UDDI Specifications TC

UDDI as the registry for ebXML Components

Technical Note

Document identifier:
uddi-spec-tc-tn-uddi-ebxml

Current version:
http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-uddi-ebxml-20030920.htm

Latest version:
http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-uddi-ebxml.htm

Author:
Keisuke Kibakura, Fujitsu

Editors:
Luc Clément, Microsoft
Daniel Feygin, Unitspace
Tony Rogers, Computer Associates

Contributors:
Tom Bellwood, IBM
Jacques Durand, Fujitsu
Sam Lee, Oracle
Joel Munter, Intel
Dale Moberg, Cyclone Commerce
Claus von Riegen, SAP
Alok Srivastava, Oracle
Max Voskob, MSI Business Solutions

Abstract:
This document describes the way to model ebXML-based services and ebXML components such as CPP and BPSS, and provides a practical guidance on how to use a UDDI registry as the registry for ebXML components.

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 intellectual property claims 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

This UDDI Spec Technical Committee Technical Note (TN) provides technical guidance on how to use UDDI registries within the ebXML framework of B2B services. Specifically, it addresses the issues related to enabling automated discovery of ebXML framework components, such as Collaboration Protocol Profile and Business Process Specification Schema, using UDDI.

By adopting the technical guidance of this TN, users will enable trading partners and their Web services and ebXML infrastructures to interact using UDDI as a common registry.

1.1 Problem Statement

Multiple consortia have initiated pilot projects using the ebXML framework for business-to-business transactions, while corporations have also begun adopting ebXML technologies for internal use. At the same time Web service technologies, which have significant momentum due to unprecedented industry support, are also being rolled out. UDDI can play a major role in enabling trading partners and their Web services and ebXML infrastructures to interact using UDDI as a common registry. This is the focus of this Technical Note.

In addition to being a universal technology for publication and discovery of service metadata, UDDI also enables discovery of ebXML framework components such as Collaboration Protocol Profile and Business Process Specification Schema. This capability can help enable interoperability among trading partners that use UDDI and ebXML framework components. However, a prescribed methodology of modeling services and components which are conformant to ebXML specifications is required to make interoperable solutions possible.

1.2 Goals

This note provides technical guidance:

- to model an ebXML-based service;
- to register an ebXML-based service and ebXML components in a UDDI registry; and
- to query a UDDI registry for an ebXML-based service and ebXML components.

1.3 Non-goals

This note does *not* intend:

- to merge UDDI and ebXML registry technologies;
- to substitute an ebXML registry with a UDDI registry; or
- to migrate entity data from an ebXML Registry to a UDDI registry (or vice versa).

1.4 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

ebXML

ebXML is a common XML-based e-business framework which consists of some distinct technologies.

ebXML Registry

ebXML Registry is intended to provide registry and repository services in the ebXML framework. [ebRR1][ebRR2]

ebXML Message Services

ebXML Messaging Services provides a method for conducting e-business transactions within the framework of ebXML technologies. It extends the SOAP protocol to address specific business-to-business transaction requirements such as authentication, reliability and non-repudiation. **[ebMS1][ebMS2]**

ebXML Collaboration Protocol Profile and Agreement (CPPA)

This is a specification which defines CPP and CPA. **[ebCPPA1][ebCPPA2]**

ebXML Collaboration Protocol Profile (CPP)

A CPP defines one business partner's technical capabilities to engage in electronic business collaborations with other partners by exchanging electronic messages. These capabilities include business processes, document formats, and technical communication parameters required to communicate with a trading partner. Before starting to do business with a partner via ebXML, trading partners create and exchange their CPPs, recognize the other's technical capability, negotiate for common ground, and build a CPA (Collaboration Protocol Agreement).

ebXML Collaboration Protocol Agreement (CPA)

A CPA documents the technical agreement between two (or more) partners to engage in electronic business collaboration. It may be formed from two (or more) CPPs, or may be created from a CPA template.

ebXML CPA template

CPA template is a trading partner's "fill in the blanks" proposal to a prospective trading partner. It has place-holding values that are intended to be replaced by the actual values when a CPA is derived. CPA template may be used to form a CPA as the alternative method to merging two CPPs.

ebXML Business Process Specification Schema (BPSS)

BPSS provides a machine readable description of a business process. **[ebBPSS]**

2.2 Technical Note Behavior

2.2.1 Reference Scenario

What follows is a typical use case which this Technical Note will use as its reference scenario:

- a fictitious consortium, named "ABC Consortium" creates a standardized business process for a given industry vertical and describes it as an ebXML BPSS;
- fictitious seller Company A and buyer Company B each create their own CPPs;
- Company A implements and registers an ebXML-based service S_a which is conformant to the standard process defined by ABC Consortium.

