



WebSphere® Application Server V5.1

UDDI Utility Tools

WebSphere software



 business on demand software

Updated November 1, 2003

© 2003 IBM Corporation

Agenda

- Overview of UDDI
- Overview of UDDI Utility Tools
- Tool Commands
- Programmatic APIs
- Security
- Problem Determination
- Summary

Section

UDDI : Overview

UDDI Basics

- Universal Description, Discovery, and Integration
- UDDI registries are used to advertise and find services and businesses on the web
- UDDI specification provides standardized formats for programmatic business and service discovery
- UDDI specification can be found at <http://www.uddi.org/>



This presentation is primarily going to focus on the technical details involved with using the UDDI Utility Tools provided with WebSphere Application Server Deployment Manager 5.1. However, before starting this discussion it is useful to review some of the basic UDDI concepts to make sure we are familiar with the UDDI Information model and architecture before looking at the features the utility tooling provides. This section will cover the UDDI basics.

UDDI stands for “Universal Description, Discovery, and Integration”. The UDDI project is aimed at providing a universal way for businesses to publish details and descriptions of the web services they provide, as well as the ability to query these services and those published by other businesses.

This presentation does not outline the UDDI specification in detail, however the key points about the specification is that the UDDI project solves the problem of universal business/service description and discovery by specifying a standard:

1. information model which is used to describe the data (entities) that is stored in the UDDI Registry
2. API for programmatically manipulating and working with data stored in the UDDI registry

The result is that the UDDI architecture makes registering and locating web services more flexible and standardized then using a direct publish approach whereby the service details are exchanged via email or FTP.

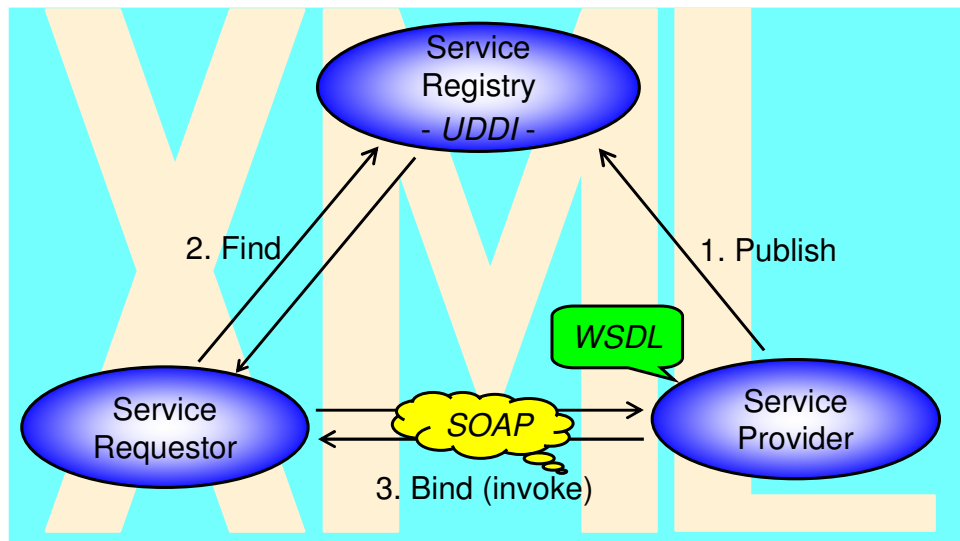
The entire UDDI specification can be found at www.uddi.org, along with other UDDI resources.

IBM Confidential

WASv51_UDDI_UtilityTool.ppt

Page 4 of 38

Relating UDDI to Web Services



Note: Service Registry is Optional. The Requestor may get WSDL from other places than the Registry

5

UDDI Utility Tools

© 2003 IBM Corporation

Before walking through an example scenario relating UDDI and Web Services, let's review some key points about UDDI that will help explain the picture in this slide.

- The UDDI architecture is built upon a number of widely expected standards-based technologies, such as TCP/IP, HTTP, XML, and SOAP
- All UDDI data structures (entities) are described using XML
- At the heart of the UDDI specification is a SOAP-based API

Before a business can start publishing and locating services on the web, standards bodies populate the registry with standard service descriptions (these are called technical models). Once this foundation has been built, service providers can add the services they provide to the UDDI business registry (step 1). When the service provider does this, the UDDI business registry assigns a unique identifier to each service published. Once the services have been added to the UDDI business registry, service requesters such as online marketplaces, search engines, etc. can query the registry to find the services they need (step 2). Finally, the data that is returned to the service requester is used to invoke methods on remote Web Services (step 3).

