UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

Summary

UDDI is a platform-agnostic technology for publishing, discovering and using Web services. It provides a registry of Web services data and metadata, APIs, and interfaces for accessing the services.

Note

Universal Description, Discovery and Integration (UDDI) is a registry, not a repository. The UDDI registry only stores the interfaces to Web services, but not the Web services themselves; the implementations are available at another location. A repository, on the other hand, would store both the interfaces and the implementations at the same location.

Table of Contents

  Technology Basics
  Introduction
  The UDDI Registry and Documents
  The UDDI APIs
  Using the UDDI Registry
  The UDDI Technology Roadmap
  Technology Analysis
  What the UDDI Specifications Actually Specify
  Web Services and UDDI
  UDDI and ebXML: Competitors or Complements?
  Next-Generation Web Services
  Business Use
  Benefits and Risks
  Standards
  Technology Leaders
  Technology Alternatives
  Insight

List Of Tables

  Table 1: WSDL and UDDI Element Correspondence
  Table 2: UDDI Specifications and Documents
List Of Figures

Figure 1: The Web Services Stack
Figure 2: UDDI Registry Data Structures
Figure 3: tModel File
Figure 4: Publisher Assertion
Figure 5: Mappings Between UDDI and WSDL
Technology Basics

Introduction

UDDI is a technology for publishing, querying, finding, and invoking Web services using a registry that provides data, metadata, and pointers to these services. The technology builds on established standards, including Extensible Markup Language (XML) schemas and files, Simple Object Access Protocol (SOAP) and HTTP messaging, Domain Name System (DNS) lookup, and Web Services Description Language (WSDL) Web service interface descriptions.

UDDI consists of four parts —

- The UDDI Registry
- The data, metadata, bindings and documents included in the registry
- The UDDI specifications
- Application Programming Interfaces (APIs) and services for publishing to, querying, and managing information in the Registry.

The UDDI Registry lets organizations describe themselves and their Web services, provide instructions for invoking services, or search for Web services from other organizations. The registry’s data, metadata, and bindings describe Web services and how to locate and invoke them. The specification provides detailed instructions for operating both private and public UDDI Registries. The APIs provide client-side tools for publishing, deleting, managing and querying Registry entries using SOAP and XML.

The public UDDI Registry — known as the Universal Business Registry (UBR) has been available since November 2000. The UBR is open to anyone who wants to publish or search for Web services information. IBM, Microsoft, SAP AG and NTT Communications each provide nodes for accessing or publishing to the registry. Although they are physically separate, these five nodes replicate their information so that they operate as a single logical registry.

The latest version of UDDI is version 3.0, a Published Specification submitted by the Organization for the Advancement of Structured Information Standards (OASIS) on 30 July 2002. At that time, the UDDI Community, a consortium of over 300 organizations that provided the technology’s first three versions, also became part of OASIS.

UDDI, together with SOAP and WSDL, form the Web services technology “canon.” Figure “The Web Services Stack” shows the relationship between these technologies and the overall Web services stack.

Figure 1: The Web Services Stack
The UDDI Registry and Documents

A UDDI registry is a World Wide Web-based directory supplying contact information for organizations and interfaces to their Web services. The registry is actually a meta-service populated by data and metadata about Web services and pointers to implementation documents that describe the Web services and how to use them. The registry’s rich metadata stores facilitate robust queries. XML schemas describe the information in the registry, although registry documents are not restricted to XML.

XML is the registry’s lingua franca for describing, managing, querying and invoking Web services, but the actual Web Services may be built using any format or language. In addition, the registry does not store the Web services themselves. Web services are available at physical locations — called access points — identified in the registry documentation.

Registries and Nodes

Each UDDI Registry must include at least one logical node to handle its data and metadata. All of the nodes in a Registry collectively manage a well-defined set of data, primarily by replicating information so that each node provides the same data as all of the others.

UDDI Publishers and Consumers

The UDDI Registry provides the interface between Web services publishers and Web services consumers. Web services publishers — organizations who list Web services on the registry — use it to publish and describe their services. Web services consumers — that is, clients who are looking for Web services — download documents from the registry and use the information to locate and work with the services.

UDDI Registry Listing

The Registry uses five data structures to describe and locate Web services:

- businessEntity
- businessService
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

- bindingTemplate
- tModel
- publisherAssertion

The UDDI registration document — which is completed when a Web service is listed in the Registry — includes the first three structures. tModels and publisherAssertions are separate documents. UDDI version 3 allows each structure to be signed with XML digital signatures.

Initially, UDDI’s creators likened the registry to a telephone directory — albeit a large public directory accessible through a Web browser — with different colored pages. The “white” and “yellow” pages — the businessEntity and businessService elements — provide general service descriptions and contact information.

On the other hand, the “green pages” — which include the bindingTemplate and tModel — are the most important parts of the Registry. These elements provide rich technical descriptions of Web services and detailed information for discovering, invoking and using the Web services. Since the registry was designed to help technical people work with Web Services rather than as a general-purpose directory, much of the specification work has been concentrated in these two elements (especially the tModel), and the telephone book analogy is now rarely used.

The businessEntity, businessService, bindingTemplate and tModel elements include “property bags” that furnish typed name/value elements that describe a given entity. These provide established classifications such as tax identifiers, geographic indexes, or Dun & Bradstreet business categories, or information from other known classification systems.

Figure “UDDI Registry Data Structures” shows how all five data structures populate the registry. The sections below describe these structures in detail.

Figure 2: UDDI Registry Data Structures
**Business Entity**

The `businessEntity` element describes the businesses and service providers in the Registry. As shown in Figure "UDDI Registry Data Structures," `businessEntity` is the topmost element in the UDDI XML registration. It supplies the organization’s name, address, contact information, a list of the Web services it provides, the registry site where the information is located and an identifier. This may be a `businessKey` element furnished by the publisher (the key concept is new to UDDI 3.0) or, less commonly, a Unique User ID (UUID) generated by the Registry.