Using the guidance provided for by this Technical Note, these partners model their ebXML components and interact as follows to enable their business processes:

1. ABC Consortium creates a tModel to represent the standard business process and publishes it.
2. Company A registers the location of its CPP as a business service.
3. Company A registers a service S_a.
4. Company B finds A's service S_a using a UDDI registry.
5. Company B locates A's CPP using a UDDI registry.

Figure 1 shows these five steps.

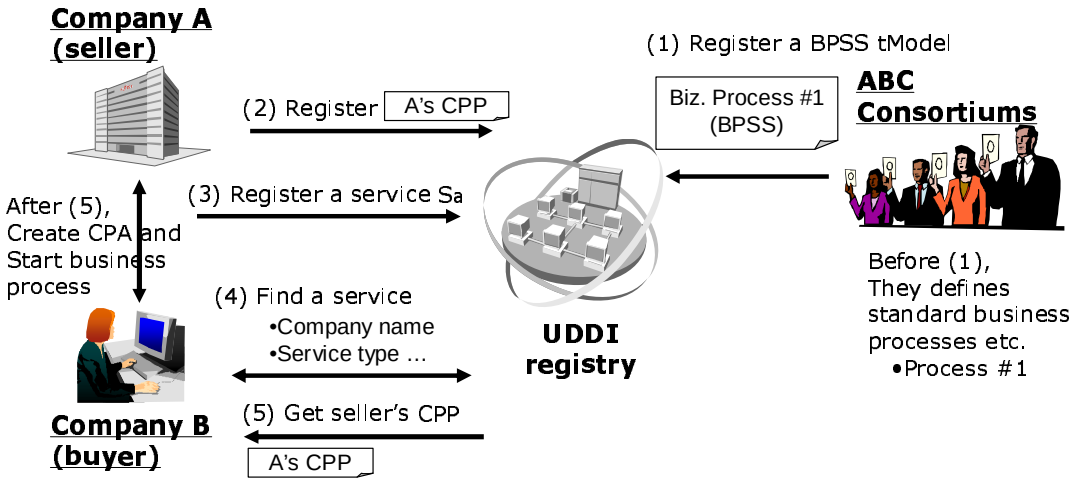

Figure 1: Reference scenario

These five steps show how Company B obtains Company A's CPP. Company B can then begin to negotiate and build a CPA, allowing it to do business with Company A. What happens after the discovery is complete is specific to ebXML and is out of this document's scope.

A CPA provides run-time information required for trading partners' business collaboration software components. A CPA is generally only accessible to the parties that created it and is not disclosed to any third party, since it implies business partnership between the parties to a CPA, which can be regarded as a potential trade secret. Therefore this TN does not model any recommended registration of the CPA in UDDI.

The following sections provide specific modeling recommendations covering this scenario.

2.2.2 Modeling Outline and Rationale

ebXML services and components must be modeled to match UDDI data structures. The bases for modeling are as follows:

- A company that provides ebXML-based services is modeled as a businessEntity.
- An ebXML-based service is modeled as a businessService.
- A tModel is created to represent each ebXML specification.
- A company's CPP is modeled as a businessService belonging to the businessEntity whose capabilities it describes. This fits the one-to-many cardinality of company-to-CPP relationships, which are never shared with external parties, just as a businessEntity's contained businessServices.
- A business process definition (BPSS) is modeled as a tModel. A BPSS instance can be regarded as one of technical characteristics of a service interface, and can be used by multiple organizations.

This Technical Note defines three types of tModels:

Specifications Taxonomy tModel

This taxonomy is used to categorize each ebXML specification represented by a Common ebXML tModel (see below) to aid in their discovery.

Common ebXML tModels

These tModels refer to specific versions of ebXML specifications/concepts, such as CPP v1.0, BPSS v1.01, Message Service v2.0, and so on.

Proprietary tModels

These tModels refer to specific instances or implementations of ebXML specifications. For instance, a particular business process, described in an ebXML BPSS, is an instance of BPSS, and it can be defined as a proprietary tModel.

The models described in this Technical Note apply equally to both UDDI v2 and v3 registries, though all the examples are written in terms of UDDI v2.

