# XML based Advanced UDDI Search Mechanism for B2B Integration

Liang-Jie Zhang, Haifei Li, Henry Chang
*IBM T.J. Watson Research Center, Route 134, Yorktown Heights, NY 10598*
*{zhanglj,haifeili,hychang}@us.ibm.com*

## Abstract

*Exploring an appropriate business application published as a Web Service in the UDDI registry is a critical issue. Search for such an application should be effective in terms of time and uniformed in terms interfaces. In this paper, an XML-based UDDI exploring engine that provides developers with standard interfaces for efficiently searching business and service information in single or multiple UDDI registries is proposed. We refer this engine as "Business Explorer for Web Services" (BE4WS). It is based on the proposed UDDI Search Markup Language (USML) for carrying a search request including multiple queries, key words, UDDI sources, and aggregation operators. The Advanced UDDI Search Engine (AUSE), a core component of BE4WS, processes the USML request and performs advanced exploring. The basic idea of an AUSE is to aggregate search results from different UDDI registries based on the USML request and its supporting intelligent search facilities. Examples of these facilities are Instant Notification Broker, UDDI Source Dispatching Broker, and Information Aggregation Broker, all of which have both prior knowledge of the meanings of specific categories and the ability to cross-reference across multiple categories. An aggregation example using BE4WS is presented to create a shipping agent for B2B integration. The conclusions are given in the end of this paper.*

## 1. Introduction

The emergence of Web Services represents the next evolution of e-business [1]. Web services are Internet-based, modular applications that perform a specific business task and conform to a particular technical format. The technical format ensures each of these self-contained business services is an application that will easily integrate with other services to create a complete business process. This interoperability allows businesses to dynamically publish, discover, and aggregate a range of Web services through the Internet to more easily create innovative products, business processes and value chains. Exploring an appropriate business application published as a Web Service in the UDDI registry is a critical issue. Search for such an application should be effective in terms of time and uniformed in terms interfaces.

Web Services will be published to a public or private UDDI registry. The design of UDDI allows simplified forms of searching and allows trading partners to publish data about themselves, and their advertised Web services to voluntarily provide categorization data. In general, UDDI can locate businesses whose identities are well known so that users can find out what services they are offering and how to interface with them electronically. The current UDDI search is focused on a single search criterion such as business name, business location, business category, and service type by name, business identifier as well as discovery URL.

But at the same time, it is impractical to assume that the UDDI registry will be useful for general-purpose business search. With a projected near-term population of several hundred thousand to million distinct entities, it is unlikely that searching for businesses that satisfy a particular set of criteria will yield a manageably sized result set [2]. So it is a critical problem to come up with an efficient search engine for locating a proper Web Services. The second concern is the accuracy. When a specific category along with UDDI registration data is registered, only people searching for that exact category will find the results [2].

From an e-business application developer's point of view, it is necessary to send a few sequential or programmed search commands to UDDI registry for information aggregation. That is to say, the information sources may include multiple UDDI registries and other searchable sources. Obviously, there is a need to provide an advanced search mechanism for Web Services to dramatically extend the current search capability, which is based on category or key words, through its efficiency improvement and performance enhancement.

All existing UDDI search engines only support one single UDDI registry. For example, Microsoft's UDDI search technology just allows user search its UDDI registry using one single search criteria [4]. A single search criterion is based on one of the following categories: business name, business location, business category, and service type by name, business identifier, discovery URL. The known category and taxonomy types include NAICS, UNSPSC, SIC, a geographic code (GEO). The known identifier types include D-U-N-S, Thomas Registry numbers, and Tax ID.

In this paper, "Business Explorer for Web Services (**BE4WS**)" refers to searching businesses and services information from UDDI registries in an efficient and standard way. A UDDI Search Markup Language (USML) is defined for business applications to efficiently search UDDI registries. Based on the defined USML, a framework of Advanced UDDI Search Engine (AUSE) is proposed to conduct the searching process [3].

In this paper, the UDDI Search Markup Language (USML) is introduced in section 2. Then Advanced UDDI Search Engine (AUSE) is proposed to perform effective UDDI exploring process in an efficient manner in section 3. The USML construction and search criteria are described in section 4. In section 5, some aggregation operators are defined and illustrated. Further, two types of APIs are defined for **BE4WS** to allow developers to easily explore UDDI registries using standard interfaces in section 6. In section 7, a sample application of BE4WS is used to create an intelligent shipping agent to search UDDI registry based on two search criteria defined in a USML for B2B integration. The conclusions of this effective UDDI search mechanism are given in the end of this paper.

## 2. USML (UDDI Search Markup Language)

USML is an XML based language proposed to uniform the search query format and dramatically reduce requesting times in a search. An USML-based search request incorporates multiple search queries, UDDI sources and aggregation operators. Thus, it takes several criteria into account such as keywords to search for, identifiers, categories and so on for the desired search from a single or multiple registries.