Each `businessEntity` includes at least one `businessService` child element.
**Business Service**

The `businessService` element, shown in Figure “UDDI Registry Data Structures,” describes a logical group of one or more Web services identified in the `businessEntity`. This element provides nontechnical information, in business language, including names, human-readable descriptions, classifications and a unique `serviceKey` or UUID.

Every `businessService` contains at least one `bindingTemplate` child element.

**Binding Template**

A `bindingTemplate` element represents an individual Web service and is the child of one `businessService`. It supplies technical information needed by applications to bind to and interact with this Web service. The `bindingTemplate` identifies the access point — usually an Internet uniform resource locator (URL) address — where the Web service is located or an indirect reference leading to the access point — often in the form of a `tModel`. Each `bindingTemplate` has a unique `bindingKey` or UUID.

**tModel**

`tModels`, or technical models, represent concepts and constructs relating to Web services. `tModels` exist outside of the `businessEntity` element and its descendents; however, the `bindingTemplate` points to the `tModel` files for the Web service described in its parent `businessEntity`. Each `tModel` file has its own `tmodelKey` or UUID.

`tModels` contain metadata about (or pointers to) additional specifications or APIs; interface definitions from or even entire WSDL files; information for programmers; explanations of unusual datatypes, operations, and message formats, and the URLs where the actual documents are located. There is a `tModel` for each distinct specification, transport, protocol, or namespace.

Only the `tModel` file is stored in the Registry, The information and supporting documents referenced by the `tModel` are stored at other URLs identified in the file.

`tModels` are reusable. A single `tModel` can be used by any number of `bindingTemplates`. Likewise, one `bindingTemplate` can point to multiple `tModels`. This reusability makes `tModels` applicable to other purposes within UDDI such as definitions of transports and protocols, descriptions of identifier or categorization systems, and other types of information.

The `tModel` concept is critical to how UDDI represents data and metadata. UDDI version 3 defines a set of canonical `tModels` for modeling transport protocols, generating publisher assigned keys, working with inquiries and result sets, and other tasks. Users can create their own information models if needed by saving a `tModel` with the URLs of the documents it references.

Figure “tModel File” is a sample `tModel` file describing how to use the Web service in the example.

**Figure 3: tModel File**
Publisher Assertion

Publisher assertions, introduced in UDDI 2.0, declare that a formal relationship exists between two Business Entities in the UDDI registry. Commonly, publisher assertions identify relationships between a corporation and its internal units (that is, departments or subsidiaries), define private exchanges with suppliers and customers, or show membership in an industry consortium. The assertions are only valid if both entities in the relationship publish the same information, thus preventing one entity from declaring a relationship without the other’s knowledge.

The publisher assertion file must contain two main elements, including 1) each entity’s businessKey (or UUID) and 2) a keyedReference ID provided by the UDDI Registry. Also the assertion can be formatted as a tModel or a separate document. Figure “Publisher Assertion” shows a sample Publisher Assertion between Sweeney’s Meat Pies and another organization in the UDDI registry.

Figure 4: Publisher Assertion

Entity Identification and Categorization

UDDI uses published formal identifier systems and categorizations to help users find information in UDDI registries: Private UDDI registries can also use private identifiers or categories known only to the community using the registry. Registries only include pointers to identifier and classification systems; the actual classification and categorization reference information is available in tModel files. Supported identifiers and categorizations include:

- North American Industry Classification Systems (NAICS)
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

- Standard Industrial Classification (SIC)
- Dun & Bradstreet Data Universal Numbering System (D-U-N-S)
- Thomas Register
- ISO 3166 Geographic Taxonomy
- GeoWeb Geographic Classification
- Universal Standard Products and Service Codes (UNSPSC)
- U.S. Tax Codes

Public and Private UDDI Registries

UDDI versions 1 and 2 provided detailed instructions for operating public registries. Version 3 now also includes detailed procedures for nonpublic registry operations.

Public Registry

A public UDDI Registry is open to anyone who wants to publish descriptive information about their services or search for available services. The UDDI Business Registry is an example of a public registry. Although it is a single logical registry, there are interfaces — called nodes — maintained by IBM, Microsoft, NTT and SAP AG at physically distributed locations. Each node provides a standard Web interface for browsing, querying, retrieving, publishing and deleting information. All nodes are required to replicate data at frequent intervals, so that the same information should be available at each node.

Private Registries

Private UDDI registries may be operated by one or more organizations and are usually located behind a firewall or within an organization’s demilitarized zone. Private registries are becoming increasingly popular. The Business Use section contains detailed information about using private registries.

The UDDI APIs

UDDI 2.0 APIs

The UDDI specification includes two programmer’s APIs for working with Web services in the UDDI Registry. The Publisher API is for publishing information to the registry, and Inquiry API lets customers query and locate Web services in the registry. The APIs are based on XML documents and SOAP messages.

Programmers can use these commands to create applications in virtually any programming or scripting language — such as Java, Perl, Visual Basic, or C++ — for accessing the UDDI registry and its contents. Numerous toolkits are available that make access via programming language much easier.

Publisher API

This API lets organizations add, update or remove UDDI registry documentation. Any information relating to Business Entities, Business Services, Binding Templates, tModels or Publisher Assertions can be added or changed. The Publisher API consists primarily of save and delete procedures for these data structures. There are also three security-related messages for retrieving or discarding authentication tokens and retrieving all businessKeys and tModelKeys registered with a specific Business Entity.

Inquiry API
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

The Inquiry API offers find and get functions for locating and retrieving information in the registry. Users can search for lists of Business Entities or tModels, locate types of services or bindings used by a specific Business Entity; or retrieve an entire Business Entity, Business Service, Binding Template, or tModel. There are also provisions for what to do if a Web services invocation fails.

UDDI 3.0 APIs

The latest UDDI specification expands the existing publisher and inquiry APIs from UDDI 2.0 and adds many new APIs to standardize behavior and communication between UDDI implementations. UDDI 3.0 separates these into the two API Sets shown below.