2.2.3 ebXML Specifications Taxonomy tModel

ebXML is a framework that consists of several distinct technical specifications, each of which must be represented by tModels in a UDDI registry. UDDI tModels are usually tagged with categorizations denoting information that facilitates their discovery. That is why before defining tModels representing ebXML specifications, we define an ebXML Specifications Taxonomy tModel, which is used to categorize these tModels (described in the following section).

ebXML Specification Taxonomy tModel:

tModel Name:	***ebxml-org:specifications***
tModel Description:	ebXML Specifications Taxonomy
tModel UDDI Key (V3):	uddi:ebxml.org:specifications
Derived V1, V2 format Key:	uuid:da52cf72-2cb2-39a9-ad1e-577e66d8a6f6
Categorization:	categorization
Checked:	No

```
<tModel tModelKey="uuid:da52cf72-2cb2-39a9-ad1e-577e66d8a6f6">
  <name>ebxml-org:specifications</name>
```

```
  <description xml:lang="en">ebXML Specifications Taxonomy</description>
  <overviewDoc>
    <overviewURL>http://www.oasis-
open.org/committees/tc_home.php?wg_abbrev=ebxml-jc/</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
       tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
       keyName="uddi-org:types"
       keyValue="categorization" />
    <keyedReference
       tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
       keyName="uddi-org:types"
       keyValue="unchecked" />
  </categoryBag>
</tModel>
```

The following values are defined for this ebXML Specifications taxonomy. These values are useful for classifying ebXML-related tModels and are helpful for others who want to find those tModels.

ebXML:CPPA	ebXML Collaboration Protocol Profile and Agreement
ebXML:MS	ebXML Message Service
ebXML:BPSS	ebXML Business Process Specification Schema

Please note that this tModel is a categorization system for ebXML framework specifications and does not represent ebXML specifications themselves.

2.2.4 Common ebXML tModels

In UDDI, a service interface (which is called a binding template in UDDI) is associated with tModels, which refer to the technical specification of the interface. Those tModels represent service type information. This collection of tModels is sometimes referred to as the technical fingerprint of the service interface.

In the case of ebXML specifications, Message Service, CPP, CPA template[1] and BPSS all convey relevant service type information. It can be captured in tModels, which we will refer to as common ebXML tModels. "Common" means that these tModels are relied on by all users who use UDDI as the registry for ebXML services and components. Since these tModels are categorized by ebXML Specifications Taxonomy (ebxml-org:specifications) defined in the previous section, they are easy to find.

ebXML Message Service v1.0 tModel

tModel Name:	***ebxml-org:MessageService:v1_0***
tModel Description:	ebXML Message Service v1.0
tModel UDDI Key (V3):	uddi:ebxml.org:messageservice:v1.0
Derived V1, V2 format Key:	uuid:c8692873-1842-3b32-b980-8fa6d16676d2
Categorization:	specification

[1] This tModel is not used in the scenario, but is used in its variation. See Section 2.2.7.

```
<tModel tModelKey="uuid:c8692873-1842-3b32-b980-8fa6d16676d2">
  <name>ebxml-org:MessageService:v1_0</name>
  <description xml:lang="en">ebXML Message Service v1.0</description>
  <overviewDoc>
    <overviewURL>http://www.ebxml.org/specs/ebMS.pdf</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
      tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
      keyName="uddi-org:types"
      keyValue="specification" />
    <keyedReference
      tModelKey="uuid:da52cf72-2cb2-39a9-ad1e-577e66d8a6f6"
      keyName="ebXML Message Service"
      keyValue="ebXML:MS" />           <!—- Message Service spec -->
  </categoryBag>
</tModel>
```

ebXML Message Service v2.0 tModel

tModel Name:	***ebxml-org:MessageService:v2_0***
tModel Description:	ebXML Message Service v2.0
tModel UDDI Key (V3):	uddi:ebxml.org:messageservice:v2.0
Derived V1, V2 format Key:	uuid:9a3b93be-515e-34c7-90c7-b05cdfdba8c3
Categorization:	specification

```
<tModel tModelKey="uuid:9a3b93be-515e-34c7-90c7-b05cdfdba8c3">
  <name>ebxml-org:MessageService:v2_0</name>
  <description xml:lang="en">ebXML Message Service v2.0</description>
  <overviewDoc>
    <overviewURL>http://www.oasis-open.org/committees/ebxml-
msg/documents/ebMS_v2_0.pdf</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
      tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
      keyName="uddi-org:types"
      keyValue="specification" />
    <keyedReference
      tModelKey="uuid:da52cf72-2cb2-39a9-ad1e-577e66d8a6f6"
      keyName="ebXML Message Service"
      keyValue="ebXML:MS" />           <!—- Message Service spec -->
  </categoryBag>
</tModel>
```

