UDDI Spec TC

Versioning Value Sets in a UDDI Registry, Version 1.12

Technical Note

Document identifier:

uddi-spec-tc-tn-versioning-value-sets-v112-20030829

Current version:

http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-versioning-value-sets-v112-20030829.htm

Latest version:

http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-versioning-value-sets.htm

Editor:

Claus von Riegen, SAP

Contributors:

Tom Bellwood, IBM

Abstract:

Through the use of value sets in UDDI registries, businesses are able to find each other and the services that meet their needs. However, value set publishers often change their value sets by adding or deleting values and/or changing their meaning in order to meet the needs of a certain domain. This document provides guidelines for providers of value sets on how to register different versions of value sets for use in UDDI versions 2 and 3.

Status:

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

Committee members should send comments on this technical note to the uddi-spec@lists.oasis-open.org list. Others should subscribe to and send comments to the uddi-spec-comment@lists.oasis-open.org list. To subscribe, send an email message to uddi-spec-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

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

Table of Contents

Introduction 3
1.1 Problem statement 3
1.2 Terminology 3
1.3 Background 3
2 Technical note Solution 4
2.1 Definitions 4
2.1.1 Normative per the UDDI Specification 4
2.1.2 Defined in this Technical Note 5
2.2 Technical note behavior 6
2.3 Discussion 6
2.4 update_entities Web service type 6
2.4.1 update_entities for UDDI Version 2 entities 7
2.4.2 update_entities for UDDI Version 3 entities 9
2.5 Examples 10
2.5.1 Example of uddi-org:isReplacedBy usage 10
2.5.2 Example of update_entities Web service 11
2.5.3 Example of update_entities API call 12
3 References 14
3.1 Normative 14
Appendix A. Acknowledgments 15
Appendix B. Revision History 16
Appendix C. Notices 17

Introduction

Data has little value if it is lost within a mass of other data and cannot be distinguished or discovered. This applies to UDDI data as well. If a client of UDDI cannot effectively find information within a registry, the purpose of UDDI is considerably compromised. In UDDI there are two mechanisms for providing organization to data: category systems and identifier systems. Generically, these are called “value sets”. The UDDI specification **[UDDIV3]** provides several value sets (for example, the UDDI Types category system uddi-org:types), although many others exist and are continually produced (for example, the UDDI Business Registry provides a geographic category system that is based on ISO 3166). Often industry verticals define their own value sets, which are specific to the business they conduct. Many more have yet to be produced. UDDI allows publishers to register these value sets, and associate them with UDDI entities, thus allowing publishers and searchers to distinguish and discover business relevant information with ever greater precision. But these value sets change over time, and if they do so in an uncontrolled way searchers won’t know what to ask for and publishers won’t know how to categorize and identify their UDDI entities.

1.1 Problem statement

This UDDI Technical Note describes the preferred way to deal with changes in value sets for UDDI version 2 and 3. It outlines a set of practices that allow category systems and identifier systems to change in useful ways without invalidating UDDI entities that use them. The practices are almost identical between UDDI versions 2 and 3.

1.2 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]**.

1.3 Background

This document assumes a working knowledge of the UDDI Specification. In particular, it assumes familiarity with how category systems and identifier systems are represented in UDDI, how they are used to categorize and identify entities in UDDI registries, and how they are used in UDDI find messages.

For more information see sections 1.6.5, 3.3.2.10, 3.3.2.11, 3.3.2.12, 3.3.2.13, 3.3.2.14, and 11.16 as well as Appendix E and F of **[UDDIV3]**.

This technical note takes “Versioning Taxonomy and Identifier Systems”, written by David Ehnebuske and Barbara McKee, http://uddi.org/pubs/tn-taxonomy-versioning-v1.05-Draft-20010906.pdf and modifies it to fit in **[UDDIV3]** and follows the OASIS UDDI Specification TC Technical Note format.