UDDI Entities

- Core UDDI Data Structures
 - Who? → businessEntity
 - What? → businessService
 - Where? → Service Bindings or bindingTemplate
 - How? → Technical Model or tModel
- Bags –
 - identifierBag such as D-U-N-S number
 - categoryBag such as taxonomy-based classification schemes



There are four primary entity types defined by the UDDI specification. The following is a description of these core UDDI Entities:

businessEntity – The businessEntity describes who (business or organization) provides a set of web services. A businessEntity does not necessarily have to represent a business, rather the businessEntity could be used to describe any parent service provider (for example, a department).

businessService – The businessService entity is a child of the businessEntity, and encapsulates a list of business services offered by the organization described by the parent businessEntity.

bindingTemplate – The bindingTemplate represents an individual web service. The bindingTemplate defines the technical information needed by applications to bind and interact with a web service. Note: A bindingTemplate can be the child of only one businessService.

tModel – A tModel (technical model) describes a reusable concept within the UDDI information model. Some examples of a tModel would be: a Web service type, a protocol used by Web services, or a category used to catalog services.

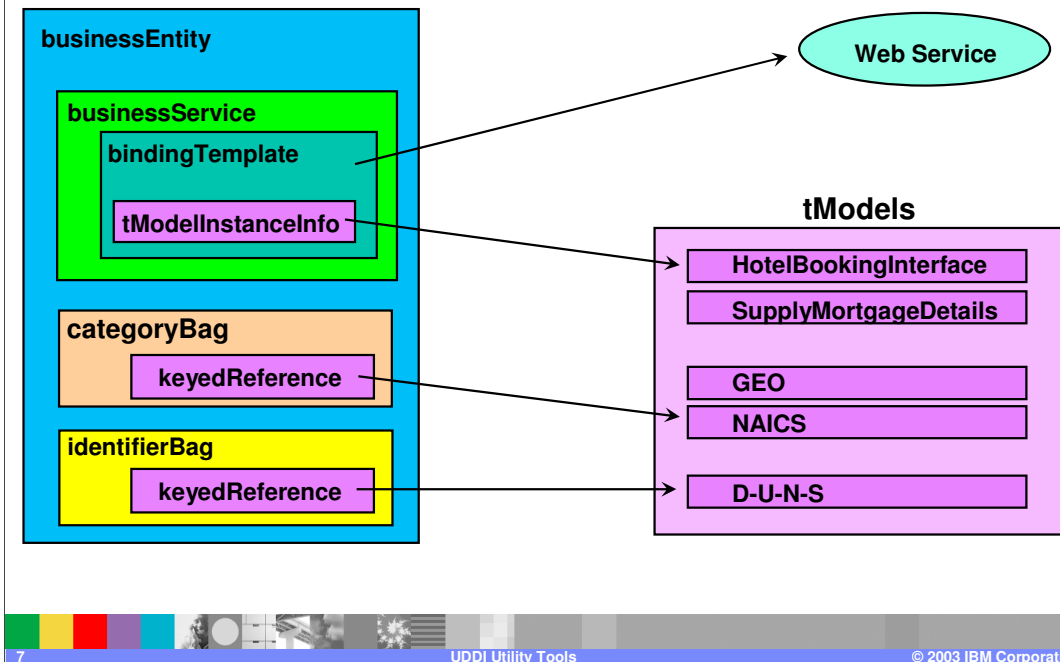
As mentioned previously, UDDI entities are described using XML, and make up that data that is stored in the UDDI registry.

IBM Confidential

WASv51_UDDI_UtilityTool.ppt

Page 6 of 38

UDDI Information Model



When discussing the UDDI information model, it is helpful draw a picture relating the primary UDDI entities. This slide shows the relationship between the primary UDDI entities. A further description of the core entities is listed below.

Each business entity contains the following information: a unique identifier (**businessKey**), the name of the organization/business (**name**), a short description of the organization, contact information (**contact** - description, persons name, phone, email, address), sets of related web services provided by the organization (**businessServices**), a list of categories (**categoryBag**), identity information (**identifierBag**), URL pointing to more information about the organization (**discoveryURL**), digital signature (**signature**).

As indicated from the picture on this slide, a business service is associated with a business entity. Each business service entry contains: the name for the set of services (**name**), a business description of the set of services (**description**), a list of categories that describe the service (**categorybag**), a list of pointers to references and information related to the services (**bindingTemplate**), and a digital signature (**signature**).

Associated with each business service entry is one or more binding templates that point to specifications and other technical information about a service. A binding template describes an instance of a web Service. A **bindingTemplate** includes: a description of the service (**description**), technical details for the web service (**tModelInstanceDetails**), information used to categorize the web service (**categoryBag**), a network address that can be used for accessing the web service (**accessPoint**), and a digital signature (**signature**).

A service type is defined by a **tModel**. A **tModel** specifies information such as: the name of the technical model, the name of the organization that published the technical model (**identifierBag**), a list of categories that describe the service type (**categoryBag**), pointers to technical specifications for the service type such as interface definitions, message formats, message protocols, and security protocols (**overviewDoc**), a description of the technical model (**description**), and a digital signature (**signature**).

