



## 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](mailto:claus.von.riegen@sap.com))  
Ivana Trickovic, SAP ([ivana.trickovic@sap.com](mailto:ivana.trickovic@sap.com))

**Contributors:**

Tom Bellwood, IBM ([bellwood@us.ibm.com](mailto: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](mailto:uddi-spec@lists.oasis-open.org) 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) 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) 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/>).

---

33 **Table of Contents**

34	1	Introduction .....	4
35	1.1	Problem statement .....	4
36	1.2	Reliance on WSDL Technical Note .....	4
37	1.3	Terminology .....	5
38	2	Technical Note Solution .....	6
39	2.1	Definitions .....	6
40	2.1.1	BPEL4WS Data Model .....	6
41	2.1.2	Mapping BPEL4WS to UDDI .....	7
42	3	tModel definitions .....	8
43	3.1	BPEL Entity Type tModel .....	8
44	3.1.1	Design Goals .....	8
45	3.1.2	Definition .....	8
46	3.2	WSDL portType Reference tModel .....	9
47	3.2.1	Design Goals .....	9
48	3.2.2	Definition .....	9
49	3.2.3	Valid Values .....	10
50	3.2.4	Example of Use .....	10
51	4	Example .....	11
52	4.1	BPEL4WS process and WSDL portTypes .....	11
53	4.2	UDDI V2 Registrations .....	13
54	4.2.1	WSDL portTypes .....	14
55	4.2.2	BPEL4WS process .....	14
56	4.2.3	WSDL port .....	15
57	4.3	Sample V2 Queries .....	15
58	4.3.1	Find tModel for process name .....	15
59	4.3.2	Find processes for portTypes .....	16
60	4.3.3	Find portTypes for process .....	16
61	4.3.4	Find implementations for process .....	16
62	4.4	UDDI V3 Registrations .....	17
63	4.4.1	WSDL portTypes .....	17
64	4.4.2	BPEL4WS process .....	18
65	4.4.3	WSDL port .....	18
66	4.5	Sample V3 Queries .....	19
67	4.5.1	Find tModel for process name .....	19
68	4.5.2	Find processes for portTypes .....	19
69	4.5.3	Find portTypes for process .....	19
70	4.5.4	Find implementations for process .....	20
71	5	References .....	21
72	5.1	Normative .....	21
73		Appendix A. Acknowledgments .....	22
74		Appendix B. Revision History .....	23

75 Appendix C. Notices ..... 24  
76

---

# 77 1 Introduction

## 78 1.1 Problem statement

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

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

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

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

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

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

107

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

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

## 117 1.2 Reliance on WSDL Technical Note

118 Since BPEL4WS abstract processes operate on WSDL artifacts, a common approach for  
119 mapping WSDL artifacts to the UDDI model is a prerequisite for this technical note in general. In  
120 particular, WSDL port types need to be registered and identified individually in UDDI. Thus, this

121 technical note assumes the application of the Technical Note for Using WSDL in a UDDI Registry,  
122 Version 2.0 **[WSDLTN]**.

### 123 **1.3 Terminology**

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

126

## 2 Technical Note Solution

127

### 2.1 Definitions

128

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.

129

130

#### 2.1.1 BPEL4WS Data Model

131

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.

132

133

134

135

136

137

138

139

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.

140

141

142

143

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

144

145

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.

