OASIS

UDDI Spec TC

Technical Note

Understanding Key Partitions

Document identifier:
uddi-spec-tc-tn-understandingkeypartitions-20061128

This Version:
http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-understandingkeypartitions-20061128.htm

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

Author:
Tony Rogers, Computer Associates

Editors:
Luc Clément, Systinet
Andrew Hately, IBM

Abstract:
The key partitions capability is an important UDDI v3.0 feature that provides a mechanism for publishers to assign meaningful and globally unique registry keys to entities being published. The purpose of this technical note is to explain in lay person's terms the use of key partitions for UDDI registry publishers.

Status:
This document is updated periodically on no particular schedule.

Committee members should send comments on this technical note to the uddi-spec@lists.oasis-open.org list. Others should submit comments via http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=uddi-spec.

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/ipr.php).

Table of Contents

1 Introduction

This document is designed to explain the key partitions framework and provide best practice guidance to any party that wishes to publish data to a UDDI registry using publisher assigned keys. In particular this technical note answers two questions:

- Why should I assign my own key value to entities that I want to publish?
- If I do assign the key, how do I determine what value to use?

Readers should have a high level understanding of UDDI in order to use this technical note but need not read any other detailed UDDI technical specifications.

1.1 Problem statement

All registry entities such as Business Entities, Business Services, Binding Templates, and tModels require a unique key that is used to identify the entity. In the UDDI version 2 specification, these keys were generated by the registry node at the time the entity was created . In the UDDI version 3 specification publishers can choose whether to let the registry node assign a key as it is the case in version 2, or whether to assign a key themselves. We will examine two business problems in this section that the use of *publisher assigned keys* addresses. First, a discussion of the portability issues and lack of readability associated with node assigned keys. Second the key management challenges associated with publisher assigned keys.

1.1.1 Node assigned keys

Imagine you are an enterprise architect who is deploying a service oriented architecture within your company, Example1 org. The centerpiece of your architecture is a UDDI registry that catalogs all your business organizational units and the business services they provide. You have defined standard classification schemes for your services and have defined associations to standard interface definitions for each service as shown on the left side of Figure 1.

In order to define the classifications, the Example1 architect published *classification* tModels to represent the classification schemes. The Example1 registry node automatically assigned keys to each tModel. For example, “DEF” (represented at a GUID in version 2) was the key assigned to the GS1 Global Location Number (GLN) scheme. When the GLN scheme is used to identify a business entity such as an Example1 plant location, the identifier bag (the means by which metadata is associated with an entity in UDDI) for the business entity will contain the GLN value (say “0012345000041”) and the tModel keyedReference “DEF”.

Problem: *Node assigned keys have no meaning and make registry data hard to read. The EAN GLN classification is just two numbers. It is only by looking up the tModel with key “DEF” that the user can discover that the classification scheme is EAN GLN.*

It is reasonable to assume that some of the services listed in the Example1 registry are for use by external parties (e.g. the RFQ service in our example). In order that the service is discoverable by the business community, Example1 will need to republish the registry entry to a public registry node such as the “National service registry node” in figure 1. The registry operator of the national service registry is likely to have created common classification schemes for the same reasons as our Example1 architect. If the registry server generated its own keys for classification schemes, the key for the same EAN GLN classification scheme is “BBB” – different from the key in the Example1 system. Therefore when the Example1 data is republished to the national registry, nearly all the classifications and associations that gave context and meaning to the Example1 data are lost unless there is administrative intervention or the end user does some manual remapping of the data