2 Technical note Solution

2.1 Definitions

2.1.1 Normative per the UDDI Specification

value set

A value set is a category system or an identifier system that is either checked or unchecked.

checked value set

A value set whose use is inspected for conformance to the requirements of the value set.

unchecked value set

A value set whose use is not inspected for conformance to the requirements of the value set.

category system

A category system is a mechanism for categorizing UDDI entities so that they can be described and discovered based on certain categories. UDDI does not prescribe any specific category system.

identifier system

An identifier system is a mechanism for Identifying UDDI entities so that they can be described and discovered based on certain identifiers. UDDI does not prescribe any specific identifier system.

tModel

A tModel is a keyed entity that provides a reference system based on abstraction. Two primary uses are as sources for determining compatibility of Web services and as a keyed namespace reference.

identifierBag

This is an optional container element of a UDDI entity that identifies it according to one or more published identifier systems, for example, Dun & Bradstreet D-U-N-S$^{®}$ numbers or tax identifiers. An identifierBag is a list of one or more keyedReference structures, each representing a single identification.

categoryBag

This is an optional container element of a UDDI entity that identifies it according to one or more published categorization systems. For example, UNSPSC product and service categorizations codes can be used to describe a product or service offering. A categoryBag is a list of one or more keyedReference structures, each representing a singe category. A categoryBag may also contain keyedReferenceGroups

keyedReference (identifier and category)

A keyedReference is a mechanism for representing a categorization or an identifier name/value pair from a specific category or identifier system. It consists of three values: tModelKey, KeyName, and KeyValue.

keyedReferenceGroup

A keyedReferenceGroup is a container for a list of keyedReference structures that logically belong together.

uddi-org:isReplacedBy

It is often desirable to gracefully retire a tModel in favor of a replacement. For example, when a Web service definition or a value set is replaced by a new version, the publisher of the specification may wish to leave the tModel for the existing definition in place so that existing uses will not be disturbed, while at the same time making it clear that there is a replacement available. The UDDI isReplacedBy identifier system, coupled with the behavior of UDDI with respect to deleting tModels, fill this need by allowing the obsolete tModel to point to its replacement. (see section 11.1.6 of **[UDDIV3]**)

2.1.2 Defined in this Technical Note

valid code

Any value that, per the definition of the given category or identifier system may appear as the value of the keyValue attribute of a keyedReference that refers to the given category or identifier system[1]. Such a code is said to be “in” the category or identifier system; codes that are not valid are not “in” the category or identifier system. For example, 15111505 is a valid code and is therefore in the UNSPSC Version 3.1 taxonomy because ECCMA, the author of UNSPSC has so indicated. The code 15111521 is not valid in UNSPSC Version 3.1 because ECCMA has chosen not to include it.

code space

The code space is the set of all valid codes for a category or identifier system. For example, all of the codes in the UNSPSC Version 3.1 category system taken as a set constitute the code space of the UNSPSC Version 3.1 category system.

upward compatibility of code spaces

A code in category system T_2 is upward compatible with a code in category system T_1 if and only if the two codes are identical and if the code’s meaning in T_2 is a superset of its meaning in T_1. A code in an identifier system I_2 is upward compatible with a code in identifier system I_1 if and only of the two codes are the same and the code’s meaning is the same. For example, UNSPSC Version 6.01 code 55121701 means “Vinyl letters”. This code is upward compatible with UNSPSC Version 6.03 because in that version 55121701 means “Vinyl letters or numbers” which is a superset of its meaning in Version 6.01.

Value set V_2 is *upward compatible* with value set V_1 if and only if every code in the code space of V_1 has an upward compatible counterpart in the code space of V_2. For example, the UNSPSC Version 7 category system is not upward compatible with the UNSPSC

[1] Note that whether or not a given value is a valid code is independent of the value set’s characteristics in terms of checking.

Version 3.1 category system because the format of all of the codes changed in version 7. In version 3.1 the form of the codes is “########” (eight decimal digits) in version 7, the format is “##.##.##.##.##” (five decimal digit pairs separated by “.”).

successor version

The successor version is a version of a value set whose tModel is linked to via a uddi-org:isReplacedBy keyedReference from that of a predecessor version.

predecessor version

The predecessor version is a version of a value set whose tModel contains an uddi-org:isReplacedBy keyedReference that links to a successor version.

