifier:**
uddi-spec-tc-tn-bpel-20040415.doc

**Location:**
http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-bpel-20040415.doc

**Authors:**
Claus von Riegen, SAP (claus.von.riegen@sap.com)
Ivana Trickovic, SAP (ivana.trickovic@sap.com)

**Contributors:**
Tom Bellwood, IBM (bellwood@us.ibm.com)

**Abstract:**

BPEL4WS abstract processes describe the observable behavior of Web services. They complement abstract WSDL interfaces (port types and operations) and the UDDI model by defining dependencies between service operations in the context of a message exchange. This technical note describes the relationships between the three models and suggests how BPEL4WS abstract processes can be used in a UDDI Registry.

**Status:**

This document is updated periodically on no particular schedule. Send comments to the editor.

Committee members should send comments on this technical note 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.

For information on whether any patents have been disclosed that may be essential to implementing this technical note, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the UDDI Spec TC web page (http://www.oasis-open.org/committees/uddi-spec/).

# Table of Contents

# 1 Introduction

## 1.1 Problem statement

Publishing and discovering individual Web services is probably the area UDDI is most often used for. Also, the question on how to do that, especially by using WSDL **[WSDL11]**, is already addressed by a number of Best Practice documents (**[WSDLBP]**, **[WSDLTN]**).

WSDL describes the static interface of Web services, which includes definitions of individual operations. This may be adequate for Web services participating in stateless message exchanges. For Web services, which participate in longer conversations, it is necessary to describe the behavior of the services in terms of dependencies, either logical or temporal, among exchanged messages. This is the focus of several efforts including **[BPEL4WS]**, now under standardization by the OASIS WSBPEL TC.

BPEL4WS abstract processes complement abstract WSDL interfaces describing behavioral aspects of Web services and providing data needed for integration with business partners. Abstract processes are used to specify the order in which business partners may invoke operations. Therefore it may be also of interest to exchange abstract processes between business partners. Software companies and standards bodies may use a UDDI registry to publish different types of services and business users may populate the registry with descriptions of services they support. BPEL4WS and WSDL may be used to describe service types, protocols that are supported and other deployment details.

While it is certainly possible to publish BPEL4WS process definitions in a UDDI registry, no guidelines are available as of today, which specify a common approach for doing that. Without such a common approach, the certainty that users find BPEL4WS process definitions or Web services that implement a given part of such a definition is limited.

This technical note provides guidelines for publishing BPEL4WS abstract processes in UDDI. The primary goals of mapping BPEL4WS artifacts to the UDDI model are to:

1. Enable the automatic registration of BPEL4WS definitions in UDDI
2. Enable optimized and flexible UDDI queries based on specific BPEL4WS artifacts and metadata
3. Provide composability with the mapping described in the *Using WSDL in a UDDI Registry, Version 2* **[WSDLTN]** Technical Note document

The following types of queries are enabled by this technical note:

- Given the namespace and/or local name of a bpws:process, find the tModel that represents that process.
- Given a tModel that represents a wsdl:portType (based on the usage of **[WSDLTN]**), find all tModels that represent bpws:processes based on that wsdl:portType.
- Given a tModel representing a bpws:process, find all tModels representing wsdl:portTypes that are used by the bpws:process.
- Given a tModel representing a bpws:process, find all bindingTemplates that implement a wsdl:portType that in turn is part of the bpws:process.

## 1.2 Reliance on WSDL Technical Note

Since BPEL4WS abstract processes operate on WSDL artifacts, a common approach for mapping WSDL artifacts to the UDDI model is a prerequisite for this technical note in general. In particular, WSDL port types need to be registered and identified individually in UDDI. Thus, this technical note assumes the application of the Technical Note for Using WSDL in a UDDI Registry, Version 2.0 **[WSDLTN]**.

## 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 Technical Note Solution

## 2.1 Definitions

This section briefly explains a sub-set of BPEL4WS features that is of interest to this technical note and concepts of the mapping of BPEL4WS into UDDI.

### 2.1.1 BPEL4WS Data Model

The BPEL4WS model supports definition of the observable behavior of a Web service participating in a long-running conversation with other Web services. More particularly, the model defines abstract processes, which may be used for describing the observable behavior. These processes are in the scope of this technical note. BPEL4WS introduces features, such as process, action, correlation, role, partner link, etc, needed to describe the behavioral aspects of Web services. Figure 1 shows a sub-set of those features of interest in the context of this note and relationships between them. A process defines sequencing of operations supported by a Web service.

A Web service may play multiple roles within a conversation. Usually, for each partner the Web service may expose a different role. The abstract process declares roles that the Web service provider implements and roles that its partners must implement in order to make conversations possible in accordance to the described abstract process.

BPEL4WS partner link type defines binary relationship between roles. It specifies at most two roles that may communicate.