Node API Sets

These consist of:

- **UDDI Inquiry and Publication APIs**, enhanced from UDDI 2.0.
- **UDDI Security Policy APIs**, for requesting or discarding authentication information.
- **UDDI Custody and Ownership Transfer APIs**, for transferring custody of businessEntity structures or tModels from one node to another, and transferring ownership of these items from one publisher to another.
- **UDDI Subscription APIs**, which let clients register (or discontinue registration) to receive information about changes in a UDDI registry or a multiregistry environment.
- **UDDI Replication APIs**, which govern how information is replicated between nodes in public and private registries.

Client APIs

These are:

- **UDDI Subscription Listener API** lets subscribers asynchronously receive information about changes to UDDI entities.
- **UDDI Value Set APIs** let third parties register value sets and control how UDDI uses these to validate information.

Using the UDDI Registry

A UDDI Registry consists of provider services, bindings and tModels. The registry includes data and metadata describing providers and services. It can also point to WSDL documents (available as tModels) that describe Web services or other types of Web service interfaces. Web services publishers — organizations who list Web services on the registry — use this information to publish and describe their services. Web services consumers — that is, clients who are looking for Web services — download and use the information to locate and work with the services. Currently the UDDI Registry:

- Lets publishers register the interfaces to their Web services on a specific registry or host node
- Replicates these interfaces to additional host nodes for the same registry, if applicable
- Lets Web services consumers find Web services interfaces for organizations within a specific industry or geographical location
- Indicates which Web service interfaces use WSDL (or some other interface), and who has implemented these services
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

- Helps consumers contact a Web services publisher about using its Web services

The UDDI Technology Roadmap

UDDI first appeared in an early beta version in mid-2000, with sponsorship from Ariba, Microsoft and IBM. The technology is now managed by the UDDI Community, a consortium of over 100 organizations that has no official “standards body” status. The consortium plans to manage the design process for the first three revisions, then license the technology to a third party — most likely a recognized standards organization.

UDDI 1.0

The first version, UDDI 1.0 became available in late 2000. It was intended to create platform- and technology-agnostic Web services using existing, widely available technologies. It proposed a basic set of services, including a free Internet public directory, formats for service descriptions and registration documents, and the API for programmatically accessing information in the registry. It supported three taxonomies and four of the five data structures (businessEntity, businessService, bindingTemplate, and tModel), and used SOAP and WSDL for messages and Web service interfaces.

UDDI 2.0

UDDI 2.0 emerged in November 2001. This version added specifications for operating a public registry node and replicating information between nodes. It also proposed a fifth data structure, publisherAssertions. In addition version 2 introduced “checked taxonomies” that display well-known business identifiers (that is, Dun & Bradstreet) with which a publisher is registered.

UDDI 3.0

OASIS published UDDI 3.0 in late July 2002. The UDDI Community initially offered Version 3.0 as a working draft in early July 2002. It then joined OASIS, which took over UDDI at the end of that month. Version 3.0 consolidates all of the specifications into a single document. Improvements include:

- Consolidating the UDDI specification into a single document
- Coverage for private and semi-private as well as public UDDI Registries
- New topologies for creating multiple interrelated registries, including root and affiliate registries that can share information and keys
- Human-readable entity keys generated from DNS names and assigned by publishers, which can be used in place of Registry-generated UUIDs
- Support for digital signatures for all UDDI entities
- Detailed guidance for establishing registry policies such as authorization models, transferring and managing data custody and ownership, key generation, information replication, external validation, and user publication limits.
- A new operationalInfo data structure containing operational data such as the date and time information was published or modified, the publisher’s identity, and the UDDI node where publication took place
- UDDI information model and categorization system extensibility
- Improved querying and discovery through complex categorization, nested queries, derived category systems, and large result set management
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

- A subscription API that proactively notifies UDDI users about any registry changes
- Improved support for WSDL
- Unicode and other internationalization features

Technology Analysis

The UDDI Registry, specification and APIs offer a technology for describing and locating Web services and the organizations that publish these services. Information is exchanged using open technologies such as XML, SOAP, HTTP, TCP/IP, and the Internet DNS. UDDI’s original purpose was to provide a technical Web service discovery for applications — not humans — but it is now obvious that human oversight must take place at some point. The technology’s long-term goal is to let organizations securely invoke remote services and dynamically integrate with other organizations over the Web.

The UDDI Registry complements — but does not replace — existing Internet searching tools, directories, marketplaces. For example, UDDI was modeled on the Internet’s DNS (Domain Name System) directory system. Where DNS maps domain names (www.mysite.com) to Internet Protocol (IP) numerical addresses (192.168.2.15), UDDI maps descriptions of Web services (UDDI Registration documents and tModels) to the actual location where a consumer can find and invoke the service.

UDDI also shares several other characteristics of DNS. Both UDDI and DNS are directories; just as DNS does not include the information at the IP address, the UDDI directory does not include the actual Web services instances, which are located somewhere else on the Internet. Both directories use commonly available technologies and are platform-neutral. Both also support hierarchically organized information (that is, a bindingTemplate points to a tModel).

UDDI can interface with an organization’s internal Lightweight Directory Access Protocol (LDAP) directories. Several vendors are working on solutions for mapping UDDI registries and LDAP directories. BEA’s WebLogic Server 7.0 includes a technology for implementing a UDDI 2.0 Registry over an LDAP server. Novell has a proposal for representing UDDI data in an LDAP directory, and Sun is developing an LDAP-compliant UDDI server.

What the UDDI Specifications Actually Specify

UDDI consists of exactly four pieces:

1. The UDDI specification(s)
2. Documents containing data and metadata about services, which populate the Registry.
3. A set of APIs for working with the Registries and their contents.
4. UDDI Registry implementations.

UDDI specifications do not handle:

- The www.mysite.com software and hardware (that is, the application and Web servers or host machine) on any public or private UDDI Registry node. This is managed by the host organization operating the node.
- Information security outside of the registry. UDDI only handles authentication and authorization of information published in the registry.
- Techniques for validating business and service listings except for the industrial and geographical indexes listed in the specifications. In the case of the UBR especially, caveat emptor is the rule — let
the buyer (in this case, the Web services consumer) beware. Within UDDI, some categorization schemes will validate whether the categorization is valid, but UDDI does not provide mechanisms to validate the business itself. Publisher assertions can be used to establish trust. UDDI 2.0 does not validate the service provider, but UDDI 3.0 includes provisions to authenticate and authorize providers.