146

147

148

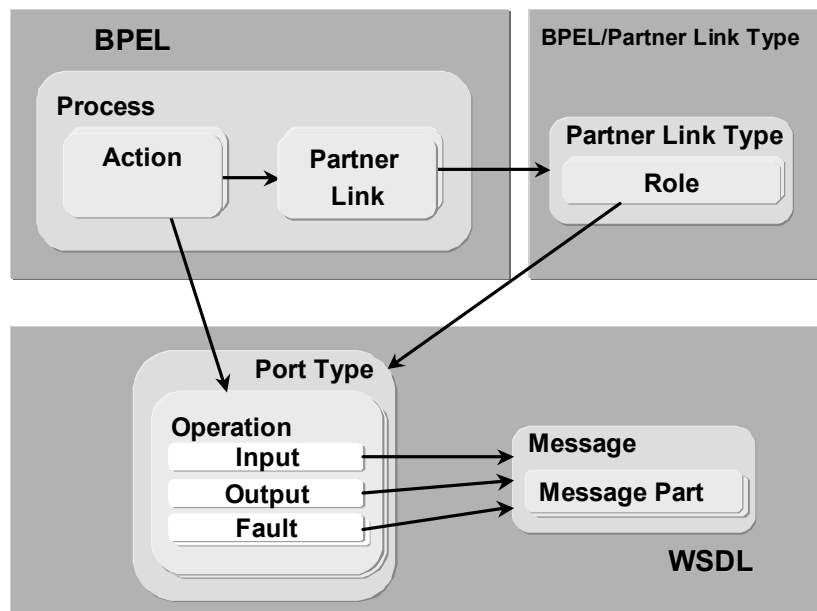
149

150

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

151

152



153

154

Figure 1: The BPEL model and its relationship with WSDL

### 155 2.1.2 Mapping BPEL4WS to UDDI

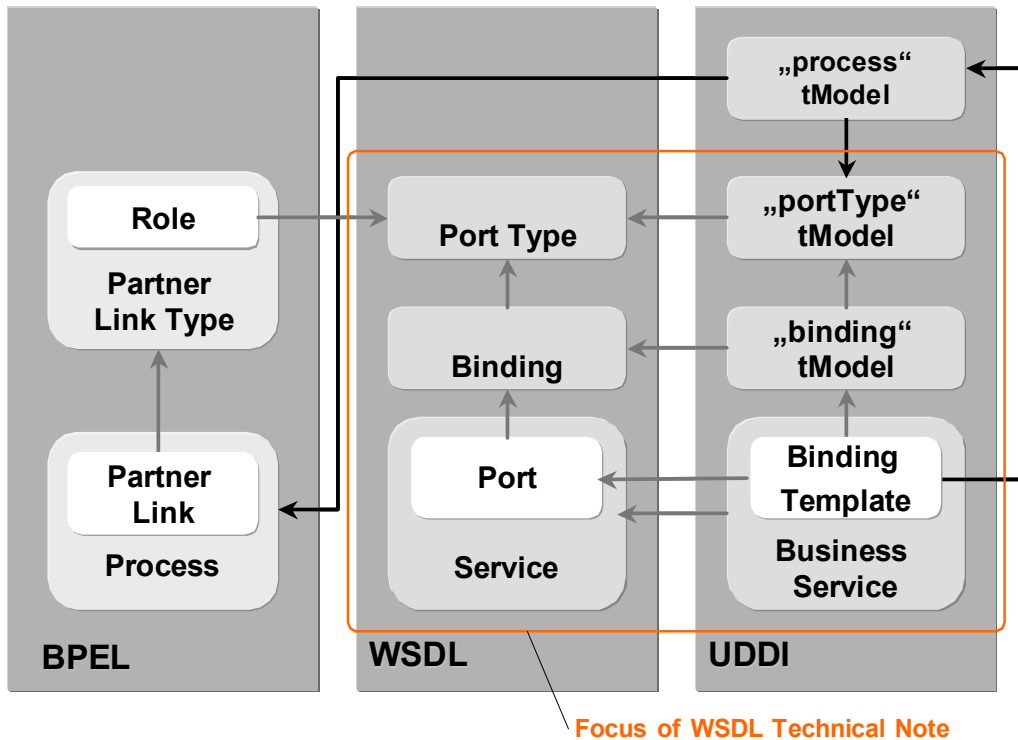
156 BPEL4WS abstract processes are published as separate UDDI tModels. They are named with  
157 the BPEL4WS process name. They are categorized as BPEL4WS process definitions, using a  
158 category system defined in this technical note. Their overviewDoc references an external  
159 BPEL4WS document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

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

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

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

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



174

175

Figure 2: Mapping BPEL to UDDI

---

## 176 3 tModel definitions

### 177 3.1 BPEL Entity Type tModel

#### 178 3.1.1 Design Goals

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

#### 184 3.1.2 Definition

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

#### 191 3.1.2.1 V2 tModel Structure

```
192 <tModel tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f">  
193   <name>uddi.org:bpel:types</name>  
194   <overviewDoc>  
195     <overviewURL>  
196       TBD, should point to this section when the document is  
197 published as a Technical Note by the UDDI TC  
198     </overviewURL>  
199   </overviewDoc>  
200   <categoryBag>  
201     <keyedReference  
202       keyName="uddi-org:categorization:types"  
203       keyValue="categorization"  
204       tModelKey="uuid:clacf26d-9672-4404-9d70-39b756e62ab4"/>  
205     <keyedReference  
206       keyName="uddi-org:categorization:types"  
207       keyValue="unchecked"  
208       tModelKey="uuid:clacf26d-9672-4404-9d70-39b756e62ab4"/>  
209   </categoryBag>  
210 </tModel>
```