IBM Confidential

Public and Private UDDI Registries

- Public Registries
 - Also referred to as the UDDI Business Registry (UBR)
 - Currently IBM, Microsoft, SAP, and NTT are UBR operators
 - Anyone in the world can register with or search a UBR for businesses and services.
- Private Registries
 - Resides within intranets, extranets, or private networks
 - Compliant with UDDI specification
 - WebSphere Application Server V5 Network Deployment provides a private UDDI Registry as an optional component



There are two flavors of UDDI registries, public and private.

Public registries can be used by anyone to publish or search for businesses and services. A public registry is sometimes referred to as a UDDI Business Registry (UBR). The current list of UBR operators includes: IBM, Microsoft, NTT, and SAP. It is important to note that the data in all of the registries run by the UBR operators is replicated to the other operator nodes. The IBM UBR is accessed at: <http://www-3.ibm.com/software/solutions/webservices/uddi/>. To find out more information about the operators of UBR nodes, visit <http://www.uddi.org/find.html/>.

Private UDDI registries are registries that are compliant with the UDDI specification, but reside within intranets, extranets, or private networks. These registries aim to facilitate business application integration within a department, within a company, or between business partners, etc. With a private UDDI registry you have more control over the administration and content in the registry.

IBM WebSphere UDDI Registry Key Points

- The IBM WebSphere UDDI Registry is compliant with UDDI v2 specification
- The IBM WebSphere UDDI Registry can be installed in a Network Deployment or Single Server environment
- UDDI Taxonomy Tools allow management of user defined categories ←5.0.2
- UDDI Utility Tools used to manage entities in a UDDI registry ← 5.1



The private UDDI registry packaged with IBM WebSphere Network Deployment Manager is compliant with the UDDI v2 specification. The UDDI registry is an enterprise application that can be installed in a Single Server environment or in a Network Deployment scenario. In a production environment, DB2 must be used as the persistent store. However, in a development or test environment, Cloudscape can be used.

Starting with V5.0.2, the IBM WebSphere UDDI Registry provides a tool that allows the management of custom taxonomies. This tool provides the ability to create, delete, or rename custom taxonomies in the UDDI registry.

Finally, new in V5.1, the IBM WebSphere UDDI Registry provides a suite of UDDI Utility Tools that allow management of entities in a UDDI registry. The utility tools provide a command line interface as well as APIs to provide export, import, promotion, delete, and find functions for UDDI entities. The remainder of the presentation will focus on reviewing the functions provided by the UDDI Utility Tools.

Taxonomy Overview

- Taxonomies are used to categorize UDDI entities
- IBM WebSphere UDDI Registry supports 6 published taxonomies
- Support for user defined taxonomies was introduced in version 5.0.2 of WebSphere Application Server
 - Publishers of user defined taxonomies can specify a GUI display name



Before moving into the new features provided by the UDDI Utility Tools, let's review the functions provided with the taxonomy tooling.

A taxonomy is used to catalog UDDI entities in the registry. Categorizing UDDI entities becomes increasingly important as the number of entities in the UDDI registry grows. The more categorizations referenced by an entity, the more precisely the entity is defined. Entities that are precisely defined provide a better chance of being discovered and utilized. The benefit of providing custom taxonomy support is that this allows providers the ability to better describe UDDI entities. As the usage of UDDI registry grows, categorizations will help to narrow searches and reduce the size of a result set from a given search.

IBM WebSphere UDDI Registry supports 6 published taxonomies. These include:

1. ntis-gov:naics:1997 → North American Industry Classification System - (Business Taxonomy: NAICS (1997 Release))
2. uddi-org:iso-ch:3166-1999 → ISO 3166 Geographic Code System - (Description ISO 3166-1:1997 and 3166-2:1998. Codes for names of countries and their subdivisions. Part 1: Country codes. Part 2: Country subdivision codes. Update newsletters include ISO 3166-1 V-1 (1998-02-05), V-2 (1999-10-01), ISO 3166-2 I-1 (1998))
3. unspsc-org:unspsc → United Nations Standard Products and Services Code System - (Product Taxonomy: UNSPSC)
4. unspsc-org:unspsc:3-1 → United Nations Standard Products and Services Code System - (Product Taxonomy: UNSPSC (Version 3.1))
5. uddi-org:types → UDDI Types Taxonomy - (UDDI Type Taxonomy)
6. uddi-org:general_keywords → General Keywords Taxonomy - (Special taxonomy consisting of namespace identifiers and the keywords associated with the namespaces)

In the above list, the taxonomy name is given first (short name → full name), followed by a description in parenthesis. For more information about these taxonomies refer to http://uddi.org/taxonomies/UDDI_Taxonomy_tModels.htm.

Custom taxonomies can be introduced into the IBM WebSphere UDDI Registry using the UDDITaxonomyTools.jar utility. It is also possible to specify a display name that is used as a user friendly label for the taxonomy instead of using the long name taken from the tModel's field name. IBM WebSphere UDDI Registry custom taxonomy support prohibits deleting or renaming the internal names for the 6 published taxonomies listed above.