2.2 Technical note behavior

If a new version of a value set is upward compatible with the previous version, the existing tModel should be modified to indicate that it represents the new version, and then be republished using the same tModelKey.

If a new version of a value set is not upward compatible with the previous version, a new tModel should be published to represent the new version of the value set. The existing tModel should then be made the predecessor version by adding a keyedReference to its identifierBag that refers, using the uddi-org:isReplacedBy identifier system described in section 11.1.6 in **[UDDIV3]**, to the tModel of the new version of the value set.

2.3 Discussion

Changes to value sets that are upward compatible are preferable to those that are not because registry entries containing codes from the old version are still correct under the new version without change.

When codes in an existing version become obsolete in a new version, it is helpful for them to remain valid in the new version because it lets the new version be treated as upwardly compatible. New uses of obsolete codes may be discouraged through other means, for example by marking them obsolete in value set browsers.

When new, upwardly compatible, versions of value sets are published that contain valid but obsolete codes, value set providers should consider offering a Web service that implements the uddi-org:updateEntities Web service type (see section 2.4). This can help users of the old version eliminate the valid but obsolete codes from their UDDI entities.

When it is necessary to make a successor version – i.e., one that is not upwardly compatible – publishers should:

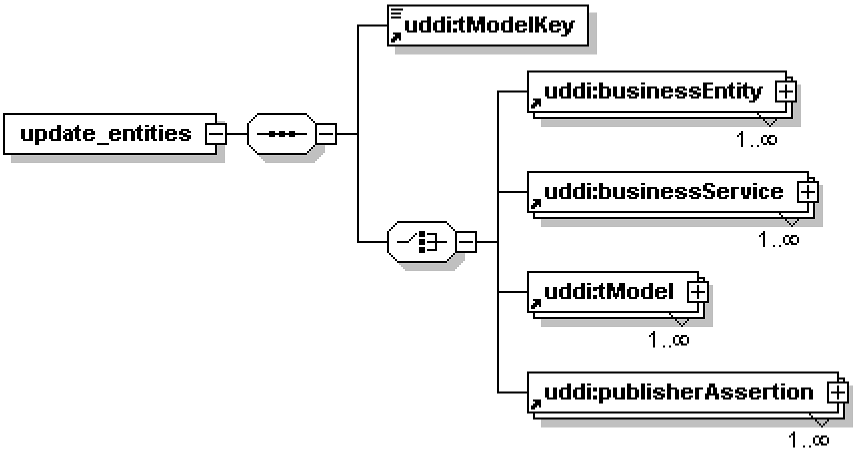
- Continue to support the predecessor, for example with value set browsers, at least until usage drops substantially.
- Chain the tModel for the predecessor version to the tModel for the successor (using uddi-org:isReplacedBy, as defined in section 11.1.6 in **[UDDIV3]**) so tools can find the most recent version.
- Consider offering a Web service that implements the uddi-org:updateEntities Web service type (see section 2.4) Such a service accepts UDDI entities and returns a modified version which is compatible with the successor version of the value set.

2.4 update_entities Web service type

The update_entities Web service is used to bring a list of UDDI entities (businessEntity, businessService, bindingTemplate, tModel, or publisherAssertion) up to date with respect to the value set tModel whose tModelKey it is passed. There are two Web service type definitions for the different UDDI versions the passed entities are based upon.

2.4.1 update_entities for UDDI Version 2 entities

Syntax:



Arguments:

- **tModelKey:** This key identifies the value set that is to be updated. keyedReferences in the entities that are also passed referring to this value set or to its predecessor versions are the targets of action for the invocation.

The following parameters are mutually exclusive

- **businessEntity:** The businessEntity elements whose contained (either directly or indirectly) keyedReferences are to be operated upon.
- **businessService:** The businessService elements whose contained keyedReferences are to be operated upon.
- **tModel:** The tModel elements whose contained keyedReferences are to be operated upon.
- **publisherAssertion:** The publisherAssertion elements whose contained keyedReferences are to be operated upon.

