

---

# 1 Technical Note

## 2 Using BPEL4WS in a UDDI registry

### 3 Document identifier:

4 [uddi-spec-tc-tn-bpel](#)

### 5 This Version:

6 <http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-bpel-20040725.htm>

### 8 Latest Version:

9 <http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-bpel.htm>

### 10 Authors:

11 Claus von Riegen, SAP ([claus.von.riegen@sap.com](mailto:claus.von.riegen@sap.com))  
12 Ivana Trickovic, SAP ([ivana.trickovic@sap.com](mailto:ivana.trickovic@sap.com))

### 13 Editors:

14 Luc Clément, Systinet ([luc.clement@systinet.com](mailto:luc.clement@systinet.com))  
15 Andrew Hately, IBM ([hately@us.ibm.com](mailto:hately@us.ibm.com))

### 16 Contributors:

17 Tom Bellwood, IBM ([bellwood@us.ibm.com](mailto:bellwood@us.ibm.com))

### 18 Abstract:

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

### 24 Status:

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

27 Committee members should send comments on this technical note to the [uddi-spec@lists.oasis-open.org](mailto:uddi-spec@lists.oasis-open.org)  
28 list. Others should subscribe to and send comments to the [uddi-spec-comment@lists.oasis-open.org](mailto:uddi-spec-comment@lists.oasis-open.org)  
29 list. To subscribe, send an email message to [uddi-spec-comment-request@lists.oasis-open.org](mailto:uddi-spec-comment-request@lists.oasis-open.org)  
30 with the word "subscribe" as the body  
31 of the message.

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

---

## Table of Contents

37	1	Introduction .....	3
38	1.1	Problem statement .....	3
39	1.2	Reliance on WSDL Technical Note .....	4
40	1.3	Terminology .....	4
41	2	Technical Note Solution.....	5
42	2.1	Definitions .....	5
43	2.1.1	BPEL4WS Data Model .....	5
44	2.1.2	Mapping BPEL4WS to UDDI.....	6
45	3	tModel definitions .....	8
46	3.1	BPEL Entity Type tModel .....	8
47	3.1.1	Design Goals.....	8
48	3.1.2	Definition .....	8
49	4	Example.....	10
50	4.1	BPEL4WS process and WSDL portTypes .....	10
51	4.2	UDDI V2 Registrations.....	13
52	4.2.1	WSDL portTypes .....	13
53	4.2.2	BPEL4WS process .....	13
54	4.2.3	WSDL port.....	14
55	4.3	Sample V2 Queries .....	15
56	4.3.1	Find tModel for process name.....	15
57	4.3.2	Find processes for portTypes.....	15
58	4.3.3	Find portTypes for process .....	15
59	4.3.4	Find implementations for process .....	16
60	4.4	UDDI V3 Registrations.....	16
61	4.4.1	WSDL portTypes .....	16
62	4.4.2	BPEL4WS process .....	17
63	4.4.3	WSDL port.....	17
64	4.5	Sample V3 Queries .....	18
65	4.5.1	Find tModel for process name.....	18
66	4.5.2	Find processes for portTypes.....	18
67	4.5.3	Find portTypes for process .....	19
68	4.5.4	Find implementations for process .....	19
69	5	References .....	20
70	5.1	Normative.....	20
71		Appendix A. Acknowledgments.....	21
72		Appendix B. Revision History.....	22
73		Appendix C. Notices .....	23
74			

75

# 1 Introduction

76

## 1.1 Problem statement

77

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]**).

78

79

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.

80

81

82

83

84

85

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.

86

87

88

89

90

91

92

93

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.

94

95

96

97

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:

100

101

102

103

104

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

105

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

106

107

108

109

110

111

112

113

- 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.

114

115

116

117

118

119

120

Publishing and discovering multi-party processes (including processes with just two participants) in a UDDI registry is out of scope of this Technical Note. BPEL4WS abstract processes could be used for describing the behavior of one participant in a multi-party process. A separate model based on BPEL4WS abstract processes is needed for describing the way how multiple Web services interact in the context of a scenario. We envisage that the proposal given in this document can be easily extended in order to store and retrieve multi-party processes to and from a UDDI registry.