IBM Confidential

WASv51_UDDI_UtilityTool.ppt

Page 10 of 38

Creating a Custom Taxonomy

- Step 1 – Import custom taxonomy data
 - Use the UDDITaxonomyTools.jar utility to import the taxonomy data
- Step 2 – Publish a categorization tModel with a categoryBag containing keyReference as follows:

tModel Key	KeyName	Key Value
(uddi-org:types)	<optional>	categorization
(uddi-org:types)	<optional>	checked
(general keywords)	urn:x-ibm:uddi:customTaxonomy:key	<Custom Taxonomy Name>
(general keywords)	urn:x-ibm:uddi:customTaxonomy:displayName	<Custom Taxonomy Display Name>



There are two steps to creating a custom taxonomy. In the first step the user will use the UDDITaxonomyTools.jar utility to import the taxonomy data. The usage for the Taxonomy tool is shown below.

Usage: java -jar UDDITaxonomyTools.jar {function} [options]
function:

-load <path> Load taxonomy data from specified file
-rename <old> <new> Rename existing taxonomy
-unload <name> Unload existing taxonomy

options:

-properties <path> Specify location of configuration file

The -load option is used to import custom taxonomy data in step one of the create process. The load option requires that a path be specified that points to an input file containing the custom taxonomy data. Taxonomy lines of data specified in the input file must conform to the following format: <name>#<code>#<description>#<parent code>. The '#' is a delimiter for the values (The '#' delimiter is configurable). The name is a unique identifier for the custom taxonomy. The code is a value with in the taxonomy used for validation. The description is typically used by the GUI as the user friendly way of describing the code value. The parent code indicates which existing code is the parent of this one. An example input file might look like this:

```
plants#00#Plants#00
plants#10#Flowers#00
plants#101#Rose#10
plants#102#Daisy#10
plants#20#Trees#00
plants#201#Maple#20
```

The tree representation might look like this:

```
→Plants
  →Flowers
    →Rose
    →Daisy
  →Trees
    →Maple
```

In the second step the custom categorization tModel needs to be published in the UDDI registry. This can be done using the SOAP API function save_tModel, or you can use the IBM WebSphere UDDI Registry GUI to publish this tModel.

NOTE:

It is recommended that the custom taxonomy data file described above be in a UTF-8 format without header bytes. WebSphere Studio Application Developer is an ideal choice for producing this file in the proper format. The data file also has maximum lengths associated with each column of data. These are: 8, 32, 128, 32 for name, code, description, and parent code, respectively. A new taxonomy may be rejected if the specified name conflicts with the name of a core taxonomy (for example, naics, geo, or unspsc7).

IBM Confidential

WASv51_UDDI_UtilityTool.ppt

Page 11 of 38

Section

UDDI Utility Tools: Overview

UDDI Utility Tools - Overview

- UDDI Utility Tools offer the following functions to manage entities in a UDDI Registry
 - Export
 - Import
 - Promote
 - Delete
 - Find Matching Entities (only available via API)
- These functions are available via command line utility or APIs
- Utility tool supports functions not available in the UDDI API



The UDDI Utility Tools offer a number of functions that can be used to manage UDDI entities between a target and a source UDDI registry. Functions included in UDDI Utility Tools are individually described below.

The **export** function takes an entity type and key OR a list of entities and keys as input, and then returns the UDDI entity information from the specified registry. The export function outputs the entity data to an Entity Definition File, which is in XML format. The export function acts on a **source** registry.

The **import** function takes a list of UDDI entities supplied in an Entity Definition File (EDF) and checks for the existence of any of the entities in the target registry. If an entity already exists, it may be updated with entity information from the EDF, depending on the value of the overwrite property (discussed further in a later slide). If the entity doesn't exist, a minimal entity is created with the specified entity key, and this is updated with the full entity data from the EDF.

The **promote** function allows entities to be exported from a source registry and imported into a target registry in one step.

The **delete** function takes an entity type and key, and then deletes the specified entity from the target registry.

The **find** function is only available via the utility tool API. As input, the find function takes a search criteria in the format of a UDDI inquiry API object for the entity types. The result is a list of keys that can be used with the other utility functions.