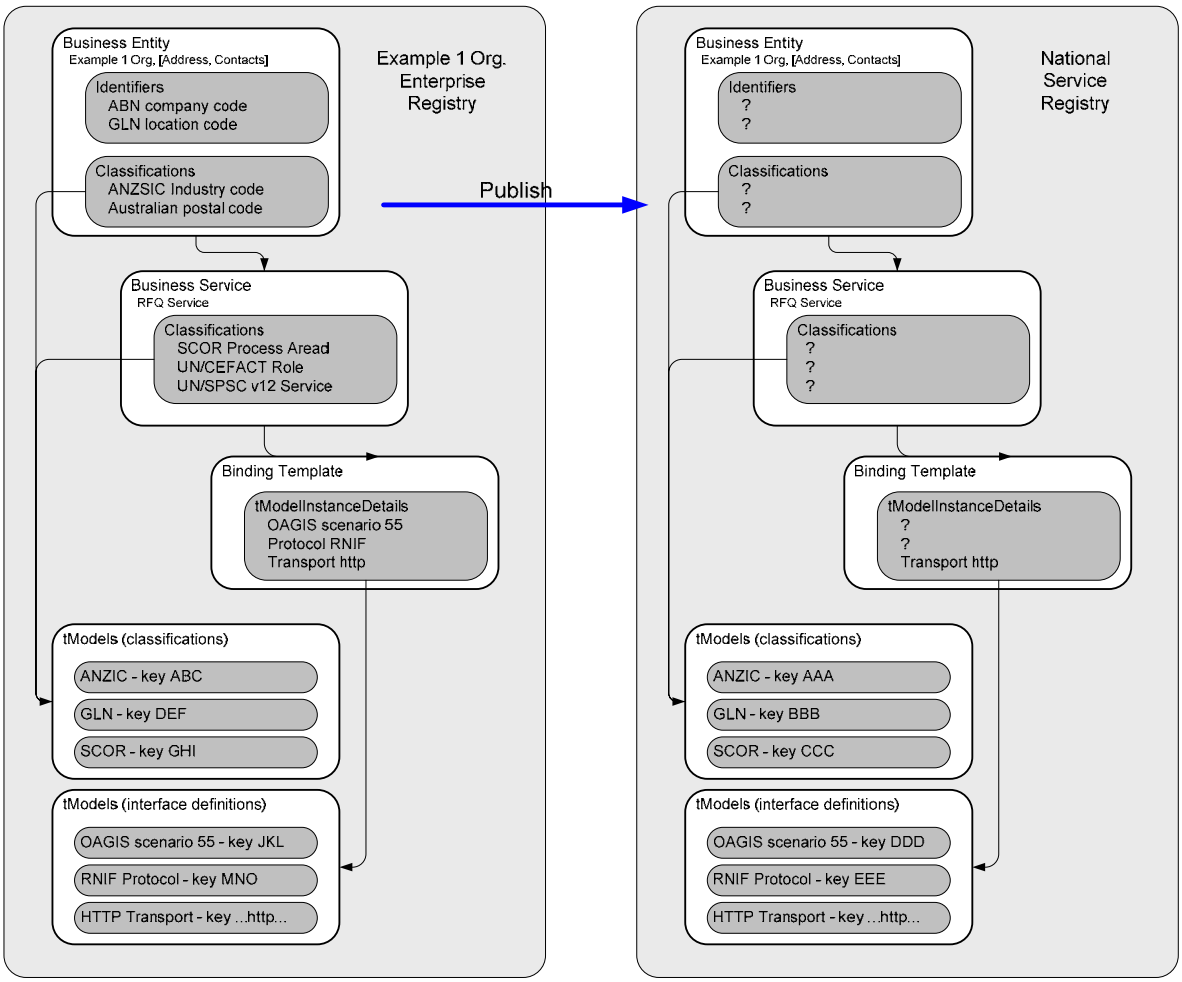

Figure 1 - republication scenario

Problem: *Node assigned keys are randomly generated by each UDDI registry so the same content (such as the GLN identifier scheme) published to two different servers will have different keys. When data is republished between registry nodes, all classifications and associations will be lost. In essence, what is lost is the means to semantically understand the information held by a node. The exception to this is subset of classifications that are defined in the UDDI specification itself (e.g. UDDI Types Category System) or published by an administrator using some product specific mechanism because they will have the same key in all compliant registry nodes. If a user has an entry they need to republish with node assigned keys, they could search the target registry for all classification systems to see if they could remap all of the keys to existing classification systems in the target registry. Another option is that the user could publish the classification systems and then update all node assigned keys used in references before adding the original data. In general, use of key partitions is designed to reduce the time required from both users and administrators when republishing entries.*