Behavior:

Web services of this type aid in updating entities that are categorized or identified by changed versions of a value set. When value set providers need to create new versions, they may also choose to provide services of this type to help those who use the value sets convert UDDI entities that refer to older versions.

A Web service of this type does not change anything in a UDDI registry; it simply operates on the XML for the UDDI entity it is passed, and returns a result. What, if anything, is done with the result is up to the caller. (Typically, though, the caller will be the entity's publisher who will use the result to republish the entity in a subsequent step.)

The resulting entity is unchanged except for those keyedReferences that refer to the given value set.

- Changed keyedReferences (in terms of the document order) indicate that a conversion to the newest version of the value set, i.e. the one that has no successor, is possible by using the returned keyedReference instead.
- Unchanged keyedReferences (in terms of the document order) indicate that a conversion to the newest version of the value set is not possible without further investigation. This may be the case if, for example, a value in a predecessor version is replaced by two values in a successor version.

Because of this behavior and depending upon the trustworthiness of the service, users should inspect the result to ensure that it meets their needs.

Returns:

The updated entities are returned in a UDDI businessDetail, serviceDetail, tModelDetail, or publisherAssertions message, depending on the type of the entities passed.

Caveats:

If any error occurs in processing this message, a dispositionReport structure will be returned to the caller in a SOAP Fault. The following error number information will be relevant:

- **E_invalidKeyPassed**: signifies that the *tModelKey* value passed did not match the key for a value set tModel that this service is capable of processing.
- **E_invalidValue**: signifies that a value that was passed in a keyValue attribute does not belong to the value set's known code space. The error text should clearly indicate the value. This error does not necessarily indicate that the value set is checked and the value did not pass validation, it simply indicates that the update_entities Web service does not recognize the value to be in the value set.

Bindings:

Web services that implement update_entities_v2 should list as tModelInstanceInfo structures, the tModelKey for uddi-org:updateEntities_v2 and the tModelKey for each value set the Web service is capable of updating. This convention allows people to find implementations of update_entities_v2 capable of handling updates to value sets in which they are interested.

tModel definition:

Name:	`uddi-org:updateEntities_v2`
Description:	Service to update UDDI Version 2 entities that make use of value sets
UDDI Key (V3):	`uddi:uddi.org:update_entities_v2`
Derived V1/V2 format key:	`uuid:74e22d5a-91fb-3b93-8052-64fb4d281556`
Categorization:	`specification, xmlSpec, soapSpec`

XML schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:uddi-org:valueset_versioning_v2"
xmlns:uddi="urn:uddi-org:api_v2" xmlns:uddi_vs_versioning="urn:uddi-
org:valueset_versioning_v2" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified"
id="uddi_vs_versioning_v2">
  <!-- Copyright (c) OASIS Open 2003, All Rights Reserved.-->
  <xsd:import namespace="urn:uddi-org:api_v2"
schemaLocation="http://uddi.org/schema/uddi_v2.xsd"/>
  <xsd:element name="update_entities">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="uddi:tModelKey"/>
        <xsd:choice>
          <xsd:element ref="uddi:businessEntity" maxOccurs="unbounded"/>
          <xsd:element ref="uddi:businessService" maxOccurs="unbounded"/>
          <xsd:element ref="uddi:tModel" maxOccurs="unbounded"/>
          <xsd:element ref="uddi:publisherAssertion" maxOccurs="unbounded"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