The BPEL4WS model is built on top of the abstract part of WSDL, which includes definitions of port types, messages and data types. Therefore, a BPEL4WS abstract process definition is reusable, that is, different services may implement the same BPEL4WS abstract process. The BPEL4WS process definition relies on WSDL operations. Each role defined in the partner link type specifies exactly one WSDL port type it implements.

A single BPEL4WS document may include multiple abstract process definitions. However, they are uniquely identified by the target namespace and its local name.

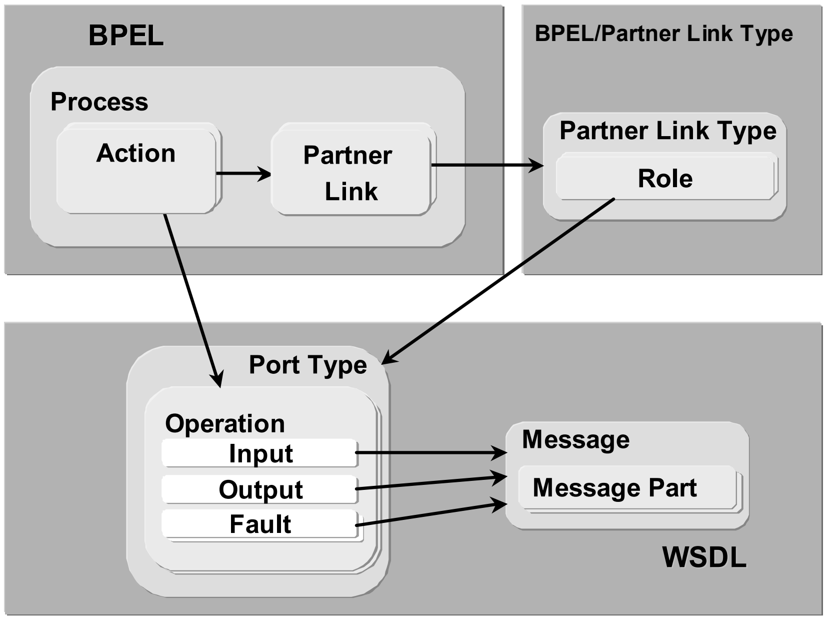

Figure 1: The BPEL model and its relationship with WSDL

### 2.1.2 Mapping BPEL4WS to UDDI

BPEL4WS abstract processes are published as separate UDDI tModels. They are named with the BPEL4WS process name. They are categorized as BPEL4WS process definitions, using a category system defined in this technical note. Their overviewDoc references an external BPEL4WS document that contains the process definition.

All WSDL portTypes that are used in the BPEL4WS process definition (via the referenced BPEL4WS partnerLinkTypes) are published as portType tModels according to **[WSDLTN]**.

The process tModel references all such WSDL portType tModels, using a separate portType reference tModel, defined in this technical note. Note that it is a characteristic of the BPEL4WS process that it defines a conversation based on WSDL portTypes. Thus, the relationship between process tModel and portType tModel is to be published by the process tModel publisher, not by the portType tModel publisher, which may be a different person.

Implementations of those WSDL portTypes that are used in a BPEL4WS process are published as a UDDI bindingTemplate and reference, additionally to the corresponding WSDL portType tModel, the process tModel that represents the BPEL4WS process. Note that it is a characteristic of a deployed Web service that it behaves as described in a particular BPEL4WS process. Thus, the relationship between bindingTemplate and process tModel is to be published by the bindingTemplate publisher, not by the process tModel publisher, which may be a different person.

An overview of this mapping approach is illustrated by Figure 2.