- Contracts, procedures, rules and other artifacts for conducting business. UDDI is designed to find business, not conduct business transactions. It does describe how to find and connect to services — a first step in integrating two organizations — but it does not eliminate the need for formal contractual business relationships that must be handled outside of the registry.

- Security for messages traveling between registries, Web services publishers and Web services consumers. Again, this is up to the publisher, consumer and host organizations.

- Transport protocols. SOAP is required as a messaging protocol, but there are no mandated transport protocols. UDDI 3.0 describes using HTTP, Simple Mail Transfer Protocol (SMTP), and FTP, and can theoretically work with any transport, but transports not covered in the specification are not recommended.

- Searching LDAP directories. Some organizations are adding LDAP to search their internal directories, but LDAP is not a substitute for UDDI. The problem is translating back and forth between LDAP and XML.

- The implementations of the Web services. UDDI is not a repository, so these are located at separate URLs.

- Procedures for organizations to transact business with each other using information in the registry. UDDI does not eliminate the need for a formal contractual business relationship. The purpose of UDDI is to find, not to contract, business.

- Late binding. UDDI is not designed for runtime connections to Web services with unknown interfaces.

**Web Services and UDDI**

UDDI makes heavy use of Web services technologies such as XML, HTTP, SOAP and WSDL. In fact, the UDDI Registry is a Web service that uses itself and the other two “canonical” Web services — SOAP and WSDL — for many operations.

**UDDI and SOAP**

SOAP is one of UDDI’s fundamental building blocks; as such, SOAP support is mandatory for all versions of the technology. UDDI specifications provide information about how Web services consumers, publishers and the UDDI Registry can use SOAP messages for communications. All transactions between the Registry and its users are described as SOAP messages. Web services consumers can contact and query UDDI registries through SOAP. Web service publishers can use SOAP to publish, edit or remove information in the UDDI Registry. The registry itself can use SOAP messages to locate Web services information that fits specific criteria such as publisher, binding or type of service.

**UDDI and WSDL**

WSDL leverages XML to provide a standard vocabulary for straightforward information about Web services. This includes interface information for describing Web services such as port types, message formats, and operations — and implementation information for locating and using Web services, including URLs and protocol bindings.
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

UDDI Registries, in turn, can leverage and aggregate information from WSDL documents to populate Web services registration documents and \textit{tModels}. In general, a UDDI \textit{businessEntity} can point to information in a WSDL Service Implementation, and a UDDI \textit{tModel} can point to any or all of the information in the WSDL Service Interface. This is shown in Figure “\textit{Mappings between UDDI and WSDL}” below.

Figure 5: Mappings Between UDDI and WSDL

There are also some very specific correspondences between UDDI and WSDL elements, where information from the WSDL document can be directly loaded into UDDI, as indicated in Table “\textit{WSDL and UDDI Element Correspondence}.”

Table 1: WSDL and UDDI Element Correspondence

<table>
<thead>
<tr>
<th>UDDI Element(s)</th>
<th>WSDL Element(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{BusinessService}</td>
<td>service</td>
</tr>
</tbody>
</table>
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

<table>
<thead>
<tr>
<th>Table 1: WSDL and UDDI Element Correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDDI Element(s)</td>
</tr>
<tr>
<td>---------------------------------------</td>
</tr>
<tr>
<td>BindingTemplate</td>
</tr>
<tr>
<td>tModel</td>
</tr>
</tbody>
</table>

UDDI as a Web Service

UDDI is itself a Web service with its own WSDL file; it exposes itself as a Web service to describe the UDDI Registry. UDDI models itself in UDDI and has two tModels — one representing the inquiry API and the other representing the published API. A user can run a query to find all implementations of WSDL.

Using Web Services With the UDDI Registry

Web service consumers can use XML, SOAP, WSDL and the UDDI Registry to bind to and invoke Web services. This is one possible scenario:

- A Web service consumer finds a service using the service publisher’s businessEntity information, and requests additional information from the UDDI Registry, usually through SOAP and HTTP messages.
- The UDDI Registry sends the service’s businessKey to the consumer as a SOAP message payload.
- The consumer’s SOAP processor reads the message and extracts the payload.
- The consumer uses the businessKey and creates an XML message requesting more details. This might be a request for the entire businessEntity, information about one or more specific businessServices, or the entire set of interface documents, including bindingTemplates and tModels.
- The consumer’s SOAP service creates the SOAP envelope, puts the message in the payload and sends the SOAP message to the Registry.
- The UDDI Registry processes the consumer’s request (most commonly via HTTP) and sends the appropriate files — often formatted as WSDL documents — back to the consumer as a SOAP message payload.
- The consumer’s SOAP processor reads the message and extracts the payload.
- The consumer’s XML parser reads the interface file, processes the elements and loads the file into the Document Object Model (DOM) or Simple API for XML (SAX).
- The consumer uses the UDDI Inquiry API and instructions in the bindingTemplate and tModels to build a client-side procedure for programmatically accessing the Web service. At this point the UDDI Registry no longer handles the interaction.
- The consumer’s SOAP service creates the SOAP envelope, adds the access procedures to the payload and sends the SOAP document (again, usually via HTTP) to the URL holding the publisher’s Web service. This is a different URL than the UDDI Registry.
- The publisher’s Web services proxy parses the message, extracts the payload and calls up the appropriate processing facilities, which return their responses to the proxy.
- The publisher’s proxy collects the response, translates it to XML, creates a SOAP message and puts the response in the payload, and returns it to the consumer via SOAP and HTTP.

Binding to Web Services
There are two ways to bind to a Web service through a UDDI registry.