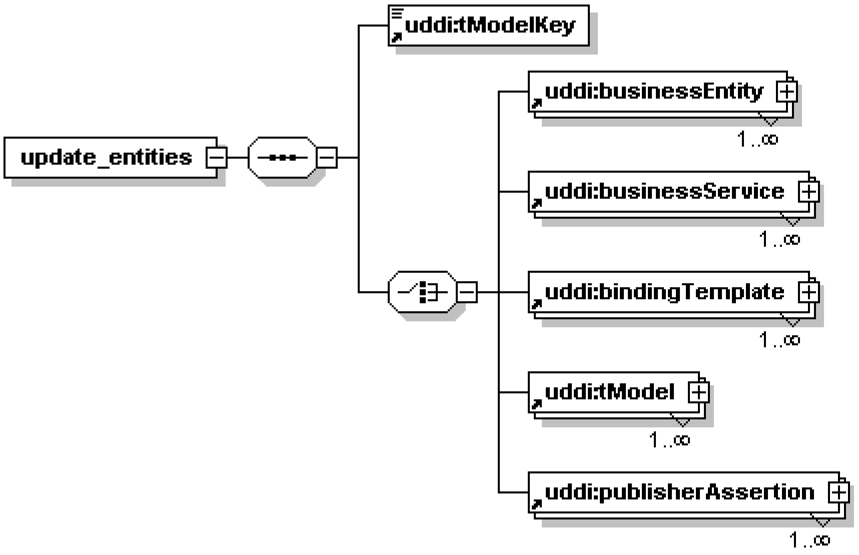
2.4.2 update_entities for UDDI Version 3 entities

The update_entities Web service type definition for UDDI Version 3 entities is a complete analogue of the one defined for UDDI Version 2 entities (see section above), except that

- It covers the update of UDDI Version 3 entities instead since the information model and semantics have changed between UDDI Version 2 and 3
- It covers the update of bindingTemplates since they are allowed to carry categoryBags in UDDI Version 3

Therefore, only the changed syntax and XML schema are listed here. The behavior is adopted from the V2 update_entities definition accordingly.

Syntax



tModel definition:

Name: `uddi-org:updateEntities_v3`

Description: Service to update UDDI Version 3 entities that make use of value sets

UDDI Key (V3): `uddi:uddi.org:update_entities_v3`

Derived V1/V2 format key: `uuid:06647e37-66ed-3973-9cce-857a0f43c12b`

Categorization: `specification, xmlSpec, soapSpec`

XML schema:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xsd:schema targetNamespace="urn:uddi-org:valueset_versioning_v3"
xmlns:uddi="urn:uddi-org:api_v3" xmlns:uddi_vs_versioning_v3="urn:uddi-
org:valueset_versioning_v3" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified"
id="uddi_vs_versioning_v3">
  <!-- Copyright (c) OASIS Open 2003, All Rights Reserved.-->
  <xsd:import namespace="urn:uddi-org:api_v3"
schemaLocation="http://uddi.org/schema/uddi_v3.xsd"/>
  <xsd:element name="update_entities">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="uddi:tModelKey"/>
        <xsd:choice>
          <xsd:element ref="uddi:businessEntity" maxOccurs="unbounded"/>
          <xsd:element ref="uddi:businessService" maxOccurs="unbounded"/>
          <xsd:element ref="uddi:bindingTemplate" maxOccurs="unbounded"/>
          <xsd:element ref="uddi:tModel" maxOccurs="unbounded"/>
          <xsd:element ref="uddi:publisherAssertion" maxOccurs="unbounded"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

2.5 Examples

2.5.1 Example of uddi-org:isReplacedBy usage

In the example below, the UNSPSC Version 3.1 category system has been replaced by the UNSPSC Version 7 category system. To indicate this, the uddi-org:isReplacedBy identifier system is used to point the unspsc-org:unspsc:3-1 tModel to the unspsc-org:unspsc tModel. To do this, the unspsc-org:unspsc:3-1 tModel has a keyedReference added to its identifierBag, as follows:

UDDI Version 2:

```
<tModel tModelKey="uuid:db77450d-9fa8-45d4-a7bc-04411d14e384">
  <name>unspsc-org:unspsc:3-1</name>
  ...
  <identifierBag>
      <!-- Use uddi-org:isReplacedBy to indicate that the UNSPSC 3.1
tModel is logically replaced by the UNSPSC 7 tModel. -->
    <keyedReference
      keyName="isReplacedBy:unspsc-org:unspsc"
      keyValue="uuid:cd153257-086a-4237-b336-6bdcbdcc6634"
      tModelKey="uuid:e59ae320-77a5-11d5-b898-0004ac49cc1e"/>
  </identifierBag>
  ...
</tModel>
```