1.1.2 Publisher assigned keys

The Example1 enterprise architect will be relieved to hear that the version 3 specification provides a solution to the two problems described in the previous section. The solution lies in the basic principle that the owner of any specification should assign an identifier (in the form of a URI) to the specification and that the URI should always be used as the UDDI key irrespective of where the specification is published. In this case, a specification is the key of the taxonomy and its value set in the case of the examples described above. This is also applicable to interfaces as will be described in the next section. In our example, the key might

be “uddi:gs1.org:gln” and would be defined and owned by GS1. All registry operators that wish to provide an GS1 GLN identifier scheme would then publish the GS1 GLN identifier scheme as a tModel with a key value of “uddi:gs1.org:gln”.

This addresses both problems described in the previous section. The key value may be meaningful AND is consistent in all registry nodes. However this solution introduces a new challenge – ensuring uniqueness across several registries or even globally. Version 2 keys are globally unique because a mathematical algorithm generates keys sufficiently long that there is a vanishingly small chance of two nodes creating the same key. On the other hand, if publishers are free to assign keys in version 3 then there is no mechanism to ensure that two publishers don’t choose the same key for entities in different registries. For example, both Example1 Org. and Example2 Co. may create a sales order service and generate a WSDL document to describe their service. If both Example1 Org. and Example2 Co. assign the same “uddi:salesOrderServiceSpecification” key to the interface tModels they published respectively, then when each attempt to promote their respective interface tModel to the national registry by publishing them at this location, only the first publication will be permitted given that there can only be one use of the “uddi:salesOrderServiceSpecification” key in the national registry. Attempting to publish both services to the national registry would produce an error on the publication of the second tModel. If the second publisher persisted, then their bindingTemplate would be referencing the wrong interface tModel.

Problem: *Whilst publisher assigned keys can address the problem of the same specification having different keys in different registries, it introduces a new problem that two different specifications may be published with the same key – leading to conflicts and confusion.*

To address this problem, UDDI version 3 introduces the concept of key partitions.

1.1.3 Relationship to UDDI Taxonomy

It is important to note that taxonomies are represented by tModels in UDDI, and there are semantics beyond the key given to the tModel, and therefore it is important that administrators of cooperating UDDI nodes agree on the semantics and content of the value sets that these tModels represent.

While not within the scope of this Technical Note, administrator and value set provider should read the following technical notes:

- Providing a Taxonomy for Use in UDDI Version 2
- Providing a Value Set For Use in UDDI Version 3
- Versioning Value Sets in a UDDI Registry, Version 1.12

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]**.

2 Technical Note Solution

2.1 Definitions

UDDI Node – a physical instance of a registry service that complies with the UDDI specification. A node may be a standalone registry or it may be part of a federated set of registries such as the UBR (Universal Business Registry).

Publisher – a person or organization with the rights to publish UDDI entities in a UDDI node. Although a publisher will be referred to as singular throughout this document, a single publisher may be multiple people, all with the right to publish under the same account.

Business entity – the UDDI entity for describing a business. The UDDI element name for this entity is businessEntity.

Business service – the UDDI entity for describing a service provided by a business entity. One business entity may provide zero or more business services. The UDDI element name for this entity is businessService.

Binding template – the UDDI entity for describing the technical binding for a business service (ie how to use the business service). One business service may be described by one or more binding templates. A binding template provides the access point (eg URL) for invoking the service and lists a number of tModels that together provide a technical fingerprint for the service. The UDDI element name for this entity is bindingTemplate.

tModel – the UDDI entity that describes a technical interface, protocol, or classification system. A tModel is typically published by a standards organization or industry group that wishes to define a standard protocol, interface, or code list for use by the community. One tModel may be referenced by multiple business entities via their services and binding templates.

UDDI key – a globally unique (within the UDDI registry) identifier for a UDDI entity such as a business entity, a business service, a binding template, or a tModel. If assigned by a publisher, it must be a Uniform Resource Identifier conforming to RFC 2396. If the publisher does not assign a key then the UDDI node must create one.

Key partitions – a UDDI mechanism for managing keys in such a way as to minimize the likelihood of creating duplicate keys. This mechanism works by partitioning the space of possible keys and assigning different partitions to different publishers. In effect, this mechanism assigns one or more key prefixes to each publisher.