121 **1.2 Reliance on WSDL Technical Note**

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

127 **1.3 Terminology**

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

---

## 130 2 Technical Note Solution

### 131 2.1 Definitions

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

#### 134 2.1.1 BPEL4WS Data Model

135 The BPEL4WS model supports definition of the observable behavior of a Web service  
136 participating in a long-running conversation with other Web services. More particularly, the model  
137 defines abstract processes, which may be used for describing the observable behavior. These  
138 processes are in the scope of this Technical Note. BPEL4WS introduces features, such as  
139 process, action, correlation, role, partner link, etc, needed to describe the behavioral aspects of  
140 Web services. Figure 1 shows a sub-set of those features of interest in the context of this note  
141 and relationships between them. An action is one of BPEL4WS activities dealing with Web  
142 services interactions (invoking an operation of another Web service or waiting for a message to  
143 be received). A process defines sequencing of Web services interactions and other BPEL4WS  
144 primitive activities.

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

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

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

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

158

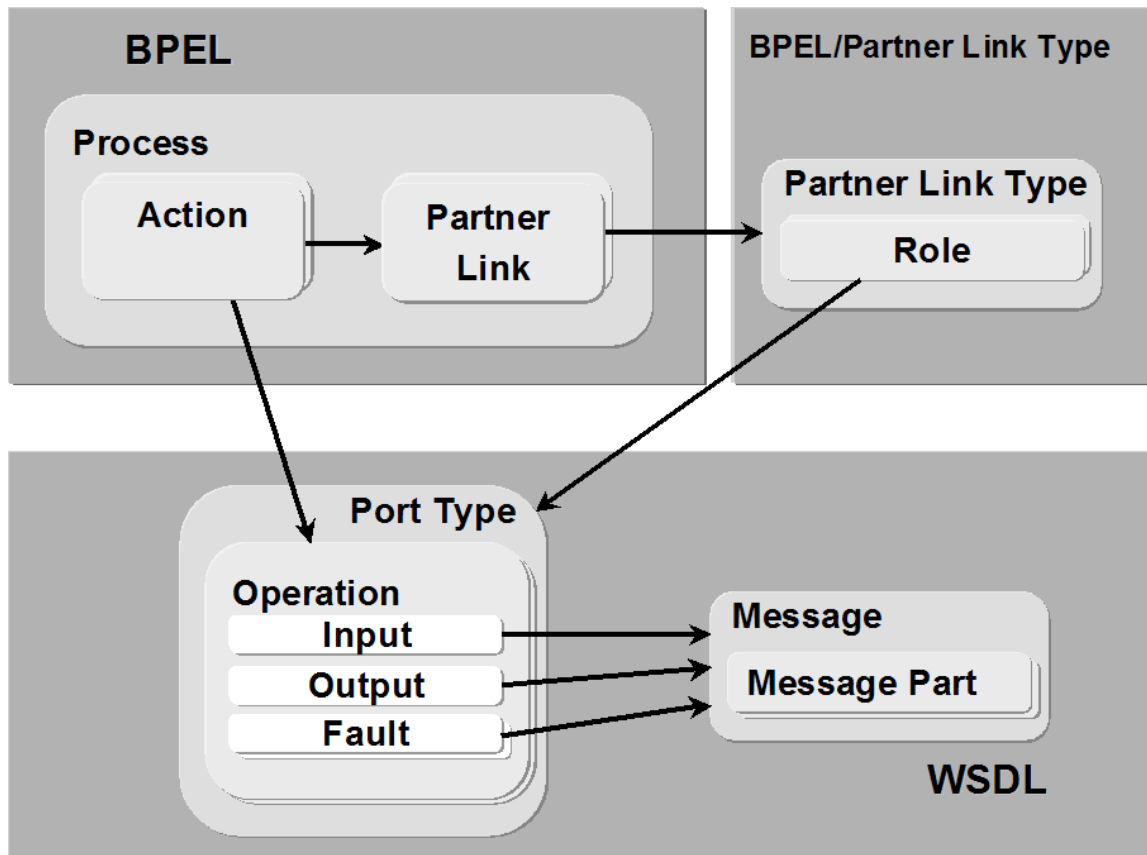


Figure 1: The BPEL model and its relationship with WSDL

### 2.1.2 Mapping BPEL4WS to UDDI

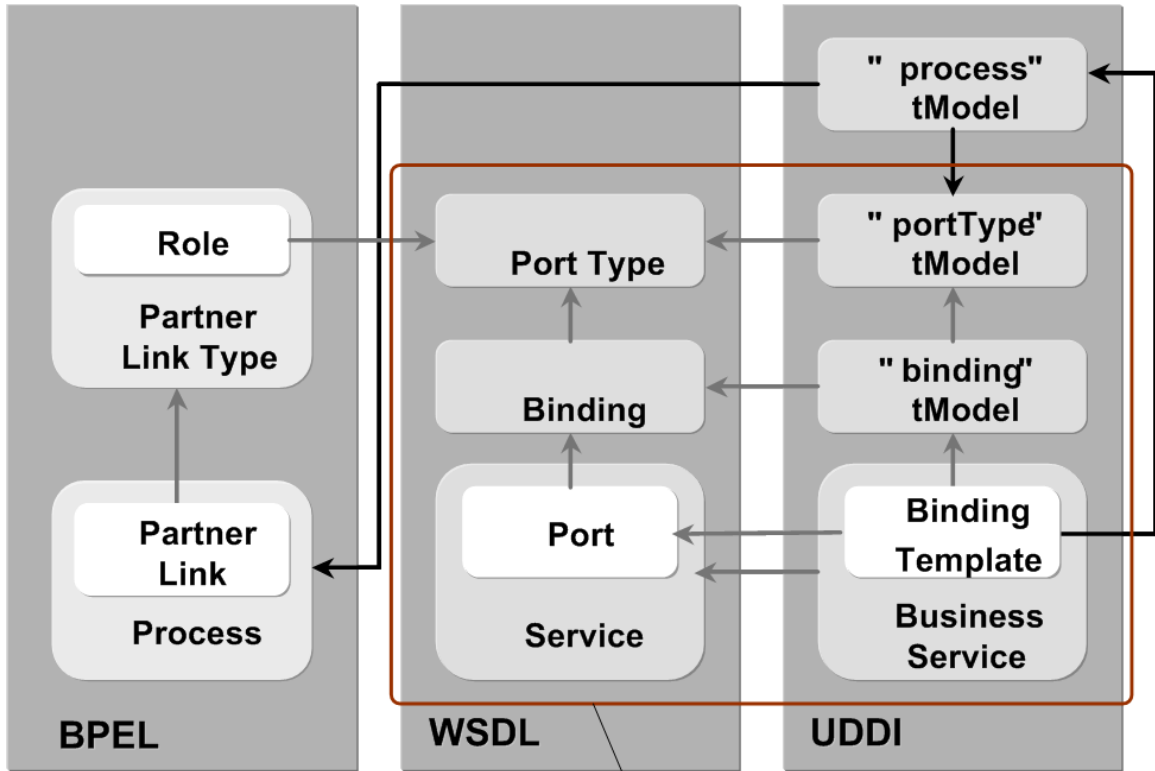
BPEL4WS abstract processes are published as separate UDDI tModels. They are named with the BPEL4WS process name. They are categorized as BPEL4WS process definitions, using a category system defined in this technical note. Their overviewDoc references an external BPEL4WS document that contains the process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



180  
181

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:** yes

##### 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

There is only one valid value that can be used with this category system:

keyValue	Description	UDDI Entity
process	Represents a UDDI entity categorized as a bpel:process	tModel



220 **3.1.2.3 Example of Use**

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

```
222 <categoryBag>
223     <keyedReference
224         tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"
225         keyName="BPEL Entity type"
226         keyValue="process"/>
227     ...
228 </categoryBag>
```

229

## 4 Example

230

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.