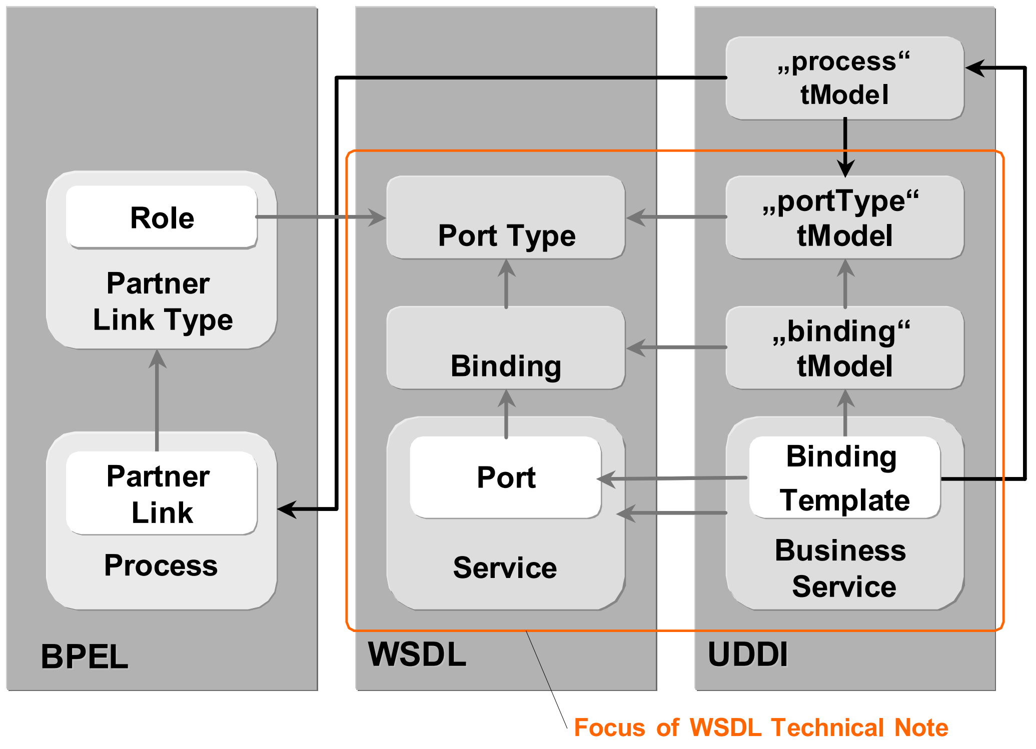

Figure 2: Mapping BPEL to UDDI

# 3 tModel definitions

## 3.1 BPEL Entity Type tModel

### 3.1.1 Design Goals

This mapping uses a number of UDDI entities to represent the various entities within a BPEL4WS document. A mechanism is required to indicate what type of BPEL4WS entity is being described by each UDDI entity. The BPEL Entity Type tModel provides a typing system for this purpose. This category system is used to indicate that a UDDI entity represents a particular type of BPEL4WS entity.

### 3.1.2 Definition

| | |
|---|---|
| **Name**: | uddi.org:bpel:types |
| **Description**: | BPEL Type Category System |
| **V3 format key**: | uddi:uddi.org:bpel:types |
| **V1,V2 format key**: | uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f |
| **Categorization**: | categorization |
| **Checked**: | no |

#### 3.1.2.1 V2 tModel Structure

```
<tModel tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f">
    <name>uddi.org:bpel:types</name>
    <overviewDoc>
        <overviewURL>
              TBD, should point to this section when the document is
published as a Technical Note by the UDDI TC
        </overviewURL>
    </overviewDoc>
    <categoryBag>
        <keyedReference
            keyName="uddi-org:categorization:types"
            keyValue="categorization"
            tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"/>
        <keyedReference
            keyName="uddi-org:categorization:types"
            keyValue="unchecked"
            tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"/>
    </categoryBag>
</tModel>
```

#### 3.1.2.2 Valid Values

While this is an unchecked category system, there is only one value that should be used with this category system:

| keyValue | Description | UDDI Entity |
|---|---|---|

| | | |
|---|---|---|
| process | Represents a UDDI entity categorized as a bpel:process | tModel |

#### 3.1.2.3 Example of Use

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

```
<categoryBag>
         <keyedReference
                tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"
                keyName="BPEL Entity type"
                keyValue="process"/>
…
</categoryBag>
```

## 3.2 WSDL portType Reference tModel

### 3.2.1 Design Goals

BPEL4WS process definitions reference, through related partnerLinkType definitions, WSDL portTypes that describe the interfaces the given process definition is based upon.

The WSDL portType Reference category system provides a mechanism to indicate that a bpel:process tModel is based on a specific wsdl:portType tModel.

### 3.2.2 Definition

| | |
|---|---|
| **Name**: | uddi.org:bpel:wsdlPortTypeReference |
| **Description**: | A category system used to reference a wsdl:portType tModel |
| **V3 format key**: | uddi:uddi.org:bpel:wsdlporttypereference |
| **V1,V2 format key**: | uuid:ef2dcc0a-edc8-343d-9913-d2e61777a90c |
| **Categorization:** | categorization |
| **Checked:** | yes |

#### 3.2.2.1 V2 tModel Structure

```
<tModel tModelKey="uuid:ef2dcc0a-edc8-343d-9913-d2e61777a90c">
    <name>uddi.org:bpel:wsdlPortTypeReference</name>
    <description xml:lang="en">
        This tModel is a category system tModel that can be used to
identify a relationship to a portType tModel.
    </description>
    <overviewDoc>
        <overviewURL>
            TBD
        </overviewURL>
    </overviewDoc>
    <categoryBag>
        <keyedReference
            keyName="uddi-org:categorization:types"
            keyValue="categorization"
            tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"/>
        <keyedReference
            keyName="uddi-org:categorization:types"
            keyValue="checked"
            tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"/>
        <keyedReference
            keyName="uddi-org:categorization:entityKeyValues"
            keyValue="tModel"
            tModelKey="uuid:916b87bf-0756-3919-8eae-97dfa325e5a4"/>
    </categoryBag>
</tModel>
```