Key generator key – a key generator key is a UDDI key that ends with the string ":keygenerator". Ownership of the key generator key implies ownership of the key partition with which it is associated. Note that only key generator tModels are permitted to have key generator keys.

Key generator tModel – a tModel whose key is a key generator key. A publisher owns a key generator key (and its associated key partition) if the publisher owns the tModel with that key. Note that such a tModel must have a keyed reference in its category bag which categorizes it as a key generator.

2.2 Technical note behavior

2.2.1 Introduction

2.2.1.1 UDDI keys

UDDI keys come in two forms, which we can describe as “Version 2” (V2 for short), and “Version 3” (V3 for short).

Every business entity, business service, binding template, and tModel in a UDDI registry must have a unique key, because that key is how the entity is referenced. Managing this uniqueness is handled differently for V2 and V3 keys.

2.2.1.1.1 UDDI V2 keys

UDDI V2 keys are the only keys available in UDDI Version 2, and are assigned by the UDDI registry when a new UDDI entity is saved. They are formatted as UUIDs [RFC4122], which means they are 128bit numbers, usually written as a string of 32 hex digits (separated in groups by hyphens).

For example: “12345678-1234-1234-1234-1234567890ab” is a valid format UDDI V2 key.

This TN is not concerned with V2 keys. Note that the uniqueness of V2 keys is a problem for the UDDI registry to manage – the registry will ensure that it assigns a new key to each new entity that is saved.

2.2.1.1.2 UDDI V3 keys

UDDI V3 keys were introduced in UDDI Version 3. UDDI V3 keys are URIs (see RFC 2396) which begin with “uddi:”. Unlike UDDI V2 keys, these keys can be assigned by the publisher.

For example: “uddi:example.org:finance:app123” is a valid format UDDI V3 key.

This TN is concerned with the management of V3 keys. Because a publisher can allocate V3 keys to entities, there is a need for a system to control contention over keys. That system is called key partitions.

2.2.1.2 UDDI V3 key partitions and key generators

In the example above, the publisher of a new UDDI entity with the key “uddi:example.org:finance:app123” must control the key partition in which that key lies – i.e. where it exists. If the publisher does not control the key partition, the request will be rejected. The example key lies in the key partition with the prefix “uddi:example.org:finance”, so that is the key partition which the publisher must control. To control that partition, the publisher must own the key generator tModel for the partition. For this example, that is the tModel with the key “uddi:example.org:finance:keygenerator”.

2.2.1.3 Using a key partition for keys

The publisher who is the owner of a particular key partition is the only person permitted to publish new UDDI entities whose keys lie within that key partition. Any attempt by another publisher to publish a UDDI entity using a new key from within that key partition will be rejected by the UDDI registry.

Note that a publisher is permitted to modify an existing entity they control, even if they do not own the key partition in which its key lies. Such a situation may arise if they received control of the entity by way of custody transfer; or if they created the entity while they owned the key partition, and then transferred ownership of the key partition. As such, the enforcement of key partitions occurs when the entity is created.

A publisher creates a new key within a key partition by taking the prefix associated with that key partition and appending a colon, followed by a sequence of characters (the list of permitted characters will be listed later). Note that the publisher is responsible for ensuring

that this sequence forms a new key, rather than clashing with an existing key, otherwise the publisher will actually perform an update to an existing entry. It is beyond the scope of this TN to describe how the publisher may manage keys within the key partition. Suffice to say, that some planning should be done to ensure that the key identifier scheme will have some longevity and have significance to the community of users that will need to interpret them.

For example: the publisher who controls the key partition associated with the prefix “uddi:example.org:finance” can create the key “uddi:example.org:finance:dept3” and publish a business entity with that key.

2.2.1.4 Making a child key partition

To create a new key partition within an existing key partition the publisher who controls the parent key partition creates a new key within the partition, then appends the reserved string “:keygenerator” to that new key, and publishes a key generator tModel with that key. The act of publishing that tModel creates the new key partition.