232

Figure 3 shows the overall process: the Travel Agent interacts with a Customer (a traveler)

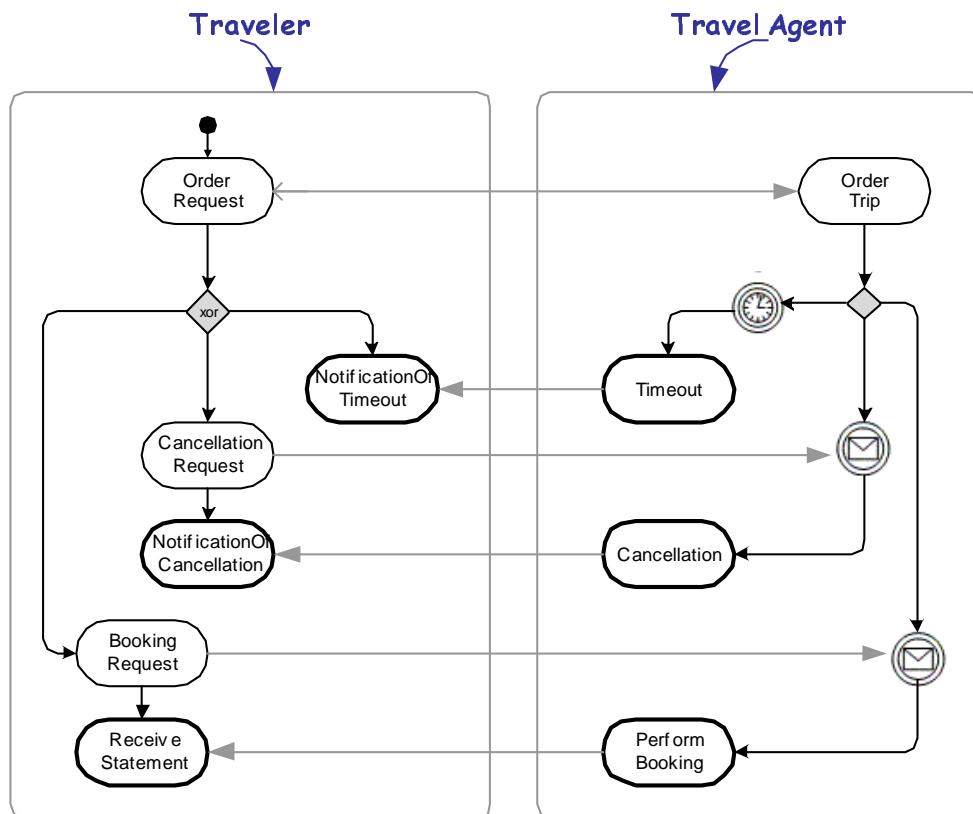
233

according to a very simplified choreography: a customer can order a trip with the travel agent,

234

and later may either cancel or confirm already reserved trip.

235



236

237

Figure 3: The Ticket Reservation scenario

238

### 4.1 BPEL4WS process and WSDL portTypes

239

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).

240

241

242

243

244

245

246

247

248

249

250

251

252

253

```

<?xml version="1.0" ?>
<definitions name="TravelAgent"
targetNamespace="http://example.com/travelagent/wsd1"
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">

```

```

254 <operation name="OrderTrip">
255   <input message="orderRequest" />
256   <output message="orderAcknowledgement" />
257 </operation>
258
259 <operation name="CancelReservation">
260   <input message="cancellationRequest" />
261 </operation>
262
263 <operation name="PerformBooking">
264   <input message="bookingRequest" />
265   <output message="bookingConfirmation" />
266 </operation>
267 </portType>
268
269 <portType name="InterfaceOfCustomer">
270   <operation name="NotificationOfCancellation">
271     <input message="cancellationResponse" />
272   </operation>
273
274   <operation name="NotificationOfTimeout">
275     <input message="timeoutMsg" />
276   </operation>
277
278   <operation name="ReceiveStatement">
279     <input message="statement" />
280   </operation>
281 </portType>
282
283 <!--partner link type definitions -->
284
285 <plnk:partnerLinkType name="TravelAgentService">
286   <plnk:role name="TravelAgent">
287     <plnk:portType name="InterfaceOfTravelAgent" />
288   </plnk:role>
289   <plnk:role name="Customer">
290     <plnk:portType name="InterfaceOfCustomer" />
291   </plnk:role>
292 </plnk:partnerLinkType>
293
294 <!--definition of properties -->
295
296 <bpws:property name="reservationID" type="xsd:string" />
297
298 <!-- property aliases are omitted-->
299 </definitions>