ebXML CPP v1.0 tModel

tModel Name:	***ebxml-org:CollaborationProtocolProfile:v1_0***
tModel Description:	ebXML Collaboration Protocol Profile v1.0
tModel UDDI Key (V3):	uddi:ebxml.org:collaborationprotocolprofile:v1.0
Derived V1, V2 format Key:	uuid:e3f3df4f-b221-33b4-a3ff-17b21410c565
Categorization:	specification

```
<tModel tModelKey="uuid:e3f3df4f-b221-33b4-a3ff-17b21410c565">
  <name>ebxml-org:CollaborationProtocolProfile:v1_0</name>
  <description xml:lang="en">ebXML Collaboration Protocol Profile
v1.0</description>
  <overviewDoc>
    <overviewURL>http://www.ebxml.org/specs/ebCPP.pdf</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
       tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
       keyName="uddi-org:types"
       keyValue="specification" />
    <keyedReference
       tModelKey="uuid:da52cf72-2cb2-39a9-ad1e-577e66d8a6f6"
       keyName="ebXML Collaboration Protocol Profile and Agreement"
       keyValue="ebXML:CPPA" />      <!—- This belongs to CPPA spec -->
  </categoryBag>
</tModel>
```

ebXML CPP v2.0 tModel

tModel Name:	***ebxml-org:CollaborationProtocolProfile:v2_0***
tModel Description:	ebXML Collaboration Protocol Profile v2.0
tModel UDDI Key (V3):	uddi:ebxml.org:collaborationprotocolprofile:v2.0
Derived V1, V2 format Key:	uuid:43ed4af4-eacf-3b20-95d1-c7c197f5d9d0
Categorization:	specification

```
<tModel tModelKey="uuid:43ed4af4-eacf-3b20-95d1-c7c197f5d9d0">
  <name>ebxml-org:CollaborationProtocolProfile:v2_0</name>
  <description xml:lang="en">ebXML Collaboration Protocol Profile
v2.0</description>
  <overviewDoc>
    <overviewURL>http://www.oasis-open.org/committees/ebxml-
cppa/documents/ebCPP-2_0.pdf</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
       tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
       keyName="uddi-org:types"
       keyValue="specification" />
    <keyedReference
       tModelKey="uuid:da52cf72-2cb2-39a9-ad1e-577e66d8a6f6"
       keyName="ebXML Collaboration Protocol Profile and Agreement"
       keyValue="ebXML:CPPA" />     <!—- This belongs to CPPA spec -->
  </categoryBag>
</tModel>
```

ebXML CPA v1.0 Template tModel

tModel Name:	***ebxml-org:CollaborationProtocolAgreement:v1_0:Template***
tModel Description:	ebXML Collaboration Protocol Agreement v1.0 Template
tModel UDDI Key (V3):	uddi:ebxml.org:collaborationprotocolagreement:v1.0:template

Derived V1, V2 format Key: uuid:5ab4e3af-2e67-3a4f-b9b7-92a436be8f43

Categorization: xmlSpec

```
<tModel tModelKey="uuid:5ab4e3af-2e67-3a4f-b9b7-92a436be8f43">
  <name>ebxml-org:CollaborationProtocolAgreement:v1_0:Template</name>
  <description xml:lang="en">
     ebXML Collaboration Protocol Agreement v1.0 Template
  </description>
  <overviewDoc>
    <overviewURL>http://www.ebxml.org/specs/ebCPP.pdf</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
       tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
       keyName="uddi-org:types"
       keyValue="xmlSpec" />
    <keyedReference
       tModelKey="uuid:da52cf72-2cb2-39a9-ad1e-577e66d8a6f6"
       keyName="ebXML Collaboration Protocol Profile and Agreement"
       keyValue="ebXML:CPPA" />      <!—- This belongs to CPPA spec -->
  </categoryBag>
</tModel>
```

ebXML CPA v2.0 Template tModel

tModel Name: ***ebxml-org:CollaborationProtocolAgreement:v2_0:Template***

tModel Description: ebXML Collaboration Protocol Agreement v2.0 Template

tModel UDDI Key (V3): uddi:ebxml.org:collaborationprotocolagreement:v2.0:template