#### 211 3.1.2.2 Valid Values

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

214

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



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

### 215 3.1.2.3 Example of Use

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

```
217 <categoryBag>
218   <keyedReference
219     tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"
220     keyName="BPEL Entity type"
221     keyValue="process"/>
222   ...
223 </categoryBag>
```

## 224 3.2 WSDL portType Reference tModel

### 225 3.2.1 Design Goals

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

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

### 230 3.2.2 Definition

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

#### 237 3.2.2.1 V2 tModel Structure

```
238 <tModel tModelKey="uuid:ef2dcc0a-edc8-343d-9913-d2e61777a90c">
239   <name>uddi.org:bpel:wsdlPortTypeReference</name>
240   <description xml:lang="en">
241     This tModel is a category system tModel that can be used to
242     identify a relationship to a portType tModel.
243   </description>
244   <overviewDoc>
245     <overviewURL>
246       TBD
247     </overviewURL>
248   </overviewDoc>
249   <categoryBag>
250     <keyedReference
251       keyName="uddi-org:categorization:types"
252       keyValue="categorization"
253       tModelKey="uuid:clacf26d-9672-4404-9d70-39b756e62ab4"/>
254     <keyedReference
255       keyName="uddi-org:categorization:types"
256       keyValue="checked"
257       tModelKey="uuid:clacf26d-9672-4404-9d70-39b756e62ab4"/>
```

```
258     <keyedReference
259         keyName="uddi-org:categorization:entityKeyValues"
260         keyValue="tModel "
261         tModelKey="uuid:916b87bf-0756-3919-8eae-97dfa325e5a4"/>
262     </categoryBag>
263 </tModel>
```

### 264 3.2.3 Valid Values

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

### 268 3.2.4 Example of Use

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

```
271 <categoryBag>
272     <keyedReference
273         tModelKey="uuid:ef2dcc0a-edc8-343d-9913-d2e61777a90c"
274         keyName="uddi-org:bpel:portType Reference"
275         keyValue="uuid:e8cf1163-8234-4b35-865f-94a7322e40c3"/>
276     ...
277 </categoryBag>
```

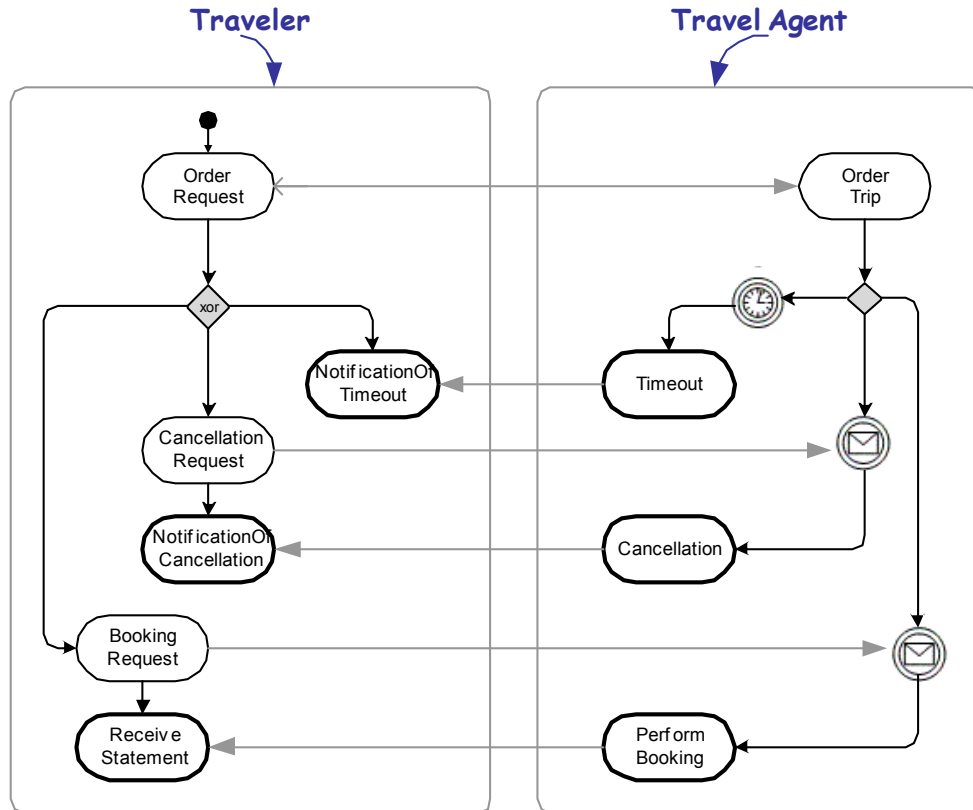
278 Note that the keyValue is a tModelKey, which, if queried for using get\_tModelDetail, would return  
279 the tModel that represents the WSDL portType.