```

300

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

```

303 <process name="ReservationAndBookingTickets"
304   targetNamespace="http://example.com/travelagent"
305   xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
306   xmlns:taw="http://example.com/travelagent/wsdl"
307   abstractProcess="yes">
308
309   <partnerLinks>
310     <partnerLink name="TravelAgency"
311       partnerLinkType="taw:TravelAgencyService"
312       partnerRole="Customer"
313       myRole="TravelAgent" />
314   </partnerLinks>
315
316   <correlationSets>
317     <correlationSet name="reservationCorrelation"
318       properties="taw:reservationID" />
319   </correlationSets>
320
321 </sequence>

```

```

322 <receive partnerLink="TravelAgency"
323     portType="taw:InterfaceOfTravelAgent"
324     operation="OrderTrip"
325     createInstance="yes">
326     <correlations>
327         <correlation set="reservationCorrelation"
328             initiate="yes"/>
329     </correlations>
330 </receive>
331 <pick>
332     <onAlarm duration="P0Y0M1D">
333         <invoke partnerLink="TravelAgency"
334             portType="taw:InterfaceOfCustomer"
335             operation="NotificationOfTimeout">
336             <correlations>
337                 <correlation set="reservationCorrelation"
338                     pattern="out"/>
339             </correlations>
340         </invoke>
341     </onAlarm>
342     <onMessage partnerLink="TravelAgency"
343         portType="taw:InterfaceOfTravelAgent"
344         operation="CancelReservation">
345         <correlations>
346             <correlation set="reservationCorrelation"/>
347         </correlations>
348         <invoke partnerLink="TravelAgency"
349             portType="taw:InterfaceOfCustomer"
350             operation="NotificationOfCancellation">
351             <correlations>
352                 <correlation set="reservationCorrelation"
353                     pattern="out"/>
354             </correlations>
355         </invoke>
356     </onMessage>
357     <onMessage partnerLink="TravelAgency"
358         portType="taw:InterfaceOfTravelAgent"
359         operation="PerformBooking">
360         <correlations>
361             <correlation set="reservationCorrelation"/>
362         </correlations>
363         <invoke partnerLink="TravelAgency"
364             portType="taw:InterfaceOfCustomer"
365             operation="ReceiveStatement">
366             <correlations>
367                 <correlation set="reservationCorrelation"
368                     pattern="out"/>
369             </correlations>
370         </invoke>
371     </onMessage>
372 </pick>
373 </sequence>
374 </process>

```

375

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

380

## 381 4.2 UDDI V2 Registrations

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

### 384 4.2.1 WSDL portTypes

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

```
387 <tModel tModelKey="uuid:a1..." >  
388   <name>InterfaceOfTravelAgent</name>  
389   <overviewDoc>  
390     <overviewURL>http://location/travelagent.wsdl</overviewURL>  
391   </overviewDoc>  
392   <categoryBag>  
393     <keyedReference  
394       tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"  
395       keyName="uddi-org:xml:namespace"  
396       keyValue="http://example.com/travelagent/wsdl" />  
397     <keyedReference  
398       tModelKey="uuid:6e090afa-33e5-36eb-81b7-1ca18373f457"  
399       keyName="uddi-org:wSDL:types"  
400       keyValue="portType" />  
401   </categoryBag>  
402 </tModel>
```

403

```
404 <tModel tModelKey="uuid:a2..." >  
405   <name>InterfaceOfCustomer</name>  
406   <overviewDoc>  
407     <overviewURL>http://location/customer.wsdl</overviewURL>  
408   </overviewDoc>  
409   <categoryBag>  
410     <keyedReference  
411       tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"  
412       keyName="uddi-org:xml:namespace"  
413       keyValue="http://example.com/travelagent/wsdl" />  
414     <keyedReference  
415       tModelKey="uuid:6e090afa-33e5-36eb-81b7-1ca18373f457"  
416       keyName="uddi-org:wSDL:types"  
417       keyValue="portType" />  
418   </categoryBag>  
419 </tModel>
```

420

### 421 4.2.2 BPEL4WS process

```
422 <tModel tModelKey="uuid:b1..." >  
423   <name>ReservationAndBookingTickets</name>  
424   <overviewDoc>  
425     <overviewURL>http://location/reservation.bpel</overviewURL>  
426   </overviewDoc>  
427   <categoryBag>  
428     <keyedReference  
429       tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"  
430       keyName="uddi-org:xml:namespace"  
431       keyValue="http://example.com/travelagent" />  
432     <keyedReference  
433       tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"  
434       keyName="uddi-org:bpel:types"  
435       keyValue="process" />  
436     <keyedReference
```