*Dynamic binding* takes place at runtime when a Web service is located. The searching application locates the Web service’s tModel on the UDDI Registry, dynamically translates the relevant information into code, and immediately locates the URL and invokes the Web service. Dynamic binding assumes a priori knowledge of the Web service’s signature and interface, so that an application can generate a proxy class to work with the object. This is only possible when a UDDI Registry mandates well-defined Web services interfaces. Private UDDI registries can support dynamic Web service binding.

*Static bindings* are individually created after locating — but before actually calling — a Web service. After downloading and examining a Web service’s tModel, a human developer creates the invocation code by hand based on the document’s contents. Once the code is completed, the developer then invokes the Web service at the specified Uniform Resource Identifiers (URI). Most public UDDI registries can only support static binding, since these registries provide very little control over registration document and tModel formats, and most search results are so general that they require human intervention.

**UDDI and ebXML: Competitors or Complements?**

UDDI and Electronic Business XML (ebXML) are complementary — not competitive — technologies designed to enhance business-to-business interactions. Although OASIS maintains both technologies, they derive from completely different initiatives.

Software vendors who needed standards for building and maintaining registries of documents and metadata describing their Web services created UDDI. It is intended to provide a consistent way for organizations to publish their Web services over the Internet and for consumers to query, locate, and use these Web services.

ebXML, on the other hand, is an initiative originally sponsored by two noncommercial organizations — United Nations Center for Trade Facilitation and Electronic Business and OASIS — to create an easily accessible global electronic marketplace. This marketplace uses freely available technologies to open up global transactions to small or remotely located organizations that might not otherwise participate in these transactions.

**Similarities**

There are similarities between UDDI and ebXML. Both technologies help organizations find each other and establish partnerships. Both use XML, the Internet, the World Wide Web, HTTP and other commonly available facilities to bring organizations together.

UDDI and ebXML can also work together. UDDI Web services can invoke ebXML workflows, and ebXML can reference UDDI Web services.

**Differences**

On the other hand, UDDI and ebXML are designed for completely different business interactions.

UDDI focuses on middleware and integration and discovery of services, and provides an infrastructure that lets businesses find and invoke Web services using implementation-independent technologies like XML and WSDL. The UDDI specifications focus on publishing, querying and locating information about Web services.

ebXML uses a business workflow backbone. Its intent is to foster a unified, multilingual, multicultural global electronic marketplace based on internationally compatible specifications.
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

Unlike UDDI, the ebXML specifications define both a registry and a repository and also describe how to transact business and store the artifacts from these transactions. The registry works much like UDDI, storing small, highly structured pieces of information. The repository is a physical back end that stores general information, business process models and schemas, trading protocols, information interchange scripts, messages, and other XML artifacts for creating executable trading agreements. An ebXML repository can also store Web services invoked through UDDI.

Next-Generation Web Services

There are a number of additional Web Service specifications that have been proposed to supplement SOAP, WSDL and UDDI. These include:

- Web Services Flow Language (WSFL)
- Web Services Conversation Language (WSCL)
- Web Services Hosting Technology (WSHT)
- Web Services Gateway (WSG)
- Web Services Invocation Framework (WSIF)
- Web Services Choreography Interface

Business Use

Most of the traction for UDDI, especially with large enterprises, is in private UDDI implementations. A private UDDI registry may be operated by a trading network, an industry association, a group of business partners or an individual organization. It can include only one node or multiple nodes at different locations. Private registry operators can limit who publishes information, the type and format of the information, and who is allowed to access the information. UDDI 3.0 provides detailed information about operating private UDDI registries.

One common scenario is to set up a UDDI Registry within an organization to foster code reuse. This allows all of the organization’s developers to expose their Web services and access information about other services available within the organization. When a new project begins, developers can check the registry to see if there are appropriate services that can be reused, thus minimizing the need for new, custom-built code. This provides a first step into applying UDDI for code reuse.

Another — and more exciting — scenario involves dynamic application reconfiguration through UDDI. Here, an organization creates a standard XML schema, which catalogs and categorizes Web services in the organization’s UDDI registry. Developers can then design an application that dynamically configures itself using Web services. For example, whenever a server moves or is updated, or a Web service binding location changes, the client code also has to be changed. The application can then query the registry to see what is needed and get current and backup binding information. If updates are needed developers only need to publish the information in the UDDI registry, and the clients then poll UDDI and receive the changes. The applications themselves dynamically search the registry and reuse bindings or other Web services.

Additional private registry scenarios can include:

- An e-marketplace registry operated by a standards body, industry association, geographical entity or other consortium. Web services publishers and consumers must be members of the organization — and validated during the membership process — to use the registry. The operators can charge
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

Membership fees, monitor the members and their Web services, and provide other value-added services.

- A partner catalog registry located behind the host organization’s firewall. Only trusted business partners can publish Web services to the Registry. The operators can validate any metadata — business or geographical codes and industrial identifiers — before it is added to the catalog. This scenario is useful when partners require in-place legal contracts to work together.

- A UDDI Registry deployed in conjunction with a Web portal residing on a server in the host organization’s demilitarized zone. Only internal entities can publish Web services to this registry, although both internal and external entities can search for or consume services. The portal registry helps organizations control Web services use.

- A test-bed registry, which lets developers experiment with the UDDI technology, create Web services, and check the performance and reliability. Test-bed registries may be public or private. IBM offers a public test-bed registry, and vendors are starting to package test or “non-production ready” sample registries with their Web services development tools and servers.

UDDI Registry Use Recommendations

Evaluators and potential customers should not consider using public UDDI registries (and WSDL, which has its own set of issues) for business-critical systems at present unless there are no suitable alternatives, and there is a quantifiable return on investment (ROI). For example, one organization uses “push” technology and SOAP to publish a non-critical Web service to a third-party public UDDI registry. The third party manages all transactions associated with the Web service. Consumers do not interact with the publishing organization, significantly reducing the risk.

On the other hand, organizations should investigate private UDDI implementations for pilot projects or non-critical internal applications. Private UDDI registries can open up access to information in corporate databases and allow internal organizations to advertise their services to each other. They also provide well-known interfaces to foster dynamic binding.