280

## 4 Example

281 This section includes tModels representing a BPEL4WS abstract process, accompanying WSDL  
282 descriptions and UDDI registrations. A Travel Agent example is used for illustration. The example  
283 gives the basic behavior exposed by a Travel Agent service in a Ticket Reservation System.

284 Figure 3 shows the overall process: the Travel Agent interacts with a Customer (a traveler)  
285 according to a very simplified choreography: a customer can order a trip with the travel agent,  
286 and later may either cancel or confirm already reserved trip.



287

288

Figure 3: The Ticket Reservation scenario

### 4.1 BPEL4WS process and WSDL portTypes

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

293

294

295

296

297

298

299

300

301

302

303

304

```

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

```

```

305 <portType name="InterfaceOfTravelAgent">
306   <operation name="OrderTrip">
307     <input message="orderRequest"/>
308     <output message="orderAcknowledgement"/>
309   </operation>
310
311   <operation name="CancelReservation">
312     <input message="cancellationRequest"/>
313   </operation>
314
315   <operation name="PerformBooking">
316     <input message="bookingRequest"/>
317     <output message="bookingConfirmation"/>
318   </operation>
319 </portType>
320
321 <portType name="InterfaceOfCustomer">
322   <operation name="NotificationOfCancellation">
323     <input message="cancellationResponse"/>
324   </operation>
325
326   <operation name="NotificationOfTimeout">
327     <input message="timeoutMsg"/>
328   </operation>
329
330   <operation name="ReceiveStatement">
331     <input message="statement"/>
332   </operation>
333 </portType>
334
335 <!--partner link type definitions -->
336
337 <plnk:partnerLinkType name="TravelAgentService">
338   <plnk:role name="TravelAgent">
339     <plnk:portType name="InterfaceOfTravelAgent"/>
340   </plnk:role>
341   <plnk:role name="Customer">
342     <plnk:portType name="InterfaceOfCustomer"/>
343   </plnk:role>
344 </plnk:partnerLinkType>
345
346 <!--definition of properties -->
347
348 <bpws:property name="reservationID" type="xsd:string"/>
349
350 <!-- property aliases are omitted-->
351 </definitions>
352

```

353

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

```

356 <process name = "ReservationAndBookingTickets"
357   targetNamespace="http://example.com/travelagent"
358   xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
359   xmlns:taw="http://example.com/travelagent/wSDL"
360   abstractProcess="yes">
361
362   <partnerLinks>
363     <partnerLink name="TravelAgency"
364       partnerLinkType="taw:TravelAgencyService"
365       partnerRole="Customer"
366       myRole="TravelAgent"/>
367   </partnerLinks>
368
369   <correlationSets>
370     <correlationSet name="reservationCorrelation"
371       properties="taw:reservationID"/>
372   </correlationSets>

```

373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427

```
<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>
```

428

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

## 433 4.2 UDDI V2 Registrations

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

## 436 4.2.1 WSDL portTypes

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

```
439 <tModel tModelKey="uuid:a1..." >  
440   <name>InterfaceOfTravelAgent</name>  
441   <overviewDoc>  
442     <overviewURL>http://location/travelagent.wsdl</overviewURL>  
443   </overviewDoc>  
444   <categoryBag>  
445     <keyedReference  
446       tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"  
447       keyName="uddi-org:xml:namespace"  
448       keyValue="http://example.com/travelagent/wsdl" />  
449     <keyedReference  
450       tModelKey="uuid:6e090afa-33e5-36eb-81b7-1ca18373f457"  
451       keyName="uddi-org:wsdl:types"  
452       keyValue="portType" />  
453   </categoryBag>  
454 </tModel>
```