437  
438  
439  
440  
441  
442  
443  
444  
445

```
        tModelKey="uuid:082b0851-25d8-303c-b332-f24a6d53e38e"  
        keyName="uddi-org:wsdl:portTypeReference"  
        keyValue="uuid:a1..." />  
    <keyedReference  
        tModelKey="uuid:082b0851-25d8-303c-b332-f24a6d53e38e"  
        keyName="uddi-org:wsdl:portTypeReference"  
        keyValue="uuid:a2..." />  
    </categoryBag>  
</tModel>
```

446

### 4.2.3 WSDL port

447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484

```
<businessService  
  serviceKey="dl..."  
  businessKey="el...">  
  ...  
  <bindingTemplates>  
    <bindingTemplate  
      bindingKey="cl..."  
      serviceKey="dl...">  
      <accessPoint URLType="http">  
        http://location/sample  
      </accessPoint>  
      <tModelInstanceDetails>  
        <tModelInstanceInfo  
          tModelKey="...">  
          <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>
```

485

486

## 487 **4.3 Sample V2 Queries**

### 488 **4.3.1 Find tModel for process name**

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

```
491 <find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">  
492   <name>ReservationAndBookingTickets</name>  
493   <categoryBag>  
494     <keyedReference  
495       tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"  
496       keyValue="process" />  
497     <keyedReference  
498       tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"  
499       keyValue="http://example.com/travelagent" />  
500   </categoryBag>  
501 </find_tModel>
```

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

### 503 **4.3.2 Find processes for portTypes**

504 Find all processes that use the InterfaceOfTravelAgent portType.

```
505 <find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">  
506   <categoryBag>  
507     <keyedReference  
508       tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"  
509       keyValue="process" />  
510     <keyedReference  
511       tModelKey="uuid:082b0851-25d8-303c-b332-f24a6d53e38e"  
512       keyValue="al..." />  
513   </categoryBag>  
514 </find_tModel>
```

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

### 516 **4.3.3 Find portTypes for process**

517 Find all portTypes used in the ReservationAndBookingTickets process.

```
518 <get_tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2">  
519   <tModelKey>uuid:b1...</tModelKey>  
520 </get_tModelDetail>
```

521

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

```
527 <get_tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2">  
528   <tModelKey>uuid:a1...</tModelKey>  
529   <tModelKey>uuid:a2...</tModelKey>  
530 </get_tModelDetail>
```

531

## 532 4.3.4 Find implementations for process

533 Find all implementations of ReservationAndBookingTickets process.

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

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

```
542 <find_service generic="2.0" xmlns="urn:uddi-org:api_v2">  
543   <tModelBag>  
544     <tModelKey>uuid:b1...</tModelKey>  
545   </tModelBag>  
546 </find_service>
```

547

548 This should return the serviceKey "d1...".

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

```
550 <find_binding serviceKey="d1..." generic="2.0" xmlns="urn:uddi-org:api_v2">  
551   <tModelBag>  
552     <tModelKey>uuid:b1...</tModelKey>  
553   </tModelBag>  
554 </find_binding>
```

555

556 This should return the bindingKey "c1...".

## 557 4.4 UDDI V3 Registrations

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

### 560 4.4.1 WSDL portTypes

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

```
566 <tModel tModelKey="uddi:TravelAgent.com:TravelAgentInterface_portType">  
567   <name>InterfaceOfTravelAgent</name>  
568   <overviewDoc>  
569     <overviewURL>http://location/travelagent.wsdl</overviewURL>  
570   </overviewDoc>  
571   <categoryBag>  
572     <keyedReference  
573       tModelKey="uddi:uddi.org:xml:namespace"  
574       keyName="uddi-org:xml:namespace"  
575       keyValue="http://example.com/travelagent/wsdl" />  
576     <keyedReference  
577       tModelKey="uddi:uddi.org:wsdl:types"  
578       keyName="uddi-org:wsdl:types"  
579       keyValue="portType" />  
580   </categoryBag>  
581 </tModel>
```