### 3.2.3 Valid Values

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

### 3.2.4 Example of Use

One would add the following keyedReference to signify that a bpws:process is based upon a specific portType:

```
<categoryBag>
    <keyedReference
        tModelKey="uuid:ef2dcc0a-edc8-343d-9913-d2e61777a90c"
        keyName="uddi-org:bpel: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 WSDL portType.

# 4 Example

This section includes tModels representing a BPEL4WS abstract process, accompanying WSDL descriptions and UDDI registrations. A Travel Agent example is used for illustration. The example gives the basic behavior exposed by a Travel Agent service in a Ticket Reservation System. Figure 3 shows the overall process: the Travel Agent interacts with a Customer (a traveler) according to a very simplified choreography: a customer can order a trip with the travel agent, and later may either cancel or confirm already reserved trip.

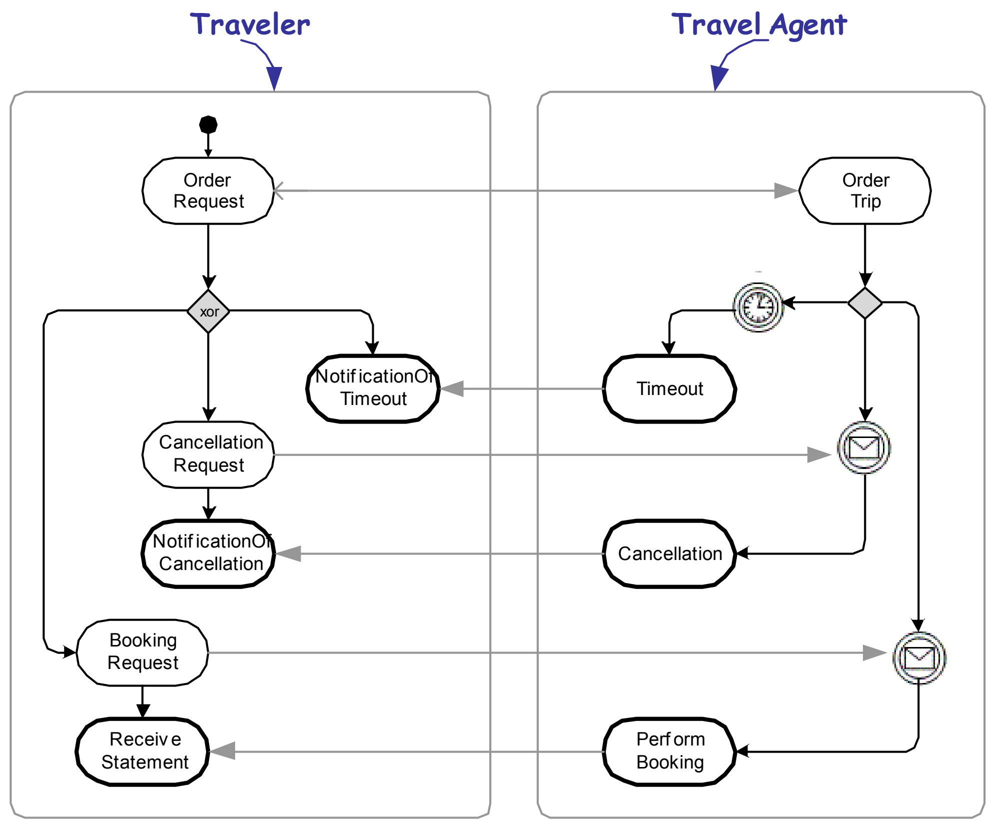

Figure 3: The Ticket Reservation scenario

## 4.1 BPEL4WS process and WSDL portTypes

The following code example shows the abstract WSDL interfaces of the Travel Agent service, the abstract WSDL interface of the Customer service, and the relationship between the two services (or corresponding roles).

```
<?xml version = "1.0" ?>
<definitions name = "TravelAgent"
targetNamespace="http://example.com/travelagent/wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:bpws=http://schemas.xmlsoap.org/ws/2003/03/business-process/
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">

<!-- data type definitions and message definitions are omitted-->

<!-- port type definitions -->

```

```
<portType name="InterfaceOfTravelAgent">
   <operation name="OrderTrip">
      <input message="orderRequest"/>
      <output message="orderAcknowledgement"/>
   </operation>

   <operation name="CancelReservation">
      <input message="cancellationRequest"/>
   </operation>

   <operation name="PerformBooking">
      <input message="bookingRequest"/>
      <output message="bookingConfirmation"/>
   </operation>
</portType>

<portType name="InterfaceOfCustomer">
   <operation name="NotificationOfCancellation">
      <input message="cancellationResponse"/>
   </operation>

   <operation name="NotificationOfTimeout">
      <input message="timeoutMsg"/>
   </operation>

   <operation name="ReceiveStatement">
      <input message="statement"/>
   </operation>
</portType>

<!—partner link type definitions -->

<plnk:partnerLinkType name=”TravelAgentService”>
   <plnk:role name=”TravelAgent”>
      <plnk:portType name=”InterfaceOfTravelAgent”/>
   </plnk:role>
   <plnk:role name=”Customer”>
      <plnk:portTYpe name=”InterfaceOfCustomer”/>
   </plnk:role>
</plnk:partnerLinkType>

<!—definition of properties -->

<bpws:property name=“reservationID” type=“xsd:string”/>

<!—- property aliases are omitted-->
</definitions>

```

The following code example shows the BPEL4WS abstract process of the Travel Agent service.