455

```
456 <tModel tModelKey="uuid:a2..." >  
457   <name>InterfaceOfCustomer</name>  
458   <overviewDoc>  
459     <overviewURL>http://location/customer.wsdl</overviewURL>  
460   </overviewDoc>  
461   <categoryBag>  
462     <keyedReference  
463       tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"  
464       keyName="uddi-org:xml:namespace"  
465       keyValue="http://example.com/travelagent/wsdl" />  
466     <keyedReference  
467       tModelKey="uuid:6e090afa-33e5-36eb-81b7-1ca18373f457"  
468       keyName="uddi-org:wsdl:types"  
469       keyValue="portType" />  
470   </categoryBag>  
471 </tModel>
```

## 472 4.2.2 BPEL4WS process

```
473 <tModel tModelKey="uuid:b1..." >  
474   <name>ReservationAndBookingTickets</name>  
475   <overviewDoc>  
476     <overviewURL>http://location/reservation.bpel</overviewURL>  
477   </overviewDoc>  
478   <categoryBag>  
479     <keyedReference  
480       tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"  
481       keyName="uddi-org:xml:namespace"  
482       keyValue="http://example.com/travelagent" />  
483     <keyedReference  
484       tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"  
485       keyName="uddi-org:bpel:types"  
486       keyValue="process" />  
487     <keyedReference  
488       tModelKey="uuid:ef2dcc0a-edc8-343d-9913-d2e61777a90c"  
489       keyName="uddi-org:bpel:portTypeReference"  
490       keyValue="uuid:a1..." />  
491     <keyedReference
```

```

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

```

## 497 4.2.3 WSDL port

```

498 <businessService
499   serviceKey="dl..."
500   businessKey="el...">
501   ...
502   <bindingTemplates>
503     <bindingTemplate
504       bindingKey="cl..."
505       serviceKey="dl...">
506       <accessPoint URLType="http">
507         http://location/sample
508       </accessPoint>
509       <tModelInstanceDetails>
510         <tModelInstanceInfo
511           tModelKey="uuid:el...">
512           <description xml:lang="en">
513             The wsdl:binding that this wsdl:port implements.
514             The instanceParms specifies the port local name.
515           </description>
516           <instanceDetails>
517             <instanceParms>TravelAgentPort</instanceParms>
518           </instanceDetails>
519         </tModelInstanceInfo>
520         <tModelInstanceInfo
521           tModelKey="uuid:a1...">
522           <description xml:lang="en">
523             The wsdl:portType that this wsdl:port implements.
524           </description>
525         </tModelInstanceInfo>
526         <tModelInstanceInfo
527           tModelKey="uuid:b1...">
528           <description xml:lang="en">
529             The bpel:process this wsdl:port supports.
530           </description>
531         </tModelInstanceInfo>
532       </tModelInstanceDetails>
533     </bindingTemplate>
534   </bindingTemplates>
535 </businessService>

```

## 536 4.3 Sample V2 Queries

### 537 4.3.1 Find tModel for process name

538 Find the process tModel for ReservationAndBookingTickets in the namespace  
539 <http://example.com/travelagent>.

```

540 <find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
541   <name>ReservationAndBookingTickets</name>
542   <categoryBag>
543     <keyedReference
544       tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"
545       keyValue="process"/>

```

```

546         <keyedReference
547             tModelKey="uuid:d01987d1-ab2e-3013-9be2-2a66eb99d824"
548             keyValue="http://example.com/travelagent"/>
549     </categoryBag>
550 </find_tModel>

```

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

### 552 4.3.2 Find processes for portTypes

553 Find all processes that use the InterfaceOfTravelAgent portType.

```

554 <find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
555     <categoryBag>
556         <keyedReference
557             tModelKey="uuid:e8d75f6c-3f24-3b8d-97fd-f168e424056f"
558             keyValue="process"/>
559         <keyedReference
560             tModelKey="uuid:ef2dcc0a-edc8-343d-9913-d2e61777a90c"
561             keyValue="a1..."/>
562     </categoryBag>
563 </find_tModel>

```

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

### 565 4.3.3 Find portTypes for process

566 Find all portTypes used in the ReservationAndBookingTickets process.

```

567 <get_tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2">
568     <tModelKey>uuid:b1...</tModelKey>
569 </get_tModelDetail>

```

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

```

575 <get_tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2">
576     <tModelKey>uuid:a1...</tModelKey>
577     <tModelKey>uuid:a2...</tModelKey>
578 </get_tModelDetail>