582



```

583 <tModel tModelKey="uddi:TravelAgent.com:CustomerInterface_portType">
584   <name>InterfaceOfCustomer</name>
585   <overviewDoc>
586     <overviewURL>http://location/customer.wsdl</overviewURL>
587   </overviewDoc>
588   <categoryBag>
589     <keyedReference
590       tModelKey="uddi:uddi.org:xml:namespace"
591       keyName="uddi-org:xml:namespace"
592       keyValue="http://example.com/travelagent/wsdl" />
593     <keyedReference
594       tModelKey="uddi:uddi.org:wsdl:types"
595       keyName="uddi-org:wsdl:types"
596       keyValue="portType" />
597   </categoryBag>
598 </tModel>

```

#### 599 4.4.2 BPEL4WS process

```

600 <tModel tModelKey="uddi:TravelAgent.com:ReservationAndBookingTicketsProcess">
601   <name>ReservationAndBookingTickets</name>
602   <overviewDoc>
603     <overviewURL>http://location/reservation.bpel</overviewURL>
604   </overviewDoc>
605   <categoryBag>
606     <keyedReference
607       tModelKey="uddi:uddi.org:xml:namespace"
608       keyName="uddi-org:xml:namespace"
609       keyValue="http://example.com/travelagent" />
610     <keyedReference
611       tModelKey="uddi:uddi.org:bpel:types"
612       keyName="uddi-org:bpel:types"
613       keyValue="process" />
614     <keyedReference
615       tModelKey="uddi:uddi.org:wsdl:porttypereference"
616       keyName="uddi-org:wsdl:portTypeReference"
617       keyValue="uddi:TravelAgent.com:TravelAgentInterface_portType" />
618     <keyedReference
619       tModelKey="uddi:uddi.org:wsdl:porttypereference"
620       keyName="uddi-org:wsdl:portTypeReference"
621       keyValue="UDDI:TravelAgent.com:CustomerInterface" />
622   </categoryBag>
623 </tModel>

```

#### 624 4.4.3 WSDL port

```

625 <businessService
626   serviceKey="uddi:TravelAgent.com:service1"
627   businessKey="uddi:TravelAgent.com:StoreFront">
628   ...
629   <bindingTemplates>
630     <bindingTemplate
631       bindingKey="uddi:TravelAgent.com:TravelAgentPort"
632       serviceKey="uddi:TravelAgent.com:service1">
633       <accessPoint useType="endPoint">
634         http://location/sample
635       </accessPoint>
636       <tModelInstanceDetails>
637         <tModelInstanceInfo
638           tModelKey="uddi:...">
639           <description xml:lang="en">
640             The wsdl:binding that this wsdl:port implements.
641             The instanceParms specifies the port local name.
642           </description>
643           <instanceDetails>
644             <instanceParms>TravelAgentPort</instanceParms>
645           </instanceDetails>

```

```

646         </tModelInstanceInfo>
647     <tModelInstanceInfo
648         tModelKey="uddi:TravelAgent.com:TravelAgentInterface_portType">
649         <description xml:lang="en">
650             The wsdl:portType that this wsdl:port implements.
651         </description>
652     </tModelInstanceInfo>
653     <tModelInstanceInfo
654         tModelKey=
655             "uddi:TravelAgent.com:ReservationAndBookingTicketsProcess">
656         <description xml:lang="en">
657             The bpel:process this wsdl:port supports.
658         </description>
659     </tModelInstanceInfo>
660 </tModelInstanceDetails>
661 </bindingTemplate>
662 </bindingTemplates>
663 </businessService>

```

## 664 4.5 Sample V3 Queries

### 665 4.5.1 Find tModel for process name

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

```

668 <find_tModel xmlns="urn:uddi-org:api_v3">
669     <name>ReservationAndBookingTickets</name>
670     <categoryBag>
671         <keyedReference
672             tModelKey="uddi:uddi.org:bpel:types"
673             keyValue="process" />
674         <keyedReference
675             tModelKey="uddi:uddi.org:xml:namespace"
676             keyValue="http://example.com/travelagent" />
677     </categoryBag>
678 </find_tModel>