Derived V1, V2 format Key: uuid:86c6fcb3-8c73-33a7-8635-38438b76aee7

Categorization: xmlSpec

```
<tModel tModelKey="uuid:86c6fcb3-8c73-33a7-8635-38438b76aee7">
  <name>ebxml-org:CollaborationProtocolAgreement:v2_0:Template</name>
  <description xml:lang="en">
     ebXML Collaboration Protocol Agreement v2.0 Template
 </description>
  <overviewDoc>
    <overviewURL>http://www.oasis-open.org/committees/ebxml-
cppa/documents/ebCPP-2_0.pdf</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
       tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
       keyName="uddi-org:types"
       keyValue="xmlSpec" />
    <keyedReference
       tModelKey="uuid:da52cf72-2cb2-39a9-ad1e-577e66d8a6f6"
       keyName="ebXML Collaboration Protocol Profile and Agreement"
       keyValue="ebXML:CPPA" />     <!—- This belongs to CPPA spec -->
  </categoryBag>
</tModel>
```

ebXML Business Process Specification Schema v1.01 tModel

tModel Name:	***ebxml-org:BusinessProcessSpecificationSchema:v1_01***
tModel Description:	ebXML Business Process Specification Schema v1.01
tModel UDDI Key (V3):	uddi:ebxml.org:businessprocessspecificationschema:v1.01
Derived V1, V2 format Key:	uuid:152d149b-6cdc-3465-9fa1-b914a3b7cf38
Categorization:	specification

```
<tModel tModelKey="uuid:152d149b-6cdc-3465-9fa1-b914a3b7cf38">
  <name>ebxml-org:BusinessProcessSpecificationSchema:v1_01</name>
  <description xml:lang="en">ebXML Business Process Specification Schema
v1.01</description>
  <overviewDoc>
    <overviewURL>http://www.ebxml.org/specs/ebBPSS.pdf</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
       tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
       keyName="uddi-org:types"
       keyValue="specification" />
    <keyedReference
       tModelKey="uuid:da52cf72-2cb2-39a9-ad1e-577e66d8a6f6"
       keyName="ebXML Business Process Specification Schema"
       keyValue="ebXML:BPSS" />      <!—- This belongs to BPSS spec -->
  </categoryBag>
</tModel>
```

2.2.5 Registering ebXML Services

The tModels defined so far enable us to model and register ebXML-based services in a UDDI registry. In accordance with the scenario presented in section 2.2.1, we show the details of each step one by one.

A consortium registers a tModel of a BPSS instance

In this section, the step below is shown:

1. ABC Consortium creates a tModel to represent the standard business process and publishes it.

It is natural to regard a business process definition as one of technical characteristics of a service interface, since it describes business choreography which the service follows. Every registered businessService conformant to an ebXML BPSS description should be associated with a tModel of the corresponding BPSS instance.

In the reference scenario, a fictitious consortium, ABC Consortium, defines a standard business process (here, we call it “ABC Standard Process #1”), which is widely used by the consortium members. The consortium describes it using a BPSS instance.

Assuming that the BPSS instance is located at “http://abc.org/process_1.bpss”, we define the tModel like below:

tModel Name:	***abc-org:StandardProcess:1***
tModel Description:	ABC Consortium’s Standard Process #1

tModel UDDI Key[2]: uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxbp1

Categorization: xmlSpec

```
<tModel>
  <name>abc-org:StandardProcess:1</name>
  <description xml:lang="en"> ABC's standard process #1</description>
  <overviewDoc>
    <overviewURL>http://abc.org/process_1.bpss</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
      tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
      keyName="uddi-org:types"
      keyValue="xmlSpec" />
    <keyedReference
      tModelKey="uuid:152d149b-6cdc-3465-9fa1-b914a3b7cf38"
      keyName="ebXML Business Process Specification Schema v1.01"
      keyValue="ebxml-org:BusinessProcessSpecificationSchema:v1_01" />
  </categoryBag>
</tModel>
```

ABC Consortium registers this tModel using `save_tModel`. Later, it is referenced from a service whose choreography is conformant to this BPSS instance.

Company A registers CPP

In this section, the step below is shown:

2. Company A registers the location of its CPP as a business service.

In UDDI, all business services must be associated with a business entity. A company must first create a businessEntity if it decides to register an ebXML service. This Technical Note recommends that the location of the CPP should be registered as an accessPoint in a businessService. Assuming that Company A's CPP is located at "http://a.com/a.cpp"[3], its businessEntity and businessService are as follows:

```
<businessEntity>
  <name xml:lang="en">Company A</name>
  <contacts>…</contacts>
  <businessServices>
    <businessService>
      <name xml:lang="en">Company A's CPP</name>
      <description xml:lang="en">Company A's CPP</description>
      <bindingTemplates>
        <bindingTemplate>
          <accessPoint URLType="http">
            http://a.com/a.cpp
          </accessPoint>
          <description xml:lang="en">A's CPP is here!</description>
          <tModelInstanceDetails>
            <tModelInstanceInfo
              tModelKey="uuid:e3f3df4f-b221-33b4-a3ff-17b21410c565">
```

[2] In the examples in this technical note, v2 format keys are used. Note that entity key compatibility with v3 and earlier versions of UDDI should be considered unless the publisher intends to use this proprietary tModel in a closed environment only (e.g. a private registry). See Section 10 "Multi-Version Support" in **[UDDIV3]**.