For example, the publisher who controls “uddi:example.org:finance” can publish a tModel with the key “uddi:example.org:finance:services:keygenerator”, and that establishes a new key partition associated with “uddi:example.org:finance:services”. The publisher can then publish a business service with the key “uddi:example.org:finance:services:inquiry”. Note that controlling the key partition “uddi:example.org:finance” is not sufficient to publish the key “uddi:example.org:finance:services:inquiry” – the intermediate step of creating the new partition is required.

2.2.1.5 Top level key partitions

There are key partitions which have no apparent parent – these are called top-level key partitions. Following the examples above, “uddi:example.org” is the prefix associated with such a top-level key partition.

The UDDI node administrator controls the allocation of top level key partitions with the exception of domain keys such as uddi:example1.org. The exact mechanism where a registry will allow publishers to establish any top level partition or domain based key partition may vary from one implementation to another, and is beyond the scope of this document.

2.2.2 Creating & assigning key partitions

The UDDI key partition behavior may be expressed as a set of rules:

1. A UDDI node administrator is the owner of the root partition – “uddi:”
2. Any sequence of letters, numbers, escaped characters (which are represented by % hexdigit hexdigit) or characters from the set ; / ? @ & = + $, - _ . ! ~ * ' ( ) may be appended to a key partition (separated by a colon) to create a UDDI key. These appended characters are called the key specific string (kss). The string “keygenerator” is not an allowed kss.
3. A key partition owner may create a child key partition by appending the reserved string “:keygenerator” to a key within the key partition and publishing a key generator tModel with that key. For example, a registry node administrator may create a new partition for Example1 by publishing a tModel with the key “uddi:example1.org:keygenerator”.
4. Top level key partitions SHOULD be based on the DNS domain of the owning organization. In general, the name of a partition should be selected in such a way that it is meaningful to the organization and would be acceptable for a long period of time in order to avoid republishing entries with new keys should the naming become unacceptable.
5. The owner of a key generator key can transfer ownership to another registry user. In this case the new owner of the key generator key becomes the owner of the key partition it represents. For example, if the registry administrator creates “uddi:example1.org:keygenerator”, then transfers ownership of it to the Example1 administrator then Example1 is the new owner of the “uddi:example1.org” key

partition and can create new keys in the partition as defined in rule 2. The transfer of ownership is executed by transferring the tModel whose key is the key generator key.

6. Once a key generator ownership is transferred the original owner may no longer create keys in the partition represented by the key generator. This does not, however, invalidate previously generated keys.

The examples in the diagram below help to clarify these rules. Each box represents a key partition with the name of the partition in the top left corner of each box. Sample key values are provided in the partition to which they belong.

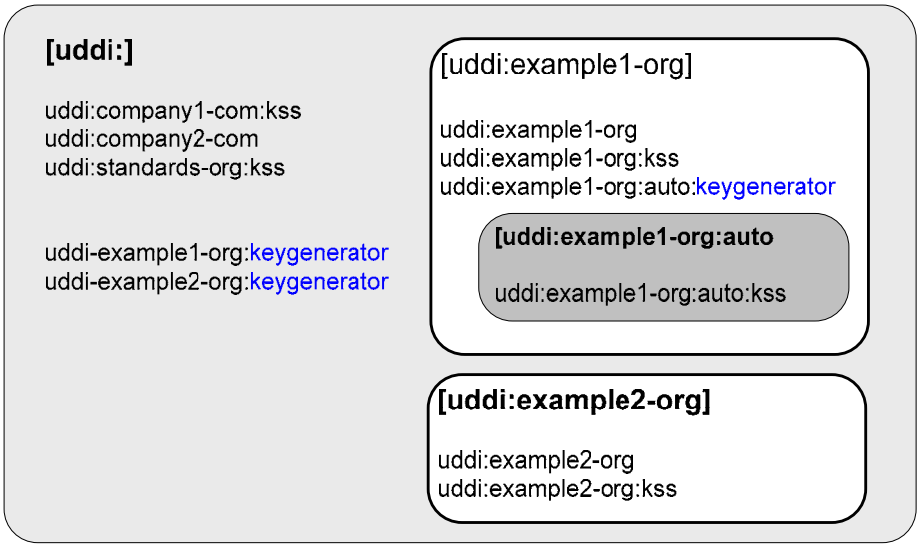

Figure 2 - key partitions example

The table below provides examples of keys that are and are NOT in key partition uddi:example1.org