```

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

### 681 4.5.2 Find processes for portTypes

682 Find all processes that use the InterfaceOfTravelAgent portType.

```

683 <find_tModel xmlns="urn:uddi-org:api_v3">
684     <categoryBag>
685         <keyedReference
686             tModelKey="uddi:uddi.org:bpel:types"
687             keyValue="process" />
688         <keyedReference
689             tModelKey="uddi:uddi.org:wsdl:porttypereference"
690             keyValue="uddi:TravelAgent.com:TravelAgentInterface_portType" />
691     </categoryBag>
692 </find_tModel>

```

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

695

### 696 **4.5.3 Find portTypes for process**

697 Find all portTypes used in the ReservationAndBookingTickets process.

```
698 <get_tModelDetail xmlns="urn:uddi-org:api_v3">  
699   <tModelKey>uddi:TravelAgent.com:ReservationAndBookingTicketsProcess  
700     </tModelKey>  
701 </get_tModelDetail>
```

702 This should return the tModel registration for the process tModel with the key  
703 "uddi:TravelAgent.com:ReservationAndBookingTicketsProcess". The tModelKeys for the  
704 portTypes used in the process can be obtained from the process tModel's categoryBag. Once  
705 retrieved, the second call is made to get the tModel registrations for the portTypes with the keys  
706 "uddi:TravelAgent.com:TravelAgentInterface\_portType" (InterfaceOfTravelAgent) and  
707 "uddi:TravelAgent.com:CustomerInterface\_portType" (InterfaceOfCustomer).

```
708 <get_tModelDetail xmlns="urn:uddi-org:api_v3">  
709 <tModelKey>uddi:TravelAgent.com:TravelAgentInterface_portType</tModelKey>  
710   <tModelKey>uddi:TravelAgent.com:CustomerInterface_portType</tModelKey>  
711 </get_tModelDetail>
```

### 712 **4.5.4 Find implementations for process**

713 Find all implementations of ReservationAndBookingTickets process.

```
714 <find_binding xmlns="urn:uddi-org:api_v3">  
715   <tModelBag>  
716     <tModelKey>uddi:TravelAgent.com:ReservationAndBookingTicketsProcess  
717   </tModelKey>  
718   </tModelBag>  
719 </find_binding>
```

720 This should return the bindingKey "uddi:TravelAgent.com:TravelAgentPort".

721

722

---

## 5 References

723

### 5.1 Normative

724

**[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.

725

726

727

728

**[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.

729

730

**[WSDL11]**

E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *Web Services Description Language (WSDL) 1.1*,

731

732

733

<http://www.w3.org/TR/2001/NOTE-wsdl-20010315>, W3C Note, March 2001.

734

**[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.

735

736

737

738

**[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.

739

740

741

---

742 **Appendix A. Acknowledgments**

743 The following individuals provided input of this technical note:

744       Jan Pridal, Systinet

745       Svatopluk Dedic, Systinet

746       Ales Lipovy, Systinet

---

## 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
1.0.1	July 19, 2004	C. v. Riegen, I. Trickovic	Addressed issues raised during UDDI TC FTF meeting June 28-30, 2004

---

## Appendix C. Notices

750 OASIS takes no position regarding the validity or scope of any intellectual property or other rights  
751 that might be claimed to pertain to the implementation or use of the technology described in this  
752 document or the extent to which any license under such rights might or might not be available;  
753 neither does it represent that it has made any effort to identify any such rights. Information on  
754 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS  
755 website. Copies of claims of rights made available for publication and any assurances of licenses  
756 to be made available, or the result of an attempt made to obtain a general license or permission  
757 for the use of such proprietary rights by implementors or users of this specification, can be  
758 obtained from the OASIS Executive Director.

759 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
760 applications, or other proprietary rights which may cover technology that may be required to  
761 implement this specification. Please address the information to the OASIS Executive Director.

762 **Copyright © OASIS Open 2004. All Rights Reserved.**

763 This document and translations of it may be copied and furnished to others, and derivative works  
764 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
765 published and distributed, in whole or in part, without restriction of any kind, provided that the  
766 above copyright notice and this paragraph are included on all such copies and derivative works.  
767 However, this document itself does not be modified in any way, such as by removing the  
768 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS  
769 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual  
770 Property Rights document must be followed, or as required to translate it into languages other  
771 than English.

772 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
773 successors or assigns.

774 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
775 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO  
776 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE  
777 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A  
778 PARTICULAR PURPOSE.