[3] In this scenario, we assume that Company A has just one CPP. Though it is possible for a business entity to have multiple CPPs, the modeling of such a case is more complex.

```
            <description>
              ebxml-org:CollaborationProtocolProfile:v1_0
            </description>
          </tModelInstanceInfo>
        </tModelInstanceDetails>
      </bindingTemplate>
    </bindingTemplates>
  </businessService>
 </businessServices>
</businessEntity>
```

The business service above can be regarded as a service which delivers A's CPP. Company A issues `save_business` API call with above contents within `save_business` element to register its CPP. If Company A is already registered in a UDDI registry, its CPP can be registered by issuing a save_service API call using businessService defined above.

Company A registers an ebXML-based service

In this section, the step below is shown:

3. Company A registers a service S_a.

Company A already has a businessEntity which contains one businessService that delivers CPP. Now it can add a service S_a which uses ebXML Message Services protocol, and is conformant to the Standard Process #1 defined by ABC Consortium. Assuming that S_a is an online wine shop, and is provided at "http://a.com/wine/acceptPurchaseOrder", the new businessService is as follows:

```
<businessService businessKey="...Company A's businessKey...">
  <name xml:lang="en">A's online wine shop</name>
  <description xml:lang="en">an ebXML-based service</description>
  <bindingTemplates>
    <bindingTemplate>
      <accessPoint URLType="http">
        http://a.com/wine/acceptPurchaseOrder
      </accessPoint>
      <description xml:lang="en">A's ebXML-based service</description>
      <tModelInstanceDetails>
        <tModelInstanceInfo
          tModelKey="uuid:c8692873-1842-3b32-b980-8fa6d16676d2">
          <description>ebxml-org:MessageService:v1_0</description>
        </tModelInstanceInfo>
        <tModelInstanceInfo
          tModelKey="uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxbp1">
          <description>abc-org:StandardProcess:1</description>
        </tModelInstanceInfo>
      </tModelInstanceDetails>
    </bindingTemplate>
  </bindingTemplates>
  <categoryBag>
    <keyedReference
      tModelKey="uuid:CD153257-086A-4237-B336-6BDCBDCC6634"
      keyValue="50.20.22.05"
      keyName="UNSPSC:Sparkling_Wine" /> <!—- Service category info -->
  </categoryBag>
</businessService>
```

Company A issues `save_service` API with above contents to register the service.

2.2.6 Searching ebXML Services and Components

In this section, we show how Company B (buyer) finds a seller and retrieves a seller's CPP. Since the examples below are for explanation, they may sometimes be a little redundant.

Company B finds a service

In this section, the step below is shown:

4. Company B finds A's service S_a using a UDDI registry.

UDDI provides some powerful search capabilities. It is possible to find by business name, service category, service type, and so on. What follows are examples that illustrate uses of some of the UDDI APIs and the return structures of services and tModels registered in accordance with this TN. There are more (efficient) ways to obtain and query UDDI if richer use of categorization is employed for example.

In our scenario, it is assumed that Company B (buyer) is capable of doing business using ebXML framework, adopts the ABC Consortium's Standard Business Process #1, and wants to consume a service which is provided by an ebXML-ready seller.

The sample XML below finds a service provider who has an ebXML CPP (which means that it is ebXML-ready):

```
<find_business>
  <tModelBag>
    <!-- ebxml-org:CollaborationProtocolProfile:v1_0 -->
    <tModelKey>uuid:e3f3df4f-b221-33b4-a3ff-17b21410c565</tModelKey>
  </tModelBag>
</find_business>
```

Company B issues the above inquiry and gets a businessList like below as a result:

```
<businessList>
  <businessInfos>
    <businessInfo businessKey="...Company A's businessKey...">
      <name>Company A</name>
      <serviceInfos>
        <serviceInfo serviceKey="...serviceKey of CPP...">
          <name>Company A's CPP</name>    <!-- Company A has a CPP! -->
        </serviceInfo>
      </serviceInfos>
    </businessInfo>
  </businessInfos>
</businessList>
```

Thus Company B finds that Company A is a seller who provides an ebXML-based online service. Next, Company B checks if Company A provides a service that is conformant to the ABC Consortium's Standard Business Process #1 adopted by Company B.