As mentioned before, e-business application developers need to send a few sequential or programmed search commands to UDDI registries for information aggregation using regular UDDI client package such as UDDI4J [5]. That means, the information sources may include multiple UDDI registries and other searchable sources. Therefore, it is essential to provide an advanced search mechanism for Web Services server to dramatically extend the current search capability, which would provide efficiency improvement and performance enhancement.

USML would prove beneficial for such an advanced search mechanism with its ability to search on multiple criteria and from multiple registries as opposed to the simple search which searches on single criteria. As an XML-based language, USML will play a significant role in communications across system boundaries.

A sample USML sample is shown as follows:

```xml
<?xml version="1.0"?>
<!DOCTYPE Search SYSTEM "UDDISearch.dtd">
<Search>
        <ProcessId>9999</ProcessId>
        <Query>
                <Source>Private UDDI</Source>
                <SourceURL>http://wsbi10 /services/
                        uddi/inquiryAPI</SourceURL>
                <BusinessName >UPS</BusinessName >
                <FindBy>Business</FindBy>
        </Query>

        <Query>
                <Source>Public UDDI</Source>
                <SourceURL>http://wsbi5/services/
                        uddi/servlet/uddi</SourceURL>
                <BusinessName >KEP</BusinessName >
                <FindBy>Business</FindBy>
        </Query>

        <AggOperator>OR</AggOperator>
</Search>
```

Two search criteria are given in this sample USML. The first query is defined to look up a private UDDI registry (wsbi10) based on business name starting with *UPS*. The second query is defined to look up a public UDDI registry (wsbi5) based on business name starting with *KEP*. Then an aggregation operator *OR* is used to aggregate the search results from these two different UDDI registries. The aggregation result, an USML response, from BE4WS using this sample USML is presented in the following JSP (Java Server Page) page.

BE4WS Aggregation Result-> Business List

| Business: | Description: | Key: | Operator: |
|---|---|---|---|
| UPS | USA. UPS, or United Parcel Service Inc., is a 93-year-old company whose history spans the bicycle to the Internet. Today the largest express carrier and largest package carrier in the world, UPS continues to develop the frontiers of logistics. | 339CFD70-E84D-11D5-B61C-970107B90C83 | wsbi10/services/uddi |
| KEPEX Fulfillment Center | KEPEX | 8EF91680-EDDA-11D5-ADF1-850C07A41C41 | wsbi5:80/services/uddi |

The detailed schema and construction of a USML will be given in section 4.

## 3. XML based Advanced UDDI Search Engine (AUSE)

The basic idea of AUSE is to aggregate search results from different UDDI registries based on the proposed USML and its supporting intelligent search facilities such as Instant Notification Broker, UDDI Source Dispatching Broker and Information Aggregation Broker, which have both priori knowledge of the meanings of specific categories and the ability to cross-reference across multiple categories. The UDDI registries include one public UDDI registry and multiple private UDDI registries. The aggregation broker will parse and reorganize the returned results from different UDDI registries based on aggregation operators and rule-based scripts. The final result is represented as an XML response to the search requestor.

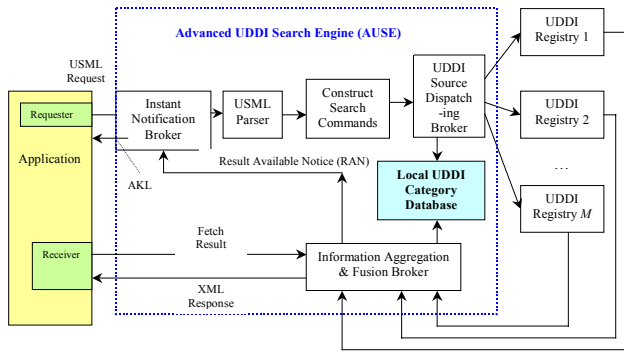The architectural diagram is shown in Figure 1.

**Figure 1 XML-based advanced UDDI Search System Architecture**

The following mechanism and ideas are used to enhance the UDDI search capability:

- Cascading Search Mechanism

It is used for refining the search results at different levels of granularity. For example, filtering mechanism can be applied to the search results that are returned from different UDDI registries. Service searchers can use USML to define criteria of filtering search results.

- UDDI Search Markup Language (USML)

USML is an XML-based language proposed to uniform the search query format and dramatically reduce requesting times in a search.

- Advanced UDDI Search Engine (AUSE)

It parses USML-based search requests and triggers search commands for different UDDI registries and other information sources and then aggregates results for the search service requestors. Additionally, the presented search engine uses an instant notification broker to communicate with the service requestors and UDDI service providers. That is, when a search requester sends out a USML-based query to the advanced UDDI search engine, the AUSE will send an acknowledgment to the requestor instantly. After the Information Aggregation broker finish the aggregation of search results from different UDDI registries, it will send out a Results Available Notice (RAN) to the instant notification broker (INB). Then the INB will send search requestor a notice so that the receiver in the application can fetch the results from AUSE as soon as possible.