Information about the utility tool is on the infocenter under:

"WebSphere Application Server Network Deployment"→Administering→Applications→Web Services→Administering UDDI Registry→UDDI Utility Tool

IBM Confidential

WASv51_UDDI_UtilityTool.ppt

Page 13 of 38

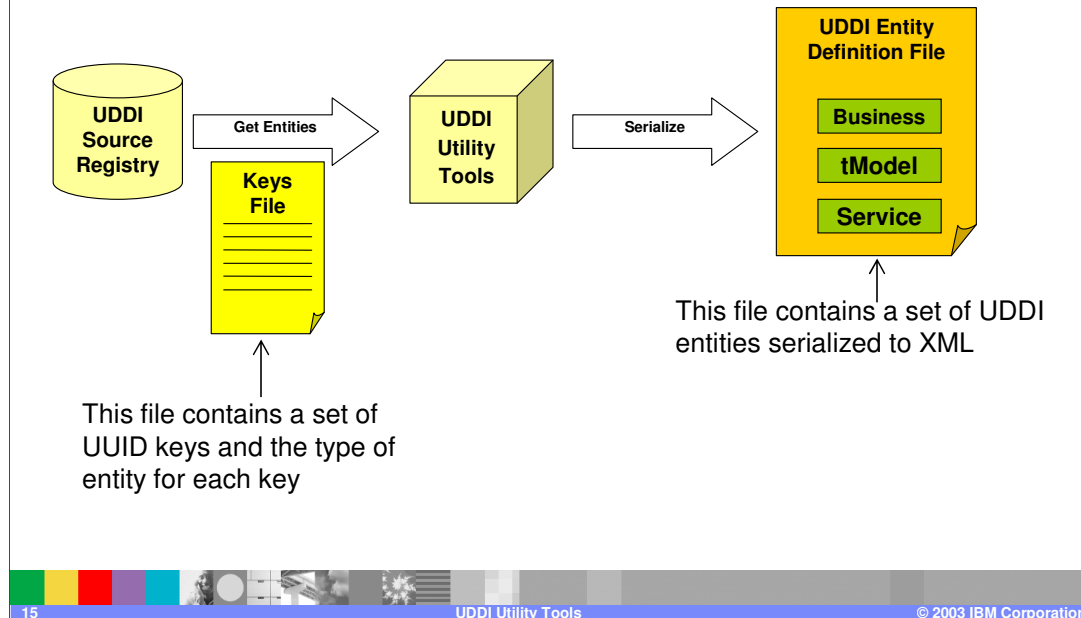
UDDI Utility Tools - Overview

- UDDI Registry that conforms to UDDI v2 specification
- If you were using UDDI APIs, data exported from one UDDI registry and imported in another registry would be assigned a different UUID
 - Tools will preserve the same UUID from the source Registry



The UDDI Utility Tools conforms to the UDDI v2 specification. One of the key advantages to the UDDI tooling is that it provides the ability to preserve the UUID from the source to the target registry. This means that the UDDI Utility Tools supports what is referred to as entity promotion.

How it works - Export

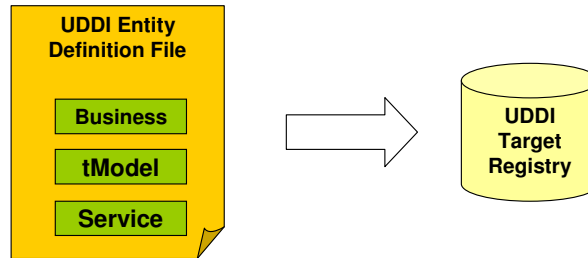


Before getting into the details using the command line interface and APIs it is helpful to get a high level understanding of how the tool works. This slide provides an overview of the export function.

A keys file is used as input into the export function to specify the set of UUID keys and the type of entity for each key that is to be exported from the UDDI source registry. With the data specified in the keys file, the export function retrieves the entity data from the source registry and writes this data to a UDDI Entity definition Data File. This file contains the set of UDDI entities taken from the source registry serialized to XML.

An example keys file will be shown later in this presentation.

How it works - Import



The Entity Definition File (EDF) is used to import data into one or more UDDI registries

After an export function has generated a UDDI Entity Definition Data File, this data can be used as input into the import function to load the data into a target registry. This two step process (export + import) provides the option to edit the Entities exported from the source registry before importing into the target registry. Although the EDF is typically generated from running an export command, it should be pointed out that this file may also be created by hand. Creating an EDF by hand may be useful when loading new or updated canonical tModels.

Note: If there is no need for this intermediate step, the promote function will perform an export followed by an import in one step.

UDDI Utility Tools – Examples

- Allow UDDI entities to be moved from one registry to another, while preserving the key associated with the entities and any child entities
- Allow entities with a specified key to be created in a UDDI registry
- Persist UDDI entities in the intermediate entity definition file so that the entities can be customized and imported into multiple registries.



This slide highlights several common ways customers will use the UDDI Utility Tools. However, a more complete list of usage scenarios is published in the infocenter along with example usage statements for the command line interface, and code examples for the APIs.