The example below finds Company A's service which uses ebXML Message Services as communication protocol and which is conformant to the Standard Process #1 defined by the ABC Consortium.

```
<find_service businessKey="...Company A's businessKey...">
  <tModelBag>
    <!-- ebxml-org:MessageService:v1_0 -->
    <tModelKey>uuid:c8692873-1842-3b32-b980-8fa6d16676d2</tModelKey>
    <!-- abc-org:StandardProcess:1 -->
    <tModelKey>uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxbp1</tModelKey>
  </tModelBag>
</find_service>
```

Thus Company B has found Company A's ebXML service which meets the requirements. The next step is to obtain Company A's CPP to negotiate communication parameters.

Company B obtains A's CPP

In this section, the step below is shown:

5. Company B locates A's CPP using the UDDI registry.

The sample `find_service` invocation below allows to locate Company A's CPP.

```
<find_service businessKey="...Company A's businessKey...">
  <tModelBag>
    <!-- ebxml-org:CollaborationProtocolProfile:v1_0 -->
    <tModelKey>uuid:e3f3df4f-b221-33b4-a3ff-17b21410c565</tModelKey>
  </tModelBag>
</find_service>
```

Company B issues the above inquiry and gets a `serviceList` like below as a result:

```
<serviceList>
  <serviceInfos>
    <serviceInfo businessKey="...Company A's businessKey..."
       serviceKey="...serviceKey of CPP...">
      <name>Company A's CPP</name>
    </serviceInfo>
  </serviceInfos>
</serviceList>
```

Once Company B has the serviceKey of the business service which contains the information about A's CPP, it can issue a `get_serviceDetail` to retrieve the access point for the CPP.

```
<get_serviceDetail>
  <serviceKey>"...serviceKey of CPP..."</serviceKey>
</get_serviceDetail>
```

Company B gets full information of the businessService and knows that Company A's CPP is located at "http://a.com/a.cpp". Now Company B can fetch that CPP and then initiate negotiations with Company A. This calls for Company B providing its own CPP, negotiating the definition of a CPA[4], and then consuming S_a.

In a real system, the inquiries shown in the examples above would not be used since they are redundant and inefficient. This section is provided to give an understanding of the concepts involved.

2.2.7 Scenario Variation: Using a CPA Template

While **[ebCPPA2]** suggests that a CPA is formed from two CPPs, it also mentions an alternative approach which uses a CPA template to create a CPA. Since a CPA template represents one company's proposed configuration of service interface to perform business collaboration, the other trading partner can simply replace the placeholder values to form an agreed CPA. It is an easier way of negotiating a CPA.

In our scenario described in section 2.2.1, a CPA template provided by Company A (seller) can be used as an alternative to Company A's CPP. In this case, Company A can publish a business service shown below at the Step 2:

```
<businessService>
  <name xml:lang="en">Company A's CPA Template</name>
  <description xml:lang="en">
     CPA template provided by Company A
  </description>
  <bindingTemplates>
    <bindingTemplate>
```

[4] The ways to exchange CPP and to negotiate CPA are not defined here.

```
      <accessPoint URLType="http">
        http://a.com/a.cpa-template
      </accessPoint>
      <description xml:lang="en">CPA template is here!</description>
      <tModelInstanceDetails>
        <tModelInstanceInfo
           tModelKey="uuid:5ab4e3af-2e67-3a4f-b9b7-92a436be8f43">
          <description>
             ebxml-org:CollaborationProtocolAgreement:v1_0:Template
          </description>
        </tModelInstanceInfo>
      </tModelInstanceDetails>
    </bindingTemplate>
  </bindingTemplates>
</businessService>
```

The rest of the steps are similar to the original scenario except that this variation uses a CPA template instead of a CPP.

2.2.8 Scenario Variation: Using Role Information

In the reference scenario, there appears to be an unspoken agreement with regard to "role" (i.e. buyer vs. seller) between a publisher and an inquirer that the former provides a service and the latter consumes the service.

Generally a BPSS instance describes all the roles involved in a business process, such as 'buyer' and 'seller'. When there are multiple parties providing a service within the same BPSS instance, we cannot suppose such implied agreement. Therefore it is impossible to determine which role of the ones described in a BPSS instance the service published to a UDDI registry belongs to.

Role information is one of the characteristics of the service provider, which varies with the service. Modeling service roles in UDDI data structures in the way that aids discovery would be useful for an inquirer to filter out irrelevant service registrations.