Note that the UDDI Source Dispatching Broker in Figure 1 is an intelligent broker. It might dynamically dispatch the constructed UDDI search commands to the pre-selected UDDI registries based on the USML request. If there is no source information defined in the USML, it might automatically dispatch the UDDI search commands to a best-known UDDI registry based on its knowledge base and intelligence.

Moreover, UDDI search is a time-consuming process. In advanced UDDI Search Engine, a Local Category Database is used to store and re-organize the UDDI category based on its knowledge and self-updating mechanism. The published category data from different UDDI registries can be used by the Local UDDI Category Database, which will be created above the UDDI technical layer. If a local category source is specified in the USML request, then the Source Dispatching Broker will route the search commands to the local UDDI category database. Of course, one USML request might include multiple search commands defined for multiple sources including Local UDDI Category Database, public UDDI registry and other private UDDI registries. At the same time, the Local UDDI Category Database will be updated in real-time when a search command is executed. Also it can be updated during a programmed time period by its own updating mechanism, which automatically sends search commands to the available UDDI registries and organizes the returned results in a well-formatted way.

It is noted that network bandwidth is an extremely valuable resource for the networked solution providers and requestors as well as e-Marketplace. Therefore, from the system point of view, the proposed advanced UDDI search mechanism will dramatically reduce the network traffic resulted from search service requestor by using only one USML-based search request and one XML-based response. In addition, it simplifies the developer's effort by avoiding mastering the UDDI search programming skills for different UDDI registries. Additionally, a quick result can be returned from the advanced UDDI search engine if the local UDDI category database is used.

We argue that Advanced UDDI Search Engine **(AUSE)** can greatly increase the efficiency of e-business application development. The goal of the advanced UDDI Search engine is to support the business-level search facilities for activities such as finding partners with products in a certain price range or availability, or finding high quality partners with good reputations in a quick way. The data in UDDI is not sufficient to accommodate this because of the cross category issues associated with high volumes and voluntary classification.

## 4. USML Construction

USML is an aggregation of different search queries that searches the UDDI registries for multiple criteria. A search could be made for Businesses, Service and Service Types matching the different criteria specified in USML by a user. Service Type is called tModel in UDDI. A tModel specifies information such as the tModel name, the name of the organization that published the tModel, a list of categories that describe the service type, and pointers to technical specifications for the service type such as interface

definitions, message formats, message protocols, and security protocols. tModel is essentially a technical "fingerprint" unique to a particular specification.

Document Type Definition (DTD) is a structural description of an XML document. It defines the elements an XML document can have, their attributes, their values and so on. A valid XML document must conform to the specified DTD. As shown in the following table, *UDDISearch.dtd* associated with USML specifies the search criteria.

```
<!ELEMENT UDDISearch (Query+,AggOperator, RequestTypeName?)>
<!ELEMENT Query (Source,SourceURL?,BusinessName?,Identifier?,
Category?,ServiceName?,ServiceTypeName?,DiscoveryURL?, FindBy)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT SourceURL (#PCDATA)>
<!ELEMENT BusinessName (#PCDATA)>
<!ELEMENT Identifier (#PCDATA)>
<!ATTLIST Identifier type (D-U-N-S|ThomasRegister) #REQUIRED>
<!ELEMENT Category (#PCDATA)>
<!ATTLIST Category type (NAICS|UNSPSC|GEO|UDDITYPE|SIC)
#REQUIRED>
<!ELEMENT ServiceName (#PCDATA)>
<!ELEMENT ServiceTypeName (#PCDATA)>
<!ELEMENT DiscoveryURL (#PCDATA)>
<!ELEMENT FindBy (#PCDATA)>
<!ELEMENT AggOperator (#PCDATA)>
<!ELEMENT RequestTypeName (#PCDATA)>
```

The following is the description of each XML element.

- "Query" specifies the query conditions. It combines keyword search, search based on identifiers, and search based on categories.

- "Source" is the name of UDDI source for the query. It can be "public UDDI" or "private UDDI" or other names.

- "SourceURL" is the URL of the source so that BE4WS can access the registry and get related information.

- "BusinessName" is the name of the business entity.

- "Identifier" is the identifier name and its associated value. Two types of identifiers are supported: D-U-N-S, and ThomasRegister.

- "Category" is the category name and its associated value. Five types of category are supported: NAICS, UNSPSC, GEO, UDDITYPE, and SIC.

- "ServiceName" is the name of the service. It is used when the search is by service name.

- "ServiceTypeName" is the name of the service type (i.e., tModel). It is used when the search is by the service type.

- "DiscoveryURL" is the URL for discovery.

- "FindBy" specifies the data type of the result value. There are three data types in UDDI: Business, Service, and ServiceType (tModel).

The following three basic types of searches for one query are defined: search by name (BusinessName, ServiceName, or ServiceTypeName. The name search is partial match (meaning that the name beginning with the specified value is matched or including the specified value), search by identifier, and search by category. It is possible to combine these basic types. The relationship among these basic types is "AND" if more than one type are specified in one query.