Information about the utility tool is on the infocenter under:

“WebSphere Application Server Network Deployment”→Administering→Task overview→>IBM WebSphere UDDI Registry→UDDI Utility Tools

UDDI Utility Tools - Advantages

- Allows easy promotion of UDDI entities from one registry to another, while maintaining the entity key
 - Services can be published and updated from a development to a production registry
- Allow customization of entities before they reach the target registry
 - Via exporting selected entities to an intermediate Entity Definition File
- Possible to promote any of the four fundamental UDDI entity types, including child entities
 - businessEntity, businessService, bindingTemplate and tModel
- Services can be inserted or updated in an existing businessEntity without affecting other services in that businessEntity



One of the main features that the UDDI Utility Tools offers is that it allows you to promote UDDI entities from one registry to another while maintaining the same entity key. The advantage of this feature is that services can first be tested in a development environment, and later promoted to a development environment. Another point to make here is that the UDDI Utility Tools make it easy to promote UDDI entities because it automatically discovers all the tModels referenced by entities and adds them to the referenced tModel section of the Entity Definition File. This saves the user a substantial amount of time and effort, because without this functionality, the referenced tModels would need to be imported separately.

The UDDI Utility Tools offer another advantage by allowing entities to be customized before reaching a target registry. It is possible before services are moved from a development to production environment it may be necessary to update various information about the services. This scenario is allowed with the UDDI utility tooling by exporting entities to an intermediate EDF.

Additionally, the UDDI Utility Tools also allow you to promote any of the primary UDDI entity types, including child types. This means that it is possible for services to be inserted or updated in an existing businessEntity without affecting other services in that businessEntity. Using the normal UDDI API, to update a businessEntity's service would require re-publishing the businessEntity and all its existing services, otherwise the other services would be lost. With the UDDI Utility Tools it is easy to specify new and updated services without having to embed them in a businessEntity structure.

Section

UDDI Utility Tools: Commands

UDDI Utility Tools - Commands

■ Invocation

- **java -jar UDDIUtilityTools.jar <function> <options>**
- **Reads information from properties file – Default is UDDIUtilityTools.properties**

■ Functions

- **-export <entity type> <key> OR -export -f <keys-file>**
- **-delete <entity type> <key> OR -delete -f <keys-file>**
- **-promote <entity type> <key> OR -promote -f <keys-file>**
- **-import -definitionFile <Entity-Definition-File>**

<entity-type> = -business | -service | -binding | -tModel



In order to run the UDDI Utility Tools, the locations of the following jar files must be indicated in the properties file:

- UDDIUtilityTools.jar
- uddi4jv2.jar
- db2java.zip
- j2ee.jar
- soap.jar
- xerces.jar

For the export, promote, and delete functions you can specify the entity type and key as a command line option, or alternatively you can choose to specify a keys file to specify the UDDI entities you wish to select for this function. The advantage of specifying a keys file is that you can select more multiple entities to export, promote, or delete. An example of what a keys file looks like is shown below.

Example keys file

businesses=97C77097-AC6C-4CA0-A6C4-452F7045C470, 4975E949-581F-4FCA-AD5F-E08280E05F9F

services=BB3864BB-1578-4833-8179-14391F14791F

bindings=

tModels=273F1727-7BFF-4FB5-A1FD-BA5C45BAFD9C

End Example keys file

NOTE: The keys file is simply a properties file. This means that the property names (bindings, tModels, businesses, and services) are case sensitive.

IBM Confidential

WASv51_UDDI_UtilityTool.ppt

Page 20 of 38

UDDI Utility Tools - Options

Option	Description
-properties <filename>	Specifies path to properties file if not using the default UDDIUtilityTools.properties
-log -v	Output verbose messages
-overwrite -o	Overwrite an entity if it already exists
-definitionFile <filename>	Specify path to UDDI entity definition file
-importReferenced	Import entities referenced by source entities



The `-definitionFile` option can be specified on an export call. This tells the UDDI Utility Tools to export the entity definition file to the specified file. If this option is not specified, then data is output to the file specified by the `UddiEntityDefinitionFile` property in the properties file. Likewise on an import, if this option is not set, the default behavior is to read the entity definition file specified by the `UddiEntityDefinitionFile` property.

Note: The following options override property settings in configuration file:

`-overwrite` , `-log`, `-definitionFile`, `-importReferenced`

UDDI Utility Tools – Properties File

- classpath
 - to various WebSphere jar files needed by the Utility tool
- fromInquiryURL and fromGetURL
 - The source registry for data export
- toInquiryURL and toPublishURL
 - The target registry for data import
- userID and password
 - Identify the owner of the data in the target registry
 - UNAUTHENTICATED and NONE should be used if the target registry is unsecured
 - userID must match the userID of the entity being updated
- dbDriver, dbUrl, dbName and dbPasswd
 - Used to access the target registry database.



Also included in the properties file are options to set:

1. Log and Trace levels
2. Log and Trace file paths
3. Security Provider information
4. Overwrite and Import Referenced Entities flags
5. Default location for the entities definition file
6. Namespace prefix

In case of failure

- Import step will not leave partially published entities in the target registry
 - If there is a failure during the operation, incomplete transactions should be rolled back

During an import or promote request, the UDDI Utility maintains data integrity if there is a failure during the operation. Any transactions that have not been committed will be rolled back, to avoid leaving partially published entities in the target registry.