```
<process name = "ReservationAndBookingTickets"
   targetNamespace="http://example.com/travelagent"
   xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
   xmlns:taw=“http://example.com/travelagent/wsdl”
   abstractProcess="yes">

   <partnerLinks>
      <partnerLink name=”TravelAgency”
         partnerLinkType=”taw:TravelAgencyService”
         partnerRole=“Customer“
         myRole=“TravelAgent“/>
   </partnerLinks>

   <correlationSets>
      <correlationSet name=”reservationCorrelation”
            properties=”taw:reservationID”/>
   </correlationSets>

   <sequence>
      <receive partnerLink="TravelAgency"
         portType="taw:InterfaceOfTravelAgent"
         operation="OrderTrip"
         createInstance="yes">
         <correlations>
            <correlation set="reservationCorrelation"
                  initiate="yes"/>
         </correlations>
      </receive>
      <pick>
         <onAlarm duration="P0Y0M1D">
            <invoke partnerLink="TravelAgency"
                  portType="taw:InterfaceOfCustomer"
                  operation="NotificationOfTimeout">
              <correlations>
                 <correlation set="reservationCorrelation"
                          pattern="out"/>
              </correlations>
            </invoke>
         </onAlarm>
         <onMessage partnerLink="TravelAgency"
                  portType="taw:InterfaceOfTravelAgent"
                  operation="CancelReservation">
            <correlations>
               <correlation set="reservationCorrelation"/>
            </correlations>
            <invoke partnerLink="TravelAgency"
                  portType="taw:InterfaceOfCustomer"
                  operation="NotificationOfCancellation">
               <correlations>
                 <correlation set="reservationCorrelation"
                    pattern="out"/>
               </correlations>
            </invoke>
         </onMessage>
         <onMessage partnerLink="TravelAgency"
                  portType="taw:InterfaceOfTravelAgent"
                  operation="PerformBooking">
             <correlations>
                <correlation set="reservationCorrelation"/>
             </correlations>
             <invoke partnerLink="TravelAgency"
                    portType="taw:InterfaceOfCustomer"
                    operation="ReceiveStatement">
                 <correlations>
                     <correlation set="reservationCorrelation"
                          pattern="out"/>
                 </correlations>
            </invoke>
         </onMessage>
      </pick>
   </sequence>
</process>
```

The Travel Agent service provider may publish this BPEL4WS abstract process and accompanying abstract WSDL interface in a UDDI registry. In this way any customer may use this description in order to understand requirements the Travel Agent service exposes in the context of this scenario.

## 4.2 UDDI V2 Registrations

The following code examples show the UDDI registrations for the abstract WSDL interfaces and the BPEL4WS abstract that were used in the previous section.

### 4.2.1 WSDL portTypes

According to the Technical Note for using WSDL in UDDI **[WSDLTN]**, the WSDL portTypes that are used in the BPEL4WS process definitions are published as separate tModels as follows:

```
<tModel tModelKey="uuid:a1..." >
    <name>InterfaceOfTravelAgent</name>
    <overviewDoc>
         <overviewURL>http://location/travelagent.wsdl<overviewURL>
    <overviewDoc>
    <categoryBag>
        <keyedReference
            tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"
            keyName="uddi-org:xml:namespace"
            keyValue="http://example.com/travelagent/wsdl" />
        <keyedReference
            tModelKey="uuid:6e090afa-33e5-36eb-81b7-1ca18373f457"
            keyName="uddi-org:wsdl:types"
            keyValue="portType" />
    </categoryBag>
</tModel>
```

```
<tModel tModelKey="uuid:a2..." >
    <name>InterfaceOfCustomer</name>
    <overviewDoc>
         <overviewURL>http://location/customer.wsdl<overviewURL>
    <overviewDoc>
    <categoryBag>
        <keyedReference
            tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"
            keyName="uddi-org:xml:namespace"
            keyValue="http://example.com/travelagent/wsdl" />
        <keyedReference
            tModelKey="uuid:6e090afa-33e5-36eb-81b7-1ca18373f457"
            keyName="uddi-org:wsdl:types"
            keyValue="portType" />
    </categoryBag>
</tModel>
```

### 4.2.2 BPEL4WS process