"AggOperator" specifies the logic relationship among queries. "AggOperator" has two values in the current design and implementation: "OR" or "AND". If "OR" is specified, all information specified in "FindBy" of each Query is returned. Unlike "AND", "OR" allows as many queries as possible. If "AND" is specified, only information related to the data type specified in RequestTypeName is returned. No more than 3 queries (one query for one data type) are allowed.

"RequestTypeName": Specify the data type name to be returned.

## 4.1 Search for Business

We can search for Businesses using any combination of Keyword, Identifier, Locator, Service Type, and Discovery URL. Only businesses that match ALL of the criteria specified are returned. At least one of the search criteria should be mentioned.

### 4.1.1 Searching by BusinessName

Specify the name of the business you are looking for in the "BusinessName" tag. The businesses with names that *start with* the characters you entered will be returned. In the USML sample above, search is for the Businesses that start with UPS and hence UPS is written in the BusinessName tag.

### 4.1.2 Searching by Identifier

A UDDI Registry allows entities to be annotated with information that uniquely identifies them. Formal identifiers such as Dun & Bradstreet numbers and Thomas Register numbers are fully supported. To search using an

identifier, specify the type of identifier in the attribute of the "Identifier" tag and write a value for the identifier.

### 4.1.3 Searching by Category

A UDDI Registry allows entities to be classified using categorization taxonomies such as North American Industry Classification System (NAICS), Universal Standard Products and Services Classification (UNSPSC), and Geographic (GEO). These classification taxonomies are generically known as 'Locators'. To search using a locator, first specify the type of locator in the attribute of the "Category" tag and then write a value for the category.

### 4.1.4 Searching by Discovery URL
A Discovery URL represents the address of URL-addressable discovery documents that contain information about a business registered in the UDDI Registry. To search using a discovery URL, specify a value into the "DiscoveryURL" tag. Businesses with discovery URLs that *start with* the characters you entered will be returned.

## 4.2 Search for Service

We can search for Service using combination of ServiceName and Category. Only services that match ALL of the criteria specified are returned. At least one of the search criteria should be given. Since business services depend on business entities, it is virtually impossible to search business services without specifying business names. If searching for business services without business names is necessary, there is a need to retrieve ALL business entities registered with a public UDDI or private UDDI, a task taking too much time. Based on the above consideration, users must also specify the business names for business service search.

### 4.2.1 Searching by ServiceName

Specify the name of the service you are looking for in the "ServiceName" tag. The business services with names that *start with* the characters you entered will be returned.

### 4.2.2 Searching by Category

In order to search the services using a category, first specify the type of category in the attribute of the "Category" tag and write a value for the "Category" element.

## 4.3 Search for Service Type
Search for Service Types using any combination of ServiceTypeName and Category is provided. Only Service Types that match ALL of the specified criteria are returned. At least one of the search criteria should be given.

### 4.3.1 Searching by ServiceTypeName

Specify the name of the service type you are looking for in the "ServiceTypeName" tag. The service types with

names that *start with* the characters you entered will be returned.

### 4.3.2 Searching by Category
In order to search using a category, first specify the type of category in the attribute of the "Category" tag and write a value for the category. For example, if you want to search for service type that start with S and having category as "NAICS", you must put "NAICS" in the "type" attribute of the Category tag in the USML, and the value "S" for the tag.

## 5. Aggregation Operators

The "AggOperator" defined in USML can take different values such as AND, OR or a function which involves a script to perform a task. Currently, only "AND" and "OR" are supported. The Script operator will be supported in the future. The results from different UDDI registries maybe be required to be aggregated depending on these operators. If the response contains redundant information, it can be filtered by the use of such operators.

Every Business is associated with a business key and every Service has a service key. Thus these operators help to combine the results of different keys and eliminate the repetitive information with the same key.

### 5.1 OR Search Criteria

If we have to search in the UDDI Registry for any business say starting with "IBM" and any service starting with "Web", we have to make 2 separate requests in a regular way: one for the business and one for the service. Similarly, request for a service type or another service or a business would require different calls, thus increasing the searching time and effort.

With the help of USML we can combine our search criteria into one request and thus get efficiency in our system by just making one call for all the desired criteria.

### 5.2. AND Search Criteria

If we search for the service types starting with "Web" and these service types must be used by businesses whose names starting with "White", we are able to specify two queries: one for service type and one for business. We use "AND" as the AggOperator tag and require the Service Type to be returned. Thus the "AND" operator is an indicator to aggregate the results obtained from the user's multiple criteria requests.

According to the UDDI specification, there are three core data types that can be queried against: business, service, and service type (tModel). If the aggregation operator is "AND",

the user is required to fill in the value for "RequestTypeName" that specifies one of the three core data types to be returned. For example, if "RequestTypeName" is business, and three queries are specified in an XML document: one for business, one for service, and one for service type, only business information that meets all the requirements specified in these three queries is returned.