```

### 579 4.3.4 Find implementations for process

580 Find all implementations of ReservationAndBookingTickets.

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

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

```

589 <find_service generic="2.0" xmlns="urn:uddi-org:api_v2">
590     <tModelBag>
591         <tModelKey>uuid:b1...</tModelKey>
592     </tModelBag>

```



593 </find\_binding>

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

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

```
596 <find_binding serviceKey="d1..." generic="2.0" xmlns="urn:uddi-org:api_v2">
597   <tModelBag>
598     <tModelKey>uuid:b1...</tModelKey>
599   </tModelBag>
600 </find_binding>
```

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

## 602 4.4 UDDI V3 Registrations

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

### 605 4.4.1 WSDL portTypes

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

```
611 <tModel tModelKey="uddi:TravelAgent.com:TravelAgentInterface_portType" >
612   <name>InterfaceOfTravelAgent</name>
613   <overviewDoc>
614     <overviewURL>http://location/travelagent.wsdl</overviewURL>
615   </overviewDoc>
616   <categoryBag>
617     <keyedReference
618       tModelKey="uddi:uddi.org:xml:namespace"
619       keyName="uddi-org:xml:namespace"
620       keyValue="http://example.com/travelagent/wsdl" />
621     <keyedReference
622       tModelKey="uddi:uddi.org:wsdl:types"
623       keyName="uddi-org:wsdl:types"
624       keyValue="portType" />
625   </categoryBag>
626 </tModel>
```

627

```
628 <tModel tModelKey="uddi:TravelAgent.com:CustomerInterface_portType" >
629   <name>InterfaceOfCustomer</name>
630   <overviewDoc>
631     <overviewURL>http://location/customer.wsdl</overviewURL>
632   </overviewDoc>
633   <categoryBag>
634     <keyedReference
635       tModelKey="uddi:uddi.org:xml:namespace"
636       keyName="uddi-org:xml:namespace"
637       keyValue="http://example.com/travelagent/wsdl" />
638     <keyedReference
639       tModelKey="uddi:uddi.org:wsdl:types"
640       keyName="uddi-org:wsdl:types"
641       keyValue="portType" />
642   </categoryBag>
643 </tModel>
```

## 644 4.4.2 BPEL4WS process

```
645 <tModel
646 tModelKey="uddi:TravelAgent.com:ReservationAndBookingTicketsProcess" >
647   <name>ReservationAndBookingTickets</name>
648   <overviewDoc>
649     <overviewURL>http://location/reservation.bpel</overviewURL>
650   </overviewDoc>
651   <categoryBag>
652     <keyedReference
653       tModelKey="uddi:uddi.org:xml:namespace"
654       keyName="uddi-org:xml:namespace"
655       keyValue="http://example.com/travelagent" />
656     <keyedReference
657       tModelKey="uddi:uddi.org:bpel:types"
658       keyName="uddi-org:bpel:types"
659       keyValue="process" />
660     <keyedReference
661       tModelKey="uddi:uddi.org:bpel:portTypeReference"
662       keyName="uddi-org:bpel:portTypeReference"
663       keyValue="uddi:TravelAgent.com:TravelAgentInterface_portType"
664   />
665     <keyedReference
666       tModelKey="uddi:uddi.org:bpel:portTypeReference"
667       keyName="uddi-org:bpel:portTypeReference"
668       keyValue="UDDI:TravelAgent.com:CustomerInterface" />
669   </categoryBag>
670 </tModel>
```

## 671 4.4.3 WSDL port

```
672 <businessService
673   serviceKey="uddi:TravelAgent.com:service1"
674   businessKey="uddi:TravelAgent.com:StoreFront">
675   ...
676   <bindingTemplates>
677     <bindingTemplate
678       bindingKey="uddi:TravelAgent.com:TravelAgentPort"
679       serviceKey="uddi:TravelAgent.com:service1">
680       <accessPoint URLType="http">
681         http://location/sample
682       </accessPoint>
683       <tModelInstanceDetails>
684         <tModelInstanceInfo
685           tModelKey="uddi:TravelAgent.com:StoreFront">
686           <description xml:lang="en">
687             The wsdl:binding that this wsdl:port implements.
688             The instanceParms specifies the port local name.
689           </description>
690           <instanceDetails>
691             <instanceParms>TravelAgentPort</instanceParms>
692           </instanceDetails>
693         </tModelInstanceInfo>
694         <tModelInstanceInfo
695           tModelKey="uddi:TravelAgent.com:TravelAgentInterface_portType">
696           <description xml:lang="en">
697             The wsdl:portType that this wsdl:port implements.
698           </description>
699         </tModelInstanceInfo>
700         <tModelInstanceInfo
701           tModelKey=
702
```

```

703
704 "uddi:TravelAgent.com:ReservationAndBookingTicketsProcess ">
705     <description xml:lang="en">
706         The bpel:process this wsdl:port supports.
707     </description>
708     </tModelInstanceInfo>
709 </tModelInstanceDetails>
710 </bindingTemplate>
711 </bindingTemplates>
712 </businessService>