key	Comments
In partition uddi:example1.org	
uddi:example1.org:invoicing	A key based on the same key as the key generator key belongs in the partition
uddi:example1.org:auto:keygenerator	A key generator key based on a derived key does belong in the partition.
uddi:example1.org	In the case of the domain being top level, the root key on which the partition is based does belong in the partition
Not in partition uddi:example1.org	
uddi:example1.org:auto:stockquery	Belongs to the partition defined by the key uddi-example1.org:auto:keygenerator
uddi:example1.org:keygenerator	The key generator key does not belong in the partition it designates
uddi:example1.org:keygenerator:hr	Does not belong in any partition – an invalid key.

Table 1 - key partition rules example

2.3 Common Questions

2.3.1 What partition is this key in?

If we are looking at a particular UDDI key, such as *uddi:Example1.org:finance:expenses*, how do we determine which partition this key is in? The answer is simple enough. The string to the right of the rightmost colon is the key specific string – in this case *expenses*. The remainder (to the left of the colon) is the name of the key partition – in this case *uddi:Example1.org:finance*. The pieces of the key are delimited by colons.

2.3.2 What is the key generator for this partition?

Continuing from the example above, if we have a key partition called *uddi:Example1.com:finance*, what is its key generator? It is the name of the partition, followed by ":keygenerator" – in this case: *uddi:Example1.com:finance:keygenerator*.

So we can see that the key generator for any key can be determined by replacing the rightmost string (the key specific string, delimited by the colon) with the special word "keygenerator".

2.3.3 Who is allowed to publish this key?

Suppose we want to determine who is allowed to publish a UDDI entity (say, a service) with the key *uddi:Example1.com:finance:expenses*. The only person allowed to publish that key (for the first time – the rules are different once the entity exists) is the owner of the key partition. And who owns the key partition? The owner of the key generator tModel is the owner of the partition. Summing it all up, the only person allowed to publish a service with the key *uddi:Example1.com:finance:expenses* is the owner of the tModel with the key *uddi:Example1.com:finance:keygenerator*.

2.3.4 How do we get ownership of the key partition?

Consider two cases, the first case where a user owns the keyGenerator uddi:example1.org and we want to create a sub partition. Publishing an entry uddi:example1.org:hr:keygenerator. The user now owns a second key partition uddi:example1.org:hr.

Lets assume another user wants to create an entry with the key *uddi:Example1.org:finance:expenses*. Further, in this example, there is no tModel with the key *uddi:Example1.com:finance:keygenerator*.

As in the previous case, the user must acquire ownership of the key generator tModel for the key partition in which this key lies.

Let us assume that the user can approach the owner of the key generator tModel for key partition uddi:Example1-org – the person who owns the key generator tModel with the key *uddi:Example1.com:keygenerator*. The user asks this person to issue us with the key generator for the key partition *uddi:Example1.com:finance*. This is done by publishing the tModel with the key *uddi:Example1.com:finance:keygenerator*. They then transfer custody of this tModel to the user who requested it. Now the user can publish keys in the *uddi:Example1.com:finance* key partition.

Note that once the owner of the *uddi:Example1.com* key partition transfers custody of the *uddi:Example1.com:finance:keygenerator* tModel , they no longer own it, and therefore they cannot publish any more keys in the associated partition.

2.3.5 What about existing keys?

All the rules about ownership of key generator tModels only apply to the publication of new keys, not to existing keys. For keys that already exist, the rules are even simpler: the owner of an existing key (technically, the owner of the UDDI entity with an existing key) is the only person who can modify it.

Suppose, for example, that the owner of the *uddi:Example1.com* key generator tModel created the *uddi:Example1.com:finance* key generator tModel (as discussed above), then immediately created a service with the key *uddi:Example1.com:finance:payroll*. This is legitimate; this person is creating a key while owning the key generator tModel for the key partition the key is in. Then, the owner transfers custody of the *uddi:Example1.com:finance:keygenerator* tModel to someone else (us, in the example above). Does this mean that they lose custody of the service *uddi:Example1.com:finance:payroll*? No. Does this mean that the new owner of the key partition can modify the service *uddi:Example1.com:finance:payroll*? No. Does this mean that the new owner of the key partition can do anything with that key? No, nothing at all, because the key is already in existence, and the entity with that key is owned by someone else. The owner of the key partition only controls the creation of new keys in the key partition, nothing more.