There are nine possible combinations of "AND" query. If we use "Type" to refer to "Service Type", and the first part of the following names as "RequstTypeName", we should have:

- BusinessServiceType: AND of all three data types, returns Business
- ServiceBusinessType: AND of all three data types, returns Service
- TypeBusinessService: AND of all three data types, returns Type
- BusinessService: AND of Business and Service, returns Business
- ServiceBusiness: AND of Service and Business, returns Service
- BusinessType: AND of Business and Type, returns Business
- TypeBusiness: AND of Type and Business, returns Type
- ServiceType: AND of Service and Type, returns Service
- TypeService: AND of Type and Service, returns Type

The semantics of "AND" is easy to understand. For each "AND" query, the intersection of keys got from subqueries must not be empty. For example, if we use the combination "TypeBusiness", the returned service type must be used by at least one business specified in the query for "Business".

Thus we can search by Businesses, Services and Service Types. We specify the source UDDI and the associated URL to be searched into. In case the URL is not specified, default URL associated with the Source name is taken from the configuration file where the Source UDDI names are mapped with their URLs. Configuration file helps in storing large number of URLs associated with various UDDI Registries. New registries can be easily added in this file at later stages without the need to modify rest of the code using this file.

# 6. BE4WS Application Programming Interfaces (APIs)

BE4WS enables interaction with XML representations of UDDI Exploring mechanism instead of direct work with UDDI for Java and other UDDI clients, which is the usual programming model. With BE4WS, the same programming model can be used regardless of how the UDDI for Java or other UDDI clients are implemented.

BE4WS consists of two types of APIs: a regular Java API and a Web Services Interface. With these APIs, the desired business entities, services, and service types (tModels) can easily be found from different UDDI registries using one single Java call or SOAP call. The only input parameter for these calls is the USML request string. The return result is also an XML-based USML response.

## 6.1 Regular Java API

Three public functions are provided in a Java class:

➢ *public String searchByString(String inputUSMLString)*

This function takes the string containing the desired search criteria as input and returns a document of USMLResponse. The parameter *inputUSMLString* is USML request template.

➢ *public String searchByURL(String inputUSML_URL)*

This function takes the string containing the desired USML URL as input and returns a document of USMLResponse. The parameter *inputUSML_URL* is USML URL string.

➢ *public String searchByFileName(String inputUSMLFileName)*

This function takes the file containing the desired search criteria as input and returns a document of USMLResponse. The parameter *inputUSMLFileName* is Name for the file containing USML request.

Steps in usage of these functions:

- Create a USML object with all the search criteria.
- Write a java application to include BE4WS package

Any e-business application developer can use this API to easily find the business entity and services from different UDDI registries using one single request.

## 6.2 Web Services API

### 6.2.1 BE4WS SOAP Service

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. Web services could be weather reports or stock quotes. Examples of transaction Web services, supporting business-to-business (B2B) or business-to-client (B2C) operations, are airline reservations or purchase orders.

Web services reflect a new "service-oriented" approach to programming, based on the idea of building applications by discovering and implementing network-available services, or by invoking available applications to accomplish some task. This "service-oriented" approach is independent of specific programming languages or operating systems. Instead, Web services rely on pre-existing transport technologies (such as HTTP) and standard data encoding techniques (such as XML) for their implementation.

**BE4WS** can be published in the private or public UDDI Registry. Then any application can find this useful UDDI search service and invoke it through the WSDL Interfaces. You can easily use WebSphere Application Server (WAS) console to install BE4WS SOAP service as an enterprise application. The installed **BE4WS** can work as an advanced UDDI search portal that allows any soap clients to easily and effectively find business information and services information from multiple UDDI registries.

Figure 2 shows an installed **BE4WS** on WebSphere Application Server 4.0. It is an enterprise application. You can stop or start it using the WAS admin control. Further, you can assign the accessed roles and set a certain security level for **BE4WS** if required.
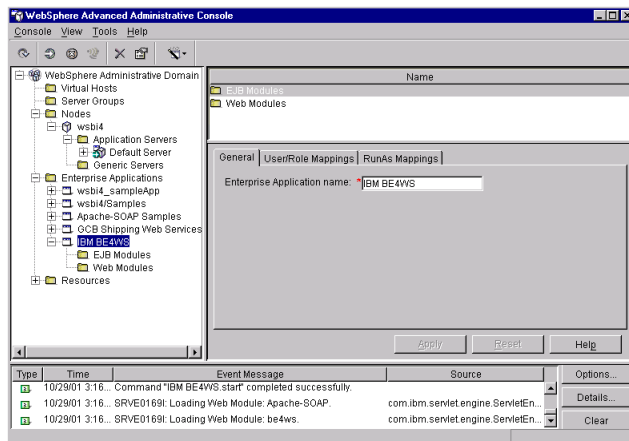
**Figure 2. BE4WS installed on WAS 4.0**