Benefits and Risks

Benefits

Rich Searching and Categorization

UDDI Registries provide excellent identification and categorization facilities as well as rich searching procedures. The original UDDI version 1 registries were based on existing directories with rigid classification systems; UDDI 2.0 added more identification and classification systems. UDDI 3.0 provides multiple layers of categorization and identification that can be amalgamated together and a flexible system to accommodate different types of taxonomies and metadata. In addition, UDDI now supports nested queries, so searches that used to require multiple steps can now be combined into a single query. UDDI also provides management for large-result sets.

UDDI Registry Based on DNS

UDDI’s designers modeled the registry on existing DNS registries. DNS translates Internet domain names into IP addresses. Browsers and portals use DNS to translate human-readable Internet domain names (for example, www.myhome.com) into numerical addresses (for example, 192.168.2.11). If one DNS server isn’t familiar with a specific domain name, it queries additional servers until it finds the name and the mapping. The public UDDI Registry works much like DNS and can query additional hosts until it finds an appropriate Web service interface document.
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

**Implementation Independent**

Like the other canonical Web services technologies, UDDI works with any language, application, or other software that supports HTTP requests. Most UDDI Registry entries are also self-describing, especially if they are created using WSDL. SOAP message support also means that communication with the Registry is completely independent from both the publisher’s and consumer’s application platforms.

**Supports Composed Web Services**

UDDI can work with composed Web services. A composed Web service is one that calls one or more additional Web services to complete a job or aggregates simpler Web services. If such calls take place, the Web service invocations are completely transparent to the consumer who originally invoked the calling service and may take place using workflow or implementation-specific information. The composed UDDI Web service can invoke a Web service available anywhere on the Web — another UDDI directory, an ebXML Registry or Repository, a database, or server URL.

**Based on Existing Technologies**

UDDI was designed using existing, commonly available technologies. It supports HTTP, SMTP, and Post Office Protocol 3 (POP3) over Internet; works with SOAP messaging; and hosts XML and, in many cases, WSDL documents in the registry. Transactions can take place through Web-based user interfaces, and developers can use virtually any programming language to create bindings to Web services.

**Extensible**

As mentioned above, developers can extend UDDI documents to include technologies or components that are not explicitly supported in the current specifications. The tModels provide the greatest degree of extensibility, since they allow Web service publishers to describe service-specific information — such as protocol or programming language mappings — that is not easily categorized. UDDI registration documents and XML Schemas can also be extended.

**Private UDDI Registry Benefits**

Private Registries have many advantages compared to public registries. Private registry operators have considerable control over what is being published, who is publishing the information, and who can access what types of registry information, facilitating querying, locating, and performing late binding to Web Services. In addition, private UDDI registries may not require replicating data between registry instances (some may elect to share data), may be more lightweight than public registries, and can support and enforce well-defined operations and taxonomies. UDDI 3.0 includes information for operating private UDDI Registries.

**Risks**

Of the three Web Services technologies, UDDI is the least mature, with much work needed. As such, there are quite a few risks for organizations to consider.

**Not Yet a Standard**

Although UDDI is intended to formalize procedures for creating and maintaining registries populated by XML documents, it’s currently a set of published specifications — not a standard. Since OASIS — a recognized standards body — has taken over maintaining UDDI, there is likely to be more interest in the technology and greater incentives to comply with the specifications. Although most vendors offering UDDI-related products try to comply with the existing directives, there is no enforcement mechanism. There are also problems with UDDI versions. For example, one UDDI registry may operate according to
UDDI 2.0, another may be UDDI 3.0-compliant, and yet another may offer some UDDI 3.0 features and some UDDI 2.0 features.

**Not a Full-Featured Business Directory or Discovery Service**

UDDI is simply a registry that hosts technical descriptions of Web services and their publishers. It is not intended to be a telephone directory with easy-to-read descriptions of vendors and services for ordinary people. UDDI does expose some features for humans, but UDDI is designed to be a technical, programmatic discovery service. Applications can use UDDI and find value in it. This is a tool for developers to use. There may be applications for a business analyst to use UDDI, but it is not a consumer directory like the telephone book. There are other Internet — and offline — directories for consumers. But there aren’t any public, replicated, specified, standardized models for describing and using Web services that offer comparable features for Web services discovery both inside and outside of an organization.

**Security Limited to Registry Information**

By design, UDDI security is limited to the information within a registry. The registry supports two types of interactions — publishing and searching — with security in the form of token passing or digital certificates available for publishing operations. Each publishing API call must be transmitted over secure HTTP (S-HTTP) and must also include an authentication token or digital signature as one of its parameters. Nevertheless, UDDI doesn’t handle any other type of security. Transactional security, security for Web services, or bulletproof business entity validation are outside the parameters of UDDI. Individual registry implementations, however, are free to establish their own security procedures such as role-based authentication.

**Does Not Validate Publishers and Web Services**

At this time, the public UDDI Global Business Registry — and many private UDDI registries as well — simply advertise Web services and their publishers. Even if a customer can validate a publisher’s identity, it is virtually impossible to verify that a Web service provides what it claims. In addition, publishers are designing UDDI and WSDL documents to maximize customers, not necessarily for technical accuracy. Like many software vendors, UDDI registry operators may also be tempted to “add value” by offering nonstandard technologies, making it difficult to search or share information between registries. Although UDDI 3.0 offers authentication for the publisher, bulletproof validation is outside the scope of UDDI and likely to be provided by third parties.

**Does Not Handle Business Transactions**

Once the Web service consumer has located the publisher and its Web services, UDDI steps out of the picture. The specifications do not handle the business transaction between the two parties. Again, this is by design, as UDDI only handles describing and discovering Web services.