```

## 713 4.5 Sample V3 Queries

### 714 4.5.1 Find tModel for process name

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

```

717 <find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
718   <name>ReservationAndBookingTickets</name>
719   <categoryBag>
720     <keyedReference
721       tModelKey="uddi:uddi.org:bpel:types"
722       keyValue="process"/>
723     <keyedReference
724       tModelKey="uddi:uddi.org:xml:namespace"
725       keyValue="http://example.com/travelagent"/>
726   </categoryBag>
727 </find_tModel>

```

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

### 730 4.5.2 Find processes for portTypes

731 Find all processes that use the InterfaceOfTravelAgent portType.

```

732 <find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
733   <categoryBag>
734     <keyedReference
735       tModelKey="uddi:uddi.org:bpel:types"
736       keyValue="process"/>
737     <keyedReference
738       tModelKey="uddi:uddi.org:bpel:portTypeReference"
739       keyValue="uddi:TravelAgent.com:TravelAgentInterface_portType"/>
740   </categoryBag>
741 </find_tModel>

```

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

### 744 4.5.3 Find portTypes for process

745 Find all portTypes used in the ReservationAndBookingTickets process.

```

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

```

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

```
756 <get_tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2">  
757 <tModelKey>uddi:TravelAgent.com:TravelAgentInterface_portType</tModelKey>  
758 <tModelKey>uddi:TravelAgent.com:CustomerInterface_portType</tModelKey>  
759 </get_tModelDetail>
```

#### 760 4.5.4 Find implementations for process

761 Find all implementations of ReservationAndBookingTickets.

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

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

```
770 <find_service generic="2.0" xmlns="urn:uddi-org:api_v2">  
771 <tModelBag>  
772 <tModelKey>uddi:TravelAgent.com:ReservationAndBookingTicketsProcess  
773 </tModelKey>  
774 </tModelBag>  
775 </find_binding>
```

776 This should return the serviceKey "uddi:TravelAgent.com:service1".

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

```
778 <find_binding serviceKey="uddi:TravelAgent.com:service1" generic="2.0"  
779 xmlns="urn:uddi-org:api_v2">  
780 <tModelBag>  
781 <tModelKey>uddi:TravelAgent.com:ReservationAndBookingTicketsProcess  
782 </tModelKey>  
783 </tModelBag>  
784 </find_binding>
```

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

786

787

---

## 5 References

788

### 5.1 Normative

- 789       **[BPEL4WS]**       T. Andrews, F. Curbera, H. Dholakia, Y. Golland, J. Klein, F. Leymann, K.  
790                           Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana,  
791                           *Business Process Execution Language for Web Services Version 1.1*,  
792                           <http://ifr.sap.com/bpel4ws>, May 2003.
- 793       **[RFC2119]**       S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,  
794                           <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 795       **[WSDL11]**       E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *Web*  
796                           *Services Description Language (WSDL) 1.1*,  
797                           <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>, W3C Note, March  
798                           2001.
- 799       **[WSDLBP]**       J. Colgrave, K. Januszewski, *Using WSDL in a UDDI Registry, Version*  
800                           *1.08*, [http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-](http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.htm)  
801                           [tc-bp-using-wsdl-v108-20021110.htm](http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.htm), OASIS UDDI TC Best Practice,  
802                           November 2002.
- 803       **[WSDLTN]**       J. Colgrave, K. Januszewski, *Using WSDL in a UDDI Registry, Version*  
804                           *2.0*, [http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-](http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm)  
805                           [tc-tn-wsdl-v2.htm](http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm), OASIS UDDI TC Technical Note, June 2003.  
806

---

807 **Appendix A. Acknowledgments**

808 The following individuals were members of the committee during the development of this  
809 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

---

812

## Appendix C. Notices

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

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

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

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

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

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