2.3.6 An anomaly

Interestingly, in the situation above the owner of the *uddi:Example1.com:finance* key partition is entitled to create a new key generator tModel with the key *uddi:Example1.com:finance:payroll:keygenerator*. If they do so, they can create UDDI entities with keys like *uddi:Example1.com:finance:payroll:wages*. This may seem a little absurd, that one person may own the UDDI entity with the key *uddi:Example1.com:finance:payroll*, while another person owns the key partition *uddi:Example1.com:finance:payroll*, but it is a consequence of the fact that the ownership of the *uddi:Example1.com:finance:payroll* key partition is not conveyed by owning the entity with that key, but rather through owning the tModel with the key *uddi:Example1.com:finance:payroll:keygenerator*.

Users should take care to avoid creating a key partition ownership structure that will produce maintenance challenges in the future by splitting key partitions across too many organizational boundaries.

2.4 Discussion

2.4.1 Assigning key generator keys

Key generator keys should ensure uniqueness, be understandable, and be consistent. In certain cases, the best practice differs between a root key partition and a child partition.

Root key partition uniqueness can be achieved by using the domain name of the controlling organization as the key generator key. In the example above, Example1 uses "Example1.com" as the root partition, which ensures that if its UDDI registry is later combined with that of another company (for example, in a merger or acquisition scenario or when producing a community or partner (extranet) directory), its key partition namespace will not conflict with that of the other company. Using the domain name also makes it clear that the partition is the highest-level partition for that organization.

For child partitions, key partitions at the same level should not overlap in meaning. For example, using Example1.com:finance and Example1.com:payables as two key partitions is undesirable, because the payables department is logically a part of the finance department. A better solution would be to use Example1.com:finance and Example1.com:finance:payables.

2.4.2 Domain owners vs. publishers

Sometimes, the most appropriate key name for an entity is in a key space outside a person's organization. For example, if the Example1 enterprise architect wants to have a classification system based on the ANZSIC industry classification scheme that is owned by the Australian Bureau of Statistics then he should request the scheme URI from ABS and publish the key accordingly (i.e. uddi:abs.gov.au:anzsic rather than uddi:Example1.com:anzsic).

2.5 Example

The Example1 company decides that they will institute their own UDDI registry to serve as an integration point for the company's Web services.

They publish, as one of their first actions, a key generator tModel upon which they will base every key in their registry. They choose to use a domain key based on their company name: "uddi:example1.org", mainly because this seems like a fairly unique key, unlikely to be confused with any other company's key. It is sensible to keep domain key generator keys fairly short, because it is likely that a hierarchy of keys will be created from them.

They publish it by issuing the call:

```
<save_tModel xmlns=”urn:uddi-org:api_v3”>
  <authInfo registry-specific />
  <tModel tModelKey="uddi:example1.org:keygenerator">
    <name>Example1 Key Generator</name>
    <categoryBag>
      <keyedReference
        tModelKey=”uddi:uddi.org:categorization:types”
        keyValue=”keyGenerator”
        keyName=”keyGenerator”
      />
    </categoryBag>
  </tModel>
</save_tModel>
```

They further choose to publish key generator tModels for some of their divisions, including both the Human Resources and Finance divisions. They publish those two tModels in a single save call:

```
<save_tModel xmlns=”urn:uddi-org:api_v3”>
  <authInfo registry-specific />
  <tModel tModelKey="uddi:example1.org:hr:keygenerator">
    <name>Example1 Human Resources Division Key Generator</name>
    <categoryBag>
      <keyedReference
        tModelKey=”uddi:uddi.org:categorization:types”
        keyValue=”keyGenerator”
        keyName=”keyGenerator”
      />
    </categoryBag>
  </tModel>
  <tModel tModelKey="uddi:example1.org:finance:keygenerator">
    <name>Example1 Finance Division Key Generator</name>
    <categoryBag>
      <keyedReference
        tModelKey=”uddi:uddi.org:categorization:types”
        keyValue=”keyGenerator”
        keyName=”keyGenerator”
      />
    </categoryBag>
  </tModel>
</save_tModel>
```