```
<tModel tModelKey="uuid:b1..." >
    <name>ReservationAndBookingTickets</name>
    <overviewDoc>
         <overviewURL>http://location/reservation.bpel<overviewURL>
    <overviewDoc>
    <categoryBag>
        <keyedReference
            tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"
            keyName="uddi-org:xml:namespace"
            keyValue="http://example.com/travelagent" />
        <keyedReference
            tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"
            keyName="uddi-org:bpel:types”
            keyValue="process" />
        <keyedReference
            tModelKey="uuid:ef2dcc0a-edc8-343d-9913-d2e61777a90c"
            keyName="uddi-org:bpel:portTypeReference"
            keyValue="uuid:a1..." />
        <keyedReference
```

```
            tModelKey="uuid:ef2dcc0a-edc8-343d-9913-d2e61777a90c"
            keyName="uddi-org:bpel:portTypeReference"
            keyValue="uuid:a2..." />
    </categoryBag>
</tModel>
```

### 4.2.3 WSDL port

```
<businessService
  serviceKey="d1..."
  businessKey="e1...">
  …
  <bindingTemplates>
    <bindingTemplate
        bindingKey="c1..."
        serviceKey="d1...">
        <accessPoint URLType="http">
          http://location/sample
        </accessPoint>
        <tModelInstanceDetails>
            <tModelInstanceInfo
                tModelKey="uuid:e1...">
                <description xml:lang="en">
                    The wsdl:binding that this wsdl:port implements.
                    The instanceParms specifies the port local name.
                </description>
                <instanceDetails>
                    <instanceParms>TravelAgentPort</instanceParms>
                </instanceDetails>
            </tModelInstanceInfo>
            <tModelInstanceInfo
                tModelKey="uuid:a1...">
                <description xml:lang="en">
                    The wsdl:portType that this wsdl:port implements.
                </description>
            </tModelInstanceInfo>
            <tModelInstanceInfo
                tModelKey="uuid:b1...">
                <description xml:lang="en">
                    The bpel:process this wsdl:port supports.
                </description>
            </tModelInstanceInfo>
        </tModelInstanceDetails>
    </bindingTemplate>
  </bindingTemplates>
</businessService>
```

## 4.3 Sample V2 Queries

### 4.3.1 Find tModel for process name

Find the process tModel for ReservationAndBookingTickets in the namespace http://example.com/travelagent.

```
<find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
    <name>ReservationAndBookingTickets</name>
    <categoryBag>
        <keyedReference
          tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"
          keyValue="process"/>
        <keyedReference
          tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"
          keyValue="http://example.com/travelagent"/>
    </categoryBag>
</find_tModel>
```

This should return the tModelKey "uuid:b1…".

### 4.3.2 Find processes for portTypes

Find all processes that use the InterfaceOfTravelAgent portType.

```
<find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
    <categoryBag>
        <keyedReference
          tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"
          keyValue="process"/>
        <keyedReference
          tModelKey="uuid:ef2dcc0a-edc8-343d-9913-d2e61777a90c"
          keyValue="a1…"/>
    </categoryBag>
</find_tModel>
```

This should return the tModelKey "uuid:b1…".

### 4.3.3 Find portTypes for process

Find all portTypes used in the ReservationAndBookingTickets process.

```
<get_tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2">
    <tModelKey>uuid:b1…</tModelKey>
</get_tModelDetail>
```

This should return the tModel registration for the process tModel with the key "uuid:b1…". The tModelKeys for the portTypes used in the process can be obtained from the process tModel's categoryBag. Once retrieved, the second call is made to get the tModel registrations for the portTypes with the keys "uuid:a1…" (InterfaceOfTravelAgent) and "uuid:a2…" (InterfaceOfCustomer).

```
<get_tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2">
    <tModelKey>uuid:a1…</tModelKey>
    <tModelKey>uuid:a2…</tModelKey>
</get_tModelDetail>
```

### 4.3.4 Find implementations for process

Find all implementations of ReservationAndBookingTickets.

Because the serviceKey attribute is required in the find_binding call in the UDDI V2 API, it is not possible to find all implementations of a process with a single call.  A find_service call must be made first to get the keys of all services that contain a bindingTemplate that references the process, then either the details of each such service must be retrieved with a get_serviceDetail call and the appropriate bindingTemplate looked for among the bindingTemplates of the service, or a find_binding call must be made for each service, with the serviceKey attribute set accordingly.  The following example shows the use of a find_binding call.

This first call gets the list of services that have a bindingTemplate that references the process.

```
<find_service generic="2.0" xmlns="urn:uddi-org:api_v2">
    <tModelBag>
        <tModelKey>uuid:b1…</tModelKey>
    </tModelBag>
```

```
</find_binding>
```

This should return the serviceKey “d1...”.

Now the second call is made to find the appropriate bindings of this particular service.