Company A registers role with service information

In Step 3 of the reference scenario, Company A could provide its role with a `keyedReference` in the `categoryBag` of the `businessService` like below:

```
  <categoryBag>
    <keyedReference
       tModelKey="uuid:CD153257-086A-4237-B336-6BDCBDCC6634"
       keyValue="50.20.22.05"
       keyName="UNSPSC:Sparkling_Wine" /> <!—- Service category info -->
    <keyedReference tModelKey="UUID:A035A07C-F362-44dd-8F95-
E2B134BF43B4"
      keyName="role"
      keyValue="seller" />
  </categoryBag>
```

In this example, uddi-org:general_keywords tModel is used to describe that the service acts as a "seller". Generally a business process (i.e., a BPSS instance) has its own vocabulary to specify a role, such as "seller", "shipper", "retailer", "service provider", and so on. Therefore, it is impossible to define a fixed word set to create a universal role taxonomy in advance. By using general_keywords taxonomy, a publisher can identify his role within the context of a specific business process.

A publisher's role is described in its CPP, using the role vocabulary that is defined by a BPSS instance. A publisher should pick the name of the role from its CPP, and use it as the keyedReference's keyValue so that a knowledgeable inquirer can use it for a query.

Company B finds a service using role

At the Step 4 in the reference scenario, Company B could find a service using role information. In the case that an inquirer seeks a service provider whose service is conformant to a certain BPSS instance, it is quite natural to suppose that the inquirer is very familiar with the business process, otherwise they would fail to collaborate even if they came to know each other.

Therefore, we can assume that Company B knows the business process well, which means that the role name of the counter party Company B looks for is also known at the time of inquiry. Company B could filter out undesirable results by using adding the below categoryBag to a `find_service` inquiry:

```
  <categoryBag>
    <keyedReference tModelKey="UUID:A035A07C-F362-44dd-8F95-
E2B134BF43B4" keyName="role" keyValue="seller" />
  </categoryBag>
```

The rest of the steps are similar to the original scenario.

3 References

3.1 Normative

[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.

[UDDIV3] L. Clément et al, *UDDI V3.0 Published Specification*, http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf, OASIS UDDI Spec TC Committee Specification, July 2002.

[ebBPSS] Business Process Project Team, *ebXML Business Process Specification Schema Version 1.01*, http://www.ebxml.org/specs/ebBPSS.pdf, May 2001.

[ebCPPA1] ebXML Trading-Partners Team, *Collaboration-Protocol Profile and Agreement Specification Version 1.0*, http://www.ebxml.org/specs/ebCPP.pdf, May 2001.

[ebCPPA2] OASIS ebXML Collaboration Protocol Profile and Agreement Technical Committee, *Collaboration-Protocol Profile and Agreement Specification Version 2.0*, http://www.ebxml.org/specs/ebcpp-2.0.pdf, September 2002.

[ebMS1] ebXML Transport, Routing & Packaging Team, *Message Service Specification Version 1.0*, http://www.ebxml.org/specs/ebMS.pdf, May 2001.

[ebMS2] OASIS ebXML Messaging Services Technical Committee, *Message Service Specification Version 2.0*, http://www.ebxml.org/specs/ebMS2.pdf, April 2002.

[ebRR1] OASIS ebXML Registry Technical Committee, *Registry Information Model v2.0*, http://www.ebxml.org/specs/ebrim2.pdf, December 2001.

[ebRR2] OASIS ebXML Registry Technical Committee, *Registry Services Specification*, http://www.ebxml.org/specs/ebrs2.pdf, December 2001.

Appendix A. Acknowledgments

The author(s) would like to thank Joel Munter, Sean MacRoibeaird, and Ed Mooney for their contributions to this work.

Appendix B. Revision History

Rev	Date	By Whom	What
1	January 8, 2003	Keisuke Kibakura	Initial version.
2	January 31, 2003	Keisuke Kibakura	Clarification statement added: - Support for UDDI v2 and v3. - No intent to substitute ebXML RR. Improved modeling: - Restructured spec taxonomy - Addition of common tModels Brief outline of modeling Terminology clarification - CPPA, CPP, CPA, CPA template Added scenario variation.
3	February 27, 2003	Keisuke Kibakura	Updates based on the F2F discussion: - Added CPA positioning. - Added modeling rationale. - tModel keys changed. - Added footnote on multiple CPPs. - Removed errors in the example.
4	March 19, 2003	Luc Clément Daniel Feygin Tony Rogers Keisuke Kibakura	Reviewed by editors.
5	April 16, 2003	Keisuke Kibakura	Refinement of tModel categorization. Added new section on role information.
6	May 8, 2003	Keisuke Kibakura Daniel Feygin Tony Rogers Luc Clément	Final Draft
7	September 20-22, 2003	Daniel Feygin	Incorporated ebXML JC's input.

Appendix C. Notices

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

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