In this company, the Payroll section comes under Finance, but has strong ties to Human Resources. It is decided that this section will maintain its own web services, and receive key generator tModels from both Human Resources and Finance key partitions – this will enable them to generate keys that are appropriate to the entities that they are publishing: using keys from the HR partition for web services that are more HR related (such as web services

supporting an employee's ability to change banking details), and keys from the Finance partition for web services that are more Finance related (such as the famous web service that supports altering an employee's salary). This is not a requirement – the section only needs a single key generator to be able to publish, but it is appropriate to do things this way.

The HR division creates a key generator tModel for the Payroll section by saving:

```
<save_tModel xmlns=”urn:uddi-org:api_v3”>
  <authInfo registry-specific />
  <tModel tModelKey="uddi:example1.org:hr:payroll:keygenerator">
    <name>Example1 HR: Payroll Section Key Generator</name>
    <categoryBag>
      <keyedReference
        tModelKey=”uddi:uddi.org:categorization:types”
        keyValue=”keyGenerator”
        keyName=”keyGenerator”
      />
    </categoryBag>
  </tModel>
</save_tModel>
```

The Finance Division creates a key generator tModel for the Payroll section by saving:

```
<save_tModel xmlns=”urn:uddi-org:api_v3”>
  <authInfo registry-specific />
  <tModel tModelKey="uddi:example1.org:finance:payroll:keygenerator">
    <name>Example1 Finance: Payroll Section Key Generator</name>
    <categoryBag>
      <keyedReference
        tModelKey=”uddi:uddi.org:categorization:types”
        keyValue=”keyGenerator”
        keyName=”keyGenerator”
      />
    </categoryBag>
  </tModel>
</save_tModel>
```

The two divisions must transfer these new key generator tModels to the payroll section before they can be used by that section to create new keys.

The examples above are functional, but it is likely, in real-life, that the key generator tModels might have overviewDoc elements containing pointers into an overall document explaining the keying system adopted by Example1. It is also highly likely that these key generator tModels will be digitally signed, because those signatures provide an additional evidence of the identity of the publisher, and of the authenticity of the tModels.

3 References

[This section should list any references to publicly available documents that the reader may find helpful during reading of this Technical noted document. These documents may expand upon any aspect of the material, for instance they may provide additional insight into the business situation dealt with or they may document standards or products used in developing the solution.]

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.

[RFC4122] P. Leach, et al, A Universally Unique Identifier (UUID) URN Namespace, http://www.rfc-archive.org/getrfc.php?rfc=4122, IETR RFC 4122, July 2005

Appendix A. Acknowledgments

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

Luc Clément, Systinet

John Colgrave, IBM

Matthew Dovey

Jason Garbis, Systinet

Andrew Hately, IBM

Rob Kochman, formerly Microsoft

Paul Macias, LMI

Oleg Mikulinsky, WebLayers

Tony Rogers, Computer Associates

Claus von Riegen, SAP AG

Additional contributors include:

Steve Capell, Red Wahoo Pty Ltd
Pat Felsted, Novell Inc
Ed Mooney, Sun Microsystems Inc
Dave Prout, BT plc

Appendix B. Revision History

Rev	Date	By Whom	What
	16 October 2005	Tony Rogers	Integrated feedback, Inserted 2.2.1 as an introduction
	20 February 2006	Rob Kochman	Filled in Section 2.4, removed header for security policy section, changed “WWW” example to “Example1”, incorporated other feedback from Dave Prout’s April 2005 email.
	2 May 2006	Luc Clément	Edit pass
	23 May 2006	Andrew Hately	Edit pass
	22 August 2006	Andrew Hately	Additional, hopefully final edit pass incorporating all feedback
	11 November 2006	Luc Clément	Incorporated input from Paul Denning, Mitre Corp

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.