UDDI Version 3:

```
<tModel tModelKey="uddi:db77450d-9fa8-45d4-a7bc-04411d14e384">
  <name>unspsc-org:unspsc:3-1</name>
  ...
  <identifierBag>
      <!-- Use uddi-org:isReplacedBy to indicate that the UNSPSC 3.1
tModel is logically replaced by the UNSPSC 7 tModel. -->
    <keyedReference
      keyName="isReplacedBy:unspsc-org:unspsc"
      keyValue="uddi:cd153257-086a-4237-b336-6bdcbdcc6634"
      tModelKey="uddi:uddi.org:identifier:isreplacedby"/>
  </identifierBag>
  ...
</tModel>
```

The keyName attribute is commentary serving to help readability. The keyValue specifies which tModel replaces this one – the unspsc-org:unspsc tModel in this case. The tModelKey specifies that the keyedReference is using the uddi-org:identifier:isReplacedBy identifier system.

2.5.2 Example of update_entities Web service

In the example below, the bindingTemplate represents a Web service that implements the update_entities Web service type, indicated by the first tModelInstanceInfo that references the uddi-org:update_entities_v2 and uddi-org:update_entities_v3 tModels, respectively. The second tModelInstanceInfo references the UNSPSC 3.1 category system and thus indicates that the Web service is capable in updating values from this version to its predecessor versions.

update_entities_v2 in UDDI Version 2:

```
<bindingTemplate>
  <accessPoint URLType="http">
    http://www.example.com/unspsc_conversion
  </accessPoint>
  <tModelInstanceDetails>
    <tModelInstanceInfo
      tModelKey="TBD" />
    <tModelInstanceInfo
      tModelKey="uuid:db77450d-9fa8-45d4-a7bc-04411d14e384" />
  </tModelInstanceDetails>
</bindingTemplate>
```

update_entities_v3 in UDDI Version 2:

```
<bindingTemplate>
  <accessPoint URLType="http">
    http://www.example.com/unspsc_conversion
  </accessPoint>
  <tModelInstanceDetails>
    <tModelInstanceInfo
      tModelKey="TBD" />
    <tModelInstanceInfo
      tModelKey="uuid:db77450d-9fa8-45d4-a7bc-04411d14e384" />
  </tModelInstanceDetails>
</bindingTemplate>
```

update_entities_v2 in UDDI Version 3:

```
<bindingTemplate>
  <accessPoint useType="endPoint">
    http://www.example.com/unspsc_conversion
  </accessPoint>
  <tModelInstanceDetails>
    <tModelInstanceInfo tModelKey="uddi:uddi.org:update_entities_v2" />
    <tModelInstanceInfo
      tModelKey="uddi:db77450d-9fa8-45d4-a7bc-04411d14e384" />
  </tModelInstanceDetails>
</bindingTemplate>
```

update_entities_v3 in UDDI Version 3:

```
<bindingTemplate>
  <accessPoint useType="endPoint">
    http://www.example.com/unspsc_conversion
  </accessPoint>
  <tModelInstanceDetails>
    <tModelInstanceInfo tModelKey="uddi:uddi.org:update_entities_v3" />
    <tModelInstanceInfo
      tModelKey="uddi:db77450d-9fa8-45d4-a7bc-04411d14e384" />
  </tModelInstanceDetails>
</bindingTemplate>
```

2.5.3 Example of update_entities API call

In the example below, an update_entities API call is used to update a businessEntity with respect to the UNSPSC 3.1 category system, indicated by the tModelKey uddi:db77450d-9fa8-45d4-a7bc-04411d14e384. The businessEntity contains a categoryBag that in turn contains two keyedReferences, both representing UNSPSC 3.1 categories.