**Business Transactions Require Human Intervention and Due Diligence**

Many people are uncomfortable with letting computers seek out and negotiate for services without human intervention, especially when the supporting technologies are incomplete and insecure. For these reasons — and especially since UDDI may never provide bulletproof validation — UDDI Registries and associated documents must include human-readable components. Humans are likely to play a significant role in finding and using Web services for the near future, and completely automated Web services transactions are not likely except for very simple, well-defined cases. You still need to know your partners and business practices.
Standards

As of 30 July 2002, OASIS manages and approves UDDI technologies, documents, and other materials, and UDDI version 3 is the latest published specification. UDDI.org handled the first three revisions, after which it transferred the specifications to OASIS. The OASIS Web site, www.oasis-open.org, explains the transition and houses all UDDI specifications and reference documentation. OASIS has set up a UDDI Specification Technical Committee (TC) which will work on the registry foundation originally created and published by UDDI.org.

Table “UDDI Specifications and Documents” lists documentation for UDDI versions 1, 2 and 3.

<table>
<thead>
<tr>
<th>Table 2: UDDI Specifications and Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Document</strong></td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td><strong>UDDI 1.0</strong></td>
</tr>
<tr>
<td>UDDI Version 1.0</td>
</tr>
<tr>
<td>Programmers API Specification</td>
</tr>
<tr>
<td>UDDI Version 1.0 XML Schema</td>
</tr>
<tr>
<td>UDDI Version 1.0 Data Structure Reference</td>
</tr>
<tr>
<td>Version 1 WSDL Service Interface Description</td>
</tr>
<tr>
<td>Version 1 Overview Documents</td>
</tr>
<tr>
<td><strong>UDDI 2.0</strong></td>
</tr>
<tr>
<td>UDDI Version 2.0</td>
</tr>
<tr>
<td>Programmers Specification</td>
</tr>
<tr>
<td>UDDI Version 2.0 XML Schema</td>
</tr>
</tbody>
</table>
Table 2: UDDI Specifications and Documents

<table>
<thead>
<tr>
<th>Document</th>
<th>Status</th>
<th>Original Publishing Organization</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UDDI 1.0</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UDDI Version 2.0 Replication</td>
<td>Published Specification</td>
<td>UDDI.org</td>
<td>Describes requirements for replicating data between two or more UDDI Registries and includes a programming interface for data replication.</td>
</tr>
<tr>
<td>Specification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UDDI Version 2.0 XML Replication</td>
<td>Published Specification</td>
<td>UDDI.org</td>
<td>XML schema identifying XML entities used for data replication between multiple UDDI Node Operators.</td>
</tr>
<tr>
<td>Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UDDI Version 2.0 XML Custody</td>
<td>Published Specification</td>
<td>UDDI.org</td>
<td>XML schema identifying data structures used in transferring custody of information in a UDDI Registry from one publisher to another.</td>
</tr>
<tr>
<td>Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UDDI Version 2.0 Operator’s</td>
<td>Published Specification</td>
<td>UDDI.org</td>
<td>Describes required tasks and functions for UDDI Node Operator organizations.</td>
</tr>
<tr>
<td>Specification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UDDI Version 2.0 WSDL Service</td>
<td>Published Specification</td>
<td>UDDI.org</td>
<td>Identifies XML elements and syntax used for inquiry and publishing operations. Consists of two XML documents: inquirev2.wsdl, publishv2.wsdl.</td>
</tr>
<tr>
<td>Interface Description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>UDDI 3.0</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 2: UDDI Specifications and Documents

<table>
<thead>
<tr>
<th>Document</th>
<th>Status</th>
<th>Original Publishing Organization</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDDI 1.0</td>
<td>Open Draft</td>
<td>OASIS</td>
<td>UDDI uses XML Schemas to formally describe data elements and structures.</td>
</tr>
<tr>
<td>UDDI Version 3.0 Schemas:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• API Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Custody Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Subscription Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Subscription Listener Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Replication Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Value Set Validation Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Value Set Caching Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Policy Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Policy Instance Parameters Schema</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 2: UDDI Specifications and Documents

<table>
<thead>
<tr>
<th>Document</th>
<th>Status</th>
<th>Original Publishing Organization</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDDI 1.0</td>
<td></td>
<td>OASIS</td>
<td>Provides definitions as a set of WSDL files.</td>
</tr>
</tbody>
</table>

- UDDI Version 3 WSDL Service Interface Descriptions:
  - API Binding
  - API Port Type
  - UDDI Custody Binding
  - UDDI Custody Port Type
  - UDDI Replication Binding
  - UDDI Replication Port Type
  - UDDI Subscription Binding
  - UDDI Subscription Listener Binding
  - UDDI Subscription Listener Port Type
  - UDDI Value Set Validation Binding
  - UDDI Value Set Validation Port Type
  - UDDI Value Set Caching Binding
  - UDDI Value Set Caching Port Type
Table 2: UDDI Specifications and Documents

<table>
<thead>
<tr>
<th>Document</th>
<th>Status</th>
<th>Original Publishing Organization</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDDI 1.0</td>
<td>Published Specification</td>
<td>OASIS</td>
<td>Addresses concerns with limitations of existing XML canonicalization algorithms for applications that use XML Schema to process and validate data.</td>
</tr>
</tbody>
</table>

Technology Leaders

**Antarcti.ca**

Tim Bray’s Antarcti.ca’s Visual Net, a set of visual mapping tools to help navigate the Internet, can be used to search a UDDI Registry. VisualNet for UDDI, which is implemented on the public UDDI Business Registry, lets Web services consumers search the registry database using two-dimensional maps and drill-down features. VisualNet searches can use Universal Standard Products and Services Classification (UNSPSC); the North American Industry Classification System (NAICS); or the Standard Industrial Classification (SIC) classification codes.

**BEA**

WebLogic Server 7.0 is the first version to include UDDI tools. The server includes a server-side interface, a UDDI Registry and a Java implementation of the UDDI client APIs. The UDDI Directory Explorer provides an interface for publishing and modifying information in a Web Server UDDI Registry and searches private and public registries. The client-side APIs provide Java implementations of the UDDI Publishing and Inquiry APIs.

BEA’s new WebLogic Workshop provides tools for developing Java and Web Services. Users can create Web Services using XML and Java, then deploy them to any UDDI 2.0-compliant registry. Workshop also allows users to download WSDL files from any UDDI Registry and then turn them into a Service control, which can invoke or call additional Web services.