After you install **BE4WS** on WebSphere Application Server, you can use the SOAP administration tool to list **BE4WS** service, start it and stop it. The deployed service information of **BE4WS** is shown in Figure 3.
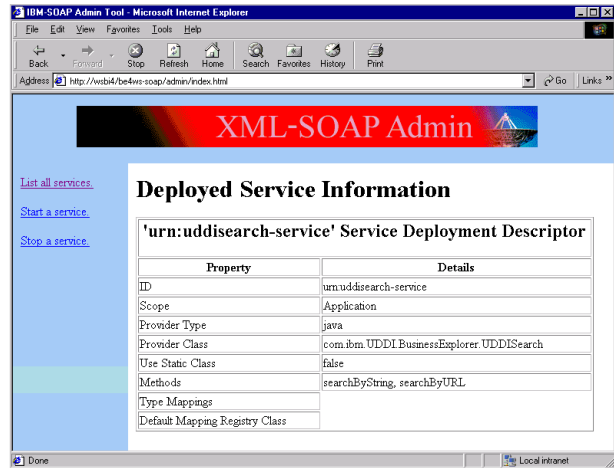
**Figure 3. SOAP Service Administration Console for BE4WS**

From the above figure, we can find there are two methods in **BE4WS** SOAP service, i.e. searchByString, searchByURL. The SOAP service ID is urn:uddisearch-service. The provider class is com.ibm.UDDI.BusinessExplorer.UDDISearch.

### 6.2.2 BE4WS SOAP Service Invocation

SOAP answers the question about how XML data can be exchanged using a standard mechanism that can be supported by middleware or application systems on any platform, in any programming language, using any network protocol, and based on any object model.

SOAP's loose-coupling model solves e-business integration problems in a number of important ways. For example, the use of SOAP simplifies the development for both the requestor and provider. By focusing on the message content only, it is possible to work with any kind of service requestor or provider regardless of implementation, without knowing any details of the implementation. You can use regular SOAP RPC program to invoke BE4WS SOAP Service.

The other way you can use is to take advantage of Web Services Invocation Framework (WSIF). It provides a standard API to invoke services; no matter how/where the service is provided as long it is described in WSDL. The architecture allows new bindings to be added at runtime. WSIF enables the user to move away from the usual Web Services programming model of working directly with the SOAP APIs, and towards a model where the user interacts with representations of the services. This allows the user to work with the same programming model regardless of how the service is implemented and accessed. You can find more information about WSIF from IBM alphaWorks [6]. The architecture of WSIF also allows stub-less invocation of Web

Services. That is, no stub is generated, and the services can be dynamically invoked.

## 7. Sample Application of BE4WS for B2B Integration

In this section, a working system is presented for demonstrating the feasibility of the proposed advanced UDDI search engine (BE4WS) for dynamic e-business integration. In this system, buyer can use Web Services to create purchase order and get the details of the created purchase orders. Meanwhile, suppliers can check the purchase order information, find transportation providers, obtain quotes for service, and assign the provider to a specific purchase order. Especially, an intelligent shipping agent using BE4WS is used to find the potential transportation service providers who serve the country of dispatch of the purchase order from a private UDDI registry.

A sample B2B integration architecture is shown in Figure 4. The PO (Purchase Order) Web Services will be invoked to create PO and get PO details. After a buyer create a PO, it will be stored in the database and processed later.
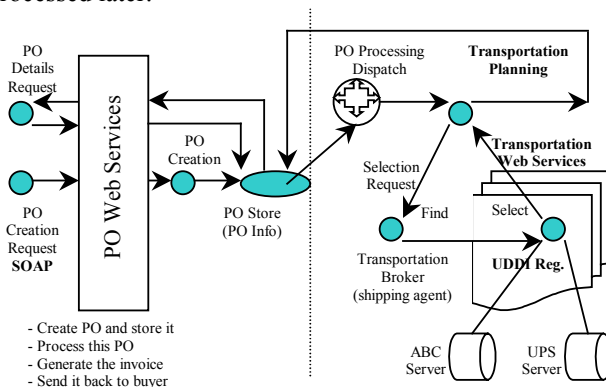


**Figure 4. B2B integration architecture**

Two purchase orders are listed in figure 5a and 5b. The dispatch country name is different. It will be used by the shipping agent to locate shipping service providers.

KEPEX PO Details -> 50-260.1008105319203

| Supplier Article Description | Nightstand |
|---|---|
| Kepex Article Num | 4 |
| Supplier Id | 999 |
| Customer Name | Metro DE |
| Qty | 80.0 |
| Dispatch Country | UK |
| Port of Loading | London |
| E T D | 2002-01-10 |
| Transport Service Provider | |

Get List of Transporter (E-Hub Memebers Only)

**Figure 5a Purchase order with Dispatch country: UK**

KEPEX PO Details -> 50-260.1010162303218

| Supplier Article Description | Hardwooden Chair |
|---|---|
| Kepex Article Num | 4 |
| Supplier Id | 999 |
| Customer Name | Sears |
| Qty | 78.0 |
| Dispatch Country | USA |
| Port of Loading | New York |
| E T D | 2002-02-10 |
| Transport Service Provider | |

Get List of Transporter (E-Hub Memebers Only)

**Figure 5b Purchase order with Dispatch country: USA**

Transportation planning is a typical PO processing process. It includes a few flow definitions (control flows and data flows) and the business rules that drive the process choreography. The supplier will select a shipping service provider for this specific PO using a shipping agent, which is designed to search UDDI registry based on BE4WS and further filter the search result using dispatch country name.