UDDI Version 2:

```
<update_entities xmlns="urn:uddi-org:valueset_versioning_v2"
xmlns:uddi="urn:uddi-org:api_v2">
  <uddi:tModelKey>uuid:db77450d-9fa8-45d4-a7bc-04411d14e384</tModelKey>
  <uddi:businessEntity businessKey="...">
    ...
    <uddi:categoryBag>
      <uddi:keyedReference
        ...
        tModelkey="uuid:db77450d-9fa8-45d4-a7bc-04411d14e384" />
      <uddi:keyedReference
        ...
        tModelKey="uuid:db77450d-9fa8-45d4-a7bc-04411d14e384" />
    </uddi:categoryBag>
    ...
  </uddi:businessEntity>
</update_entities>
```

The businessDetail return structure shown below contains the businessEntity and two keyedReferences, each corresponding to the ones given in the update_entities API call. Since the first keyedReference still refers to the UNSPSC 3.1 category system, the update_entity Web service was obviously not able to find an appropriate category in the UNSPSC 7 successor version. The second keyedReference refers to the UNSPSC 7 category system and thus, was updated.

```
<businessDetail xmlns="urn:uddi-org:api_v2">
  <businessEntity businessKey="...">
    ...
    <categoryBag>
      <keyedReference
        ...
        tModelKey="uuid:db77450d-9fa8-45d4-a7bc-04411d14e384" />
      <keyedReference
        ...
        tModelKey="uuid:cd153257-086a-4237-b336-6bdcbdcc6634" />
    </categoryBag>
    ...
  </businessEntity>
</businessDetail>
```

UDDI Version 3:

```
<update_entities xmlns="urn:uddi-org:valueset_versioning_v3"
xmlns:uddi="urn:uddi-org:api_v3">
  <uddi:tModelKey>uddi:db77450d-9fa8-45d4-a7bc-04411d14e384</tModelKey>
  <uddi:businessEntity businessKey="...">
    ...
    <uddi:categoryBag>
      <uddi:keyedReference
        ...
        tModelkey="uddi:db77450d-9fa8-45d4-a7bc-04411d14e384" />
      <uddi:keyedReference
        ...
        tModelKey="uddi:db77450d-9fa8-45d4-a7bc-04411d14e384" />
    </uddi:categoryBag>
    ...
  </uddi:businessEntity>
</update_entities>
```

The businessDetail return structure shown below contains the businessEntity and two keyedReferences, each corresponding to the ones given in the update_entities API call. Since the first keyedReference still refers to the UNSPSC 3.1 category system, the update_entity Web service was obviously not able to find an appropriate category in the UNSPSC 7 successor version. The second keyedReference refers to the UNSPSC 7 category system and thus, was updated.

```
<businessDetail xmlns="urn:uddi-org:api_v3">
  <businessEntity businessKey="...">
    ...
    <categoryBag>
      <keyedReference
        ...
        tModelKey="uddi:db77450d-9fa8-45d4-a7bc-04411d14e384" />
      <keyedReference
        ...
        tModelKey="uddi:cd153257-086a-4237-b336-6bdcbdcc6634" />
    </categoryBag>
    ...
  </businessEntity>
</businessDetail>
```

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 Version 3.0 Specification*, http://uddi.org/pubs/uddi_v3.htm, OASIS UDDI Spec TC Committee Specification, July 2002.

Appendix A. Acknowledgments

The editor would like to thank David Ehnebuske and Barbara McKee (both IBM) for their initial contribution of this technical note.

Appendix B. Revision History

Rev	Date	By Whom	What
1.09	1/02/2003	Pat Felsted	Updated to UDDI Version 3 Product Specification and Oasis technical note format.
1.10	22/4/2003	Claus von Riegen	• Aligned terminology • Separated between normative UDDI terminology and TN definitions • Changed example to UNSPSC • Added two examples • Updated update_entities Web service behavior • Added XML schema for update_entities Web service
1.11	27/5/2003	Pat Felsted	Initial update for version 2 compatibility.
1.12	29/8/2003	Claus von Riegen	Final update with regard to UDDI Version 2

Appendix C. Notices