Section

UDDI Utility Tools: Programmatic APIs

Programmatic API – PromoterAPI

- Public API to drive five primary functions – export, promote, import, delete, find
- Here is a typical usage for PromoterAPI:
 - Create a Configuration object with the desired properties
 - Create a PromoterAPI object with the Configuration instance
 - Set the entity keys or UDDI4J entities to act on using the setter method
 - Invoke the function (export, promote, import, delete, find) required



<TODO: Add a note about where you can get the javadoc once product is installed>

The `com.ibm.uddi.promoter.PromoterAPI` class is the key class that exposes the high level methods for performing the export, promote, import, delete, and find functions on UDDI entities.

This slide shows the typical steps needed to use the PromoterAPI. Here are some extra notes about the steps listed in the slide:

Step 1 – The Configuration class is `com.ibm.uddi.promoter.config.Configuration`. This class has 2 constructors. One takes a `java.util.Properties` object, and the other takes a String representing the fully qualified path name to the Properties file. The properties that can be set when creating a Configuration object are the same as those listed previously when discussing the UDDI Utility Tools.

Step 2 – The PromoterAPI class has only one constructor that takes the Configuration object created in Step 1.

Step 3 – A key and entity type are required in order to use the export, delete, or promote methods. Therefore the key and entity need to be set prior to calling these methods. There are 3 methods available for setting this information. These methods are:

- `setUddiEntities(String filename)` – Reads UDDI entity keys from a file.
- `setUddiEntities(UddiEntityKeys entityKeys)` – Sets the UDDI keys based on the `com.ibm.uddi.promoter.UddiEntityKeys` object. The `com.ibm.uddi.promoter.UddiEntityKeys` class is a container class used to manage lists of keys for each of the entity types.
- `setUddiEntity(java.lang.String entityType, java.lang.String entityKey)` – Sets the entity type and entity key if there is only one that is being processed.

The import method uses the Entity Definition File specified in the Properties object to import the entities, and does not require setting any of the above methods. Likewise, the find functionality does not require setting these methods prior to making the `extractKeysFromInquiry()` call.

Step 4 – The export, import and delete methods do not take any parameters. These methods are `exportEntities()`, `importEntities()`, and `deleteEntities()`, respectively. The `promoteEntities()` method takes a boolean which is used to indicate whether information should be written to an intermediate Entity Definition File. The find functionality is provided by the `extractKeysFromInquiry()` method. This method takes 5 UDDI4J find objects as arguments. For more information see the javadoc for the PromoterAPI, and UDDI4J packages.

The infocenter has several example programs showing key API functionality, as well as demonstrating the steps listed on this slide for API usage. This information can be found by following “WebSphere Application Server Network Deployment” → Administering → Task overview → IBM WebSphere UDDI Registry → UDDI Utility Tools

IBM Confidential

WASv51_UDDI_UtilityTool.ppt

Page 25 of 38

PromoterAPI – Method Summary

- Export – exportEntities()
 - Promote – promoteEntities()
 - Delete – deleteEntities()
 - Import – importEntities()
 - Find – extractKeysFromInquiry()
- After invoking:
setUDDIEntities(keysFile) OR
setUDDIEntity(entity type, key)
- The UDDI Entity Definition
File is specified in the
configuration object

Here are some example classes to be familiar with when using the PromoterAPI:

- com.ibm.uddi.promoter.UddiEntityKeys
- com.ibm.uddi.promoter.UddiEntities - a wrapper for Lists of the various UDDI4J types. It defines useful method for converting UDDI4J objects into corresponding entries in the UddiEntityKeys class.
- UDDI4J classes – set of java packages that provide classes used to interact with a UDDI registry

Note: The UddiEntityKeys class can write itself to a keys file or a Properties object which is useful for retaining the results of EntityFinder.findEntities()

Programmatic API – Beyond PromoterAPI

- Other classes are available for invoking the lower level methods directly – See Javadoc for details
 - EntityFinder
 - EntityDeleter
 - UddiSerializer



There are a number of lower level utility methods beyond the PromoterAPI that are also available to use. This slide points out several of classes that offer lower level methods. However, for a complete list of classes see the java documentation. The following is a description of the classes listed on this slide:

EntityFinder – Performs a find function on the source registry

EntityDeleter – Performs delete operations in the target registry given a list of entity keys

UddiSerializer – Transforms UDDI4J types to XML according the XML schema definition for UDDI Utility Tools

Sample java programs exercising some of the lower level functions are listed in the infocenter.