```
<find_binding serviceKey=”d1...” generic=”2.0” xmlns=”urn:uddi-org:api_v2”>
    <tModelBag>
        <tModelKey>uuid:b1...</tModelKey>
    </tModelBag>
</find_binding>
```

This should return the bindingKey “c1…”.

## 4.4 UDDI V3 Registrations

Illustrating all this using UDDI V3 examples that use uri’s for keys is probably clearer. The following sections illustrate our example’s registrations and searching using UDDI V3..

### 4.4.1 WSDL portTypes

Under V3, the WSDL portType tModels shown in the above section on WSDL portTypes would be published using domain keys which are based on ownership of the TravelAgent.com domain keyGenerator, which this company would have previously published in the UDDI registry. This keyGenerator acts as a “license” for publishing UDDI artifacts whose keys are derived from that domain key:

```
<tModel tModelKey="uddi:TravelAgent.com:TravelAgentInterface_portType" >
    <name>InterfaceOfTravelAgent</name>
    <overviewDoc>
         <overviewURL>http://location/travelagent.wsdl<overviewURL>
    <overviewDoc>
    <categoryBag>
        <keyedReference
            tModelKey="uddi:uddi.org:xml:namespace"
            keyName="uddi-org:xml:namespace"
            keyValue="http://example.com/travelagent/wsdl" />
        <keyedReference
            tModelKey="uddi:uddi.org:wsdl:types"
            keyName="uddi-org:wsdl:types"
            keyValue="portType" />
    </categoryBag>
</tModel>
```

```
<tModel tModelKey="uddi:TravelAgent.com:CustomerInterface_portType" >
    <name>InterfaceOfCustomer</name>
    <overviewDoc>
         <overviewURL>http://location/customer.wsdl<overviewURL>
    <overviewDoc>
    <categoryBag>
        <keyedReference
            tModelKey="uddi:uddi.org:xml:namespace"
            keyName="uddi-org:xml:namespace"
            keyValue="http://example.com/travelagent/wsdl" />
        <keyedReference
            tModelKey="uddi:uddi.org:wsdl:types"
            keyName="uddi-org:wsdl:types"
            keyValue="portType" />
    </categoryBag>
</tModel>
```

### 4.4.2 BPEL4WS process

```
<tModel
tModelKey="uddi:TravelAgent.com:ReservationAndBookingTicketsProcess" >
    <name>ReservationAndBookingTickets</name>
    <overviewDoc>
         <overviewURL>http://location/reservation.bpel<overviewURL>
    <overviewDoc>
    <categoryBag>
        <keyedReference
            tModelKey="uddi:uddi.org:xml:namespace"
            keyName="uddi-org:xml:namespace"
            keyValue="http://example.com/travelagent" />
        <keyedReference
            tModelKey="uddi:uddi.org:bpel:types"
            keyName="uddi-org:bpel:types”
            keyValue="process" />
        <keyedReference
            tModelKey="uddi:uddi.org:bpel:portTypeReference"
            keyName="uddi-org:bpel:portTypeReference"
            keyValue="uddi:TravelAgent.com:TravelAgentInterface_portType"
/>
        <keyedReference
            tModelKey="uddi:uddi.org:bpel:portTypeReference"
            keyName="uddi-org:bpel:portTypeReference"
            keyValue="UDDI:TravelAgent.com:CustomerInterface" />
    </categoryBag>
</tModel>
```

### 4.4.3 WSDL port

```
<businessService
  serviceKey="uddi:TravelAgent.com:service1"
  businessKey="uddi:TravelAgent.com:StoreFront">
  …
  <bindingTemplates>
    <bindingTemplate
        bindingKey="uddi:TravelAgent.com:TravelAgentPort"
        serviceKey="uddi:TravelAgent.com:service1">
        <accessPoint URLType="http">
          http://location/sample
        </accessPoint>
        <tModelInstanceDetails>
            <tModelInstanceInfo
                tModelKey="uddi:TravelAgent.com:StoreFront">
                <description xml:lang="en">
                    The wsdl:binding that this wsdl:port implements.
                    The instanceParms specifies the port local name.
                </description>
                <instanceDetails>
                    <instanceParms>TravelAgentPort</instanceParms>
                </instanceDetails>
            </tModelInstanceInfo>
            <tModelInstanceInfo

    tModelKey="uddi:TravelAgent.com:TravelAgentInterface_portType">
                <description xml:lang="en">
                    The wsdl:portType that this wsdl:port implements.
                </description>
            </tModelInstanceInfo>
            <tModelInstanceInfo
                tModelKey=

"uddi:TravelAgent.com:ReservationAndBookingTicketsProcess ">
                <description xml:lang="en">
                    The bpel:process this wsdl:port supports.
                </description>
            </tModelInstanceInfo>
        </tModelInstanceDetails>
    </bindingTemplate>
  </bindingTemplates>
</businessService>
```

## 4.5 Sample V3 Queries

### 4.5.1 Find tModel for process name

Find the process tModel for the ReservationAndBookingTickets business process in the namespace http://example.com/travelagent.