Request from the Web browser will be sent to the Trust & Access Manager first to finish the single-sign-on (SSO) process. Then the subsequent pages will automatically attach the user credential information to Transportation Planning Process that will invoke a command searching UDDI server using BE4WS or invoking a Web Service based on the business context.

In Figure 5a and 5b, The PO information is derived from the purchase order database using PO Web Services. The Dispatch Country in Figure 5a is UK and the port of loading is London. The Dispatch Country in Figure 5b is USA and the port of loading is New York. Based on this information, the transportation broker (intelligent shipping agent) will search the UDDI registry to get a transporter list of service providers in UK and USA.

In the intelligent shipping agent, the following USML is used to look up a private UDDI registry. For example, the shipping agent wants to get all the service providers which names include "trans" or "UPS". Those two search criteria are created by the shipping agent based on his/her knowledge.

```
<?xml version="1.0"?>
<!DOCTYPE Search SYSTEM "UDDISearch.dtd">
<Search>
          <ProcessId>9999</ProcessId>
          <Query>
                  <Source>Private UDDI</Source>
                  <SourceURL>http://9.2.168.233:80/
                   services/uddi/inquiryAPI</SourceURL>
                  <BusinessName >%trans</BusinessName >
```

8

```
                          <FindBy>Business</FindBy>
              </Query>
              <Query>

                          <Source>Private UDDI</Source>
                          <SourceURL>http://9.2.168.233:80/
                             services/uddi/inquiryAPI</SourceURL>
                          <BusinessName >UPS</BusinessName >
                          <FindBy>Business</FindBy>
              </Query>
              <AggOperator>OR</AggOperator>
</Search>
```

Then this USML is used to direct BE4WS to search UDDI registry. Actually, this will result in a set of meaningful service providers who provide shipping service. Then the shipping agent process the result from BE4WS based on the dispatch country name. For the PO shown in the figure 5a, the resulting search result from intelligent shipping agent is listed in Figure 6a.

From this example, the shipping agent doesn't need to write any code in order to search UDDI registry. The only thing the agent needs to do is to create a script-based USML.

For each service provider, they may provide multiple services. For each service, the supplier can invoke the Web Services to get the quotes for this service, for example. For example, the quotes of services in Figure 5b are shown in Figure 7. There are 7 services provided by UPS.

Supplier Portal -> Transporter Quotes List for PO

PO ID: 50-260.1010162303218

| Transporter | Service Name | Service Invoked | Quote (USD) | Select |
|---|---|---|---|---|
| ABC Transportation Inc. | ABC_Transport_Service | getTransportationQuote | 245.38 | ○ |
| Sunshine Transportation Inc. | Sunshine_Transport_Service | getTransportationQuote | 219.87 | ○ |
| UPS | Next Day Air, FROM Hawthorne, NY 10532 TO San Francisco, CA 94102 (Your Packaging, 120 pounds) | nextDayAir | 308.04 | ○ |
| UPS | Second Day Air | secondDayAir | 206.03 | ◉ |
| UPS | Third Day Select | thirdDaySelect | 144.27 | ○ |
| UPS | UPS Ground | Ground | 58.17 | ○ |
| UPS | Next Day Air Saver | nextDayAirSaver | 275.3 | ○ |
| UPS | Next Day Air Early AM | nextDayAirEarlyAM | 335.75 | ○ |
| UPS | Second Day Air AM | secondDayAirAM | 231.12 | ○ |

Accept

**Figure 7. Quotes for services**

After the supplier reviews the quotes, he/she can select one as a transportation service provider for this specific purchase order. Then the purchase order information will be updated. The updated purchase order information is shown in Figure 8. At the same time, the notification will be sent to the buyer, supplier and shipping company.

KEPEX Supplier Portal PO: Selected Transporter Review

PO ID: 50-260.1010162303218

| Kepex Article Num | Hardwooden Chair |
|---|---|
| Supplier Article No | Hardwooden Chair |
| Supplier Article Num | 4 |
| Supplier Id | 999 |
| Cust num | 597 |
| Qty | 78.0 |
| Dispatch Country | USA |
| Port of Loading | New York |
| E T D | 2002-02-10 |
| Transport Service Provider | UPS |

The PO is finalized. The notification will be sent to the buyer, shipping company and supplier.

**Figure 8 Updated PO with assigned service provider**

From this working B2B solution development, we have learned that Web Services are emerging e-business applications that can connect and interact with one another on the Web more easily and efficiently, eliminating much of the time-consuming custom coding currently required in UDDI search scenarios. At the same time, using advanced UDDI search mechanism enables development of powerful business services supporting dynamic, collaborative B2B activities powered by UDDI registry; facilitates dynamic business process brokering and intelligent agents with open, real-time business services.