Section

UDDI Utility Tools: Security Requirement

Security Relating to UDDI Registry and Tools

- UDDI Registry application (uddi.ear) uses standard J2EE Security Role authorization user/group to role mappings
- When security is NOT enabled in WebSphere
 - A userid and password is not required
 - A userid is required if entity was published with a userid and security enabled
- When Security is enabled in WebSphere
 - userid and password will be required to import, promote or delete the data
 - userid and password will be required to export the data
 - Unless the role mappings have been changed for inquiry functions
- Data ownership needs to be taken into account when importing or deleting data within the registry



The IBM WebSphere UDDI Registry uses the standard J2EE Security Role authorization user/group to role mappings. Additionally, the UDDI Utility Tools will use the security access set up by the target registry.

When WebSphere security is not enabled, a userid and password is required. However on an import, promote or delete, if an entity was published when security was enabled using a userid of johndoe, then the userid must be johndoe even if security is subsequently turned off. Note that a typical userid/password is UNAUTHENTICATED/NONE.

Section

Problem Determination

UDDI Troubleshooting - General

- IBM WebSphere UDDI Registry may issue messages to report events or errors
- The UDDI message identifiers have the form UDxxnnnnns
 - **xx** – is a two letter descriptor to identify the component involved
 - **nnnn** – is the error code being issued
 - **s** – is either I for Informational or E for error
- Enabling trace for the UDDI Registry is done the same way you enable trace for any other WebSphere component
- List of component codes is available in the infocenter



General log messages are written to the SystemOut and SystemError logs.

Trace messages are written to: <WAS_INSTALL>/logs/<server>/trace.log

UDDI Utility Tools – Troubleshooting

- The UDDI Utility Tools messages have the form UDUT*nnnns*.
- The path to the log and trace files can be configured in the UDDIUtilityTools.properties file
- Trace and log levels can also be specified in the UDDIUtilityTools.properties file



The utility tool trace is not set through the WebSphere admin console, rather it is done through a properties file.

The following properties relate to log and trace:

- verbose → turn level of output on or off (values are true | false)
- traceLevel → detail level of trace output (1=severe, 2=normal, 3=detail)
- messageLogFileName → path to message log file (relative or absolute)
- traceLogFileName → path to trace log file (relative or absolute)

UDDI Troubleshooting Tips - General

- Verify that the database configuration is correct
- Common UDDI troubleshooting tips can be found in the infocenter



There is a detailed list of common problems that may be encountered while configuring/running the IBM WebSphere UDDI Registry published on the infocenter web site. To access this information go to the infocenter and navigate to “Administering→Applications→Web services →IBM UDDI Registry→UDDI Troubleshooting tips” from the WebSphere Application Server Network Deployment root.

A quick look at the list shows that a number of issues are often caused by database configuration problems.

UDDI Utility Tools – Limitations

1. Some entities from source registry are not automatically added to the EDF
 - a) Businesses referenced in service projections
 - b) Binding Templates referenced by hostingRedirectors
 - c) Businesses referenced by an owning business keyedReference
2. Cycles are not detected for service projections
3. tModels deleted in source registry are imported and promoted to target registry as undeleted
4. When security is enabled the JSSE provider must be com.ibm.jsse.IBMJSSEProvider



There are workarounds for each of the limitations listed in this slide. These limitations and workarounds are also listed in the infocenter under “Administering→Applications→Web services →IBM UDDI Registry→UDDI Troubleshooting tips” for WebSphere Application Server Network Deployment.

The workarounds for each item is listed here for convenience:

- 1a) Add the referenced business that will 'own' the projected service to the EDF. If the business is not present in the target registry, it should be placed before the service's owning business in the EDF
- 1b) Binding templates referenced by hostingRedirectors not included in EDF → Add bindingTemplate to EDF
- 1c) Businesses referenced by an owning business keyReference not included in EDF → Add the business into target registry and then import the tModel that references
- 2) If there is a circular reference between service projections, you may remove it temporarily and import the EDF without the circular reference. Follow this by an update action to introduce the cycle to the target registry.
- 3) Delete the tModel after it has been imported.

Section

Summary

Summary

- UDDI Taxonomy Tool allows management of user defined categories
- UDDI Utility Tools allows you to manage UDDI Registries especially when trying to migrate information across UDDI Registries
- Provides command line tool as well as Programmatic APIs

Trademarks and Disclaimers

© Copyright International Business Machines Corporation 1994-2003. All rights reserved.
References in this document to IBM products or services do not imply that IBM intends to make them available in every country. The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	iSeries	OS/400	Informix	WebSphere
IBM (logo)	pSeries	AIX	Cloudscape	MQSeries
e (logo)/business	xSeries	CICS	DB2 Universal Database	DB2
Netfinity	zSeries	OS/390	IMS	

Lotus, Domino, Freelance Graphics, and Word Pro are trademarks of Lotus Development Corporation and/or IBM Corporation in the United States and/or other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. UNIX is a registered trademark of The Open Group in the United States and other countries. Other company, product and service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Copyright International Business Machines Corporation 2003. All Rights reserved.
Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.