```
<find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
    <name>ReservationAndBookingTickets</name>
    <categoryBag>
        <keyedReference
          tModelKey="uddi:uddi.org:bpel:types"
          keyValue="process"/>
        <keyedReference
          tModelKey="uddi:uddi.org:xml:namespace"
          keyValue="http://example.com/travelagent"/>
    </categoryBag>
</find_tModel>
```

This should return the tModelKey "uddi:TravelAgent.com:ReservationAndBookingTicketsProcess".

### 4.5.2 Find processes for portTypes

Find all processes that use the InterfaceOfTravelAgent portType.

```
<find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
    <categoryBag>
        <keyedReference
          tModelKey="uddi:uddi.org:bpel:types"
          keyValue="process"/>
        <keyedReference
          tModelKey="uddi:uddi.org:bpel:portTypeReference"
          keyValue="uddi:TravelAgent.com:TravelAgentInterface_portType"/>
    </categoryBag>
</find_tModel>
```

This should return the tModelKey "uddi:TravelAgent.com:ReservationAndBookingTicketsProcess".

### 4.5.3 Find portTypes for process

Find all portTypes used in the ReservationAndBookingTickets process.

```
<get_tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2">
    <tModelKey>uddi:TravelAgent.com:ReservationAndBookingTicketsProcess
           </tModelKey>
</get_tModelDetail>
```

This should return the tModel registration for the process tModel with the key “uddi:TravelAgent.com:ReservationAndBookingTicketsProcess”. The tModelKeys for the portTypes used in the process can be obtained from the process tModel’s categoryBag. Once retrieved, the second call is made to get the tModel registrations for the portTypes with the keys “uddi:TravelAgent.com:TravelAgentInterface_portType” (InterfaceOfTravelAgent) and “uddi:TravelAgent.com:CustomerInterface_portType” (InterfaceOfCustomer).

```
<get_tModelDetail generic=”2.0” xmlns=”urn:uddi-org:api_v2”>
<tModelKey>uddi:TravelAgent.com:TravelAgentInterface_portType</tModelKey>
    <tModelKey>uddi:TravelAgent.com:CustomerInterface_portType</tModelKey>
</get_tModelDetail>
```

### 4.5.4 Find implementations for process

Find all implementations of ReservationAndBookingTickets.

Because the serviceKey attribute is required in the find_binding call in the UDDI V2 API, it is not possible to find all implementations of a process with a single call.  A find_service call must be made first to get the keys of all services that contain a bindingTemplate that references the process, then either the details of each such service must be retrieved with a get_serviceDetail call and the appropriate bindingTemplate looked for among the bindingTemplates of the service, or a find_binding call must be made for each service, with the serviceKey attribute set accordingly.  The following example shows the use of a find_binding call.

This first call gets the list of services that have a bindingTemplate that references the process.

```
<find_service generic=”2.0” xmlns=”urn:uddi-org:api_v2”>
    <tModelBag>
        <tModelKey>uddi:TravelAgent.com:ReservationAndBookingTicketsProcess
               </tModelKey>
    </tModelBag>
</find_binding>
```

This should return the serviceKey “uddi:TravelAgent.com:service1”.

Now the second call is made to find the appropriate bindings of this particular service.

```
<find_binding serviceKey=”uddi:TravelAgent.com:service1” generic=”2.0”
xmlns=”urn:uddi-org:api_v2”>
    <tModelBag>
        <tModelKey>uddi:TravelAgent.com:ReservationAndBookingTicketsProcess
               </tModelKey>
    </tModelBag>
</find_binding>
```

This should return the bindingKey “uddi:TravelAgent.com:TravelAgentPort”.

# 5 References

## 5.1 Normative

| | |
|---|---|
| **[BPEL4WS]** | T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana, *Business Process Execution Language for Web Services Version 1.1*, http://ifr.sap.com/bpel4ws, May 2003. |
| **[RFC2119]** | S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997. |
| **[WSDL11]** | E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *Web Services Description Language (WSDL) 1.1*, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, W3C Note, March 2001. |
| **[WSDLBP]** | J. Colgrave, K. Januszewski, *Using WSDL in a UDDI Registry, Version 1.08*, http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.htm, OASIS UDDI TC Best Practice, November 2002. |
| **[WSDLTN]** | J. Colgrave, K. Januszewski, *Using WSDL in a UDDI Registry, Version 2.0*, http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm, OASIS UDDI TC Technical Note, June 2003. |

# Appendix A. Acknowledgments

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

## Appendix B. Revision History

| Rev | Date | By Whom | What |
|---|---|---|---|
| 0.8 | Jan 29, 2004 | C. v. Riegen, I. Trickovic | First complete draft |
| 0.9 | March 22, 2004 | T. Bellwood | Corrected a few typos; Added sections on V3 registrations and queries |
| 1.0 | April 15, 2004 | I. Trickovic | Corrected figure #2 (included in section 2.1.2); Corrected the BPEL4WS abstract process (section 4.1); Addressed a few additional wording issues |

## Appendix C. Notices