# 8. Conclusions

With the emergence of interoperable, ubiquitous Web Services published in the UDDI, businesses have a means to describe their services in a global environment and potential trading partners have a way to discover and interact with each other. It is a critical problem to efficiently find proper Web Services in less time and effort. UDDI Search Markup Language (USML) aids in an efficient search by considering multiple criteria for the desired search from a single or multiple registries. USML is an XML-based language proposed to uniform the search query format and dramatically reduce requesting times in a search. An USML-based search request incorporates multiple search queries, UDDI sources and aggregation operators. Thus, it takes several criteria in account such as keywords to search for, identifiers, categories and so on for the desired search from a single or multiple registries.

Based on the defined USML, a framework of Advanced UDDI Search Engine (AUSE) is proposed to conduct the searching process. The basic idea of AUSE is to aggregate search results from different UDDI registries based on the proposed USML and its supporting intelligent search facilities such as Instant Notification Broker, UDDI

9

Source Dispatching Broker and Information Aggregation Broker, which have both priori knowledge of the meanings of specific categories and the ability to cross-reference across multiple categories. The aggregation broker will parse and reorganize the returned results from different UDDI Registries based on aggregation operators and rule-based script. The final result is represented as a XML response to the search requestor.

The proposed advanced UDDI search engine consists of two types of APIs for the business application developers: regular Java API and Web Services Interface.

Finally, we would like to mention here the advanced UDDI Search Engine has an extensible architecture. It can incorporate not only different UDDI technical layers (e.g. UDDI4J or similar UDDI client package), but also different sources (e.g. UDDI registries, enhanced UDDI registry or other sources such as LDAP, Web and so on).

# 9. References

[1] Web Services and UDDI, IBM Corporation. http://www.ibm.com/services/uddi.

[2] UDDI 2.0, http://www.uddi.org/, UDDI.org, 2001.

[3] Liang-Jie Zhang, Haifei Li, Henry Chang, Business Explorer for Web Services, downloadable software package on IBM alphaWorks, http://www.alphaworks.ibm.com/tech/be4ws, Dec, 2001.

[4] Microsoft Advanced UDDI Search, http://uddi.microsoft.com/search.aspx, 2001.

[5] UDDI for Java (UDDI4J), uddi4j.org, http://www.uddi4j.org/.

[6] Web Service Invocation Framework (WSIF), http://www.alphaWorks.ibm.com, 2001

Supplier Portal-> Transporters Available List for PO

PO ID:50-260.1008105319203

| Transporter: | Description: | Country Served: | Key: | Operator: | Select: |
|---|---|---|---|---|---|
| JoeSmith Transportation Inc. | UK. Since its inception in 1977 JoeSmith has been an integral part in the dramatic growth of Mediterranean Shipping Company to its position as 4th largest container shipping company in the world. This has been achieved by total dedication to customers | UK | 8375E2F0-E84B-11D5-B61C-970107B90C83 | wsbi10/services/uddi | ☑ |
| JuneFlower Transportation Inc. | UK. Established in Liverpool in 1989 JuneFlower have developed their business to meet the requirements of a broad cross section of British and Foreign manufacturing industries. JuneFlower have gained a reputation fo Quality, Reliablity and Flexibility. | UK | C9510DD0-E8D8-11D5-B61C-970107B90C83 | wsbi10/services/uddi | ☑ |
| Lightning Transportation Inc. | UK. Lightning Inc is an independent shipbroking company established in the UK on 1st January 1934. Contact: 220 St Dunstans Hill | UK | AFB44350-E84D-11D5-B61C-970107B90C83 | wsbi10/services/uddi | ☑ |

Get Transporter Quotes

**Figure 7a Service provider list for the PO (Dispatch Country Name is UK)**

Supplier Portal-> Transporters Available List for PO

PO ID:50-260.1010162303218

| Transporter: | Description: | Country Served: | Key: | Operator: | Select: |
|---|---|---|---|---|---|
| ABC Transportation Inc. | USA. Corporation provides integrated transportation, information, and logistics solutions through a powerful family of companies that operate independently yet compete collectively.U.S. Customer Service: 1-800-Go-ABCTS. | USA | C01923A0-E848-11D5-B61C-970107B90C83 | wsbi10/services/uddi | ☑ |
| Sunshine Transportation Inc. | USA. Sunshine, the world's largest package distribution company, transports more than 3 billion parcels and documents annually. Corporate Headquarters: 155 Greenwitch Parkway, NE, Atlanta, GA 30328 | USA | 2BF10F60-E84A-11D5-B61C-970107B90C83 | wsbi10/services/uddi | ☑ |
| UPS | USA. UPS, or United Parcel Service Inc., is a 93-year-old company whose history spans the bicycle to the Internet. Today the largest express carrier and largest package carrier in the world, UPS continues to develop the frontiers of logistics. | USA | 339CFD70-E84D-11D5-B61C-970107B90C83 | wsbi10/services/uddi | ☑ |

Get Transporter Quotes

**Figure 7b Service provider list for the PO (Dispatch Country Name is USA)**

10