**CapeClear**

Cape Clear focuses exclusively on Web services with its CapeConnect integration platform and CapeStudio development tools. Both products have strong UDDI support at all phases: CapeConnect includes a private UDDI Registry with a graphical interface as well as support for UDDI Registries built using Oracle. The Registry’s Web interface can be configured to reflect an organization’s look and feel or host industry-specific registries. CapeStudio includes a UDDI Browser for searching and publishing to a UDDI Registry or loading WSDL files into CapeStudio and a UDDI Web client for administering a private registry.

**IBM**

IBM was one of the three original developers of the UDDI Registry and has hosted a UBR node since 2000. IBM also has a beta version of a UDDI 2.0 registry and a test bed registry at its Web site.

IBM offers a number of advanced UDDI technologies and tools. Many of these originated as part of the Web Services Toolkit on the AlphaWorks site and are now included as part of the WebSphere application services platform.
IBM's UDDI4J toolkit provides the UDDI's Programmer and Inquiry APIs using Java classes. It is a complete client-side UDDI implementation of these APIs, including a Java-based UDDI proxy server for connecting to, querying and publishing to a UDDI Registry. The proxy server generates and parses XML messages sent to or received from the UDDI Registry. Hewlett-Packard and Apache also offer UDDI4J as part of their own Web Services tools.

The Web Services Toolkit includes Business Explorer for Web Services, an XML-based UDDI query engine with advanced searching capabilities. Users can create complex queries for finding businessEntities, serviceDescriptions, and tModels from multiple UDDI registries, and the Business Explorer aggregates these into a single result set.

IBM's WebSphere UDDI Registry first became available through the Web Services toolkit. This Registry, which works with WebSphere Application Server 4 and later, is a private UDDI 2.0-compliant Registry. It can be used to stage and test Web services or operate a full internal registry. It uses the Java-based UDDI4J interface to the UDDI APIs and includes a graphical interface for query search, and publication. IBM also released a Python toolkit for the registry.

Microsoft

Microsoft has been heavily involved in UDDI since the technology's inception. The vendor has hosted a node on the public UBR since 2000 and offers a UDDI 2.0 test registry on its Web site. UDDI support is available in Visual Studio 6, Visual Studio .NET, Microsoft Office Web Services Toolkit, and also in a variety of UDDI software development kits (SDKs) and server tools.

Microsoft has offered Web services' SDKs for the past two years. The latest UDDI SDK provides client-side development components, sample code, and documentation, allowing developers to add Web services registration to any type of software that needs to find and invoke remote Web services.

Windows .NET Server includes Enterprise UDDI Services, a UDDI 2.0-compliant private registry. Visual Studio .NET lets developers publish Web services information directly to UDDI, broadcast information about new or updated services from the integrated development environment (IDE) and automatically generate code for WSDL interfaces.

The Office XP Web Services Toolkit provides Web services referencing and location tools through Visual Basic for Applications (VBA). Developers can import a Web service's WSDL file directly into VBA so that Microsoft Office-based applications can locate and invoke Web services within an enterprise.

OASIS and UDDI.org

OASIS currently oversees and approves specifications for the UDDI Registry, the documents included in the Registry, and the UDDI APIs. UDDI.org, an informal business group, produced UDDI versions 1 and 2 and published a draft UDDI version 3 specification in early July 2002. By the end of July, UDDI.org transferred ownership of UDDI to OASIS, which republished UDDI 3.0 under its own name. In late August 2002 OASIS organized a UDDI Specification Technical Committee to advance the technology. Committee members include BEA Systems, Cincom, Computer Associates, E2open, Entrust, Fujitsu, IBM, Intel, IONA, Microsoft, Novell, Oracle, SAP, SeeBeyond, Sun, TIBCO, Verisign, webMethods and XML Global. OASIS also manages related XML and Web services technologies, including XML Topic Maps, ebXML, Universal Business Language (UBL), WS-Security, Web Services for Interactive Applications (WSIA) and Web Services for Remote Portals (WSRP).

SAP
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

SAP’s new SAP Web Application Server, released in spring 2002, supports all three main Web services standards, including UDDI. SAP has also been operating a node in the public UDDI Business Registry since October 2001.

Sun

Sun provides UDDI tools through the Java APIs for XML and the Java Web Services Developer Pack. The Java API for XML Registries (JAXR) provides Java tools for accessing registries based on known specifications, including UDDI and ebXML. Developers can use JAXR, along with Java API for XML Messaging (JAXM) and SOAP, to register an organization in a registry, publish or delete information, and search a registry.

Systinet

Systinet’s Web Application and Services Platform (WASP) provides an Enterprise and Standard implementation of a UDDI Registry. Both registries support UDDI 1.0 and 2.0 APIs, SOAP messaging, and Oracle and PostgreSQL relational databases. The WASP UDDI interface is based on HTML 3.2. Communications must be authorized and encrypted using Secure Sockets Layer (SSL). The Enterprise Edition adds advanced searching capabilities to the UDDI Query API for searching collections of names, categories and tModels.

Technology Alternatives

Many different technologies and services complement UDDI. Almost any type of service can be described in UDDI and listed with a pointer to other technologies for invoking the service.

Complementary technologies and services include LDAP, ebXML, XML/edi, Web Services Inspection Language (WSIL), Directory Services Markup Language (DSML), and Java API for XML Registries (JAXR). These are described below.

LDAP

LDAP is the “great-granddaddy” of Internet directory protocols, and there are many children. It emerged as a lightweight alternative to X.500’s large and cumbersome Directory Access Protocol (DAP) for clients to query directories hosted on servers. LDAP-compliant directories can exist at any level, from publicly accessible servers to private servers for organizations and smaller units, both inside and outside of a firewall. Most modern e-mail clients can query LDAP for e-mail addresses, and LDAP servers frequently host security-related information such as authentication and authorization rules. Several vendors are looking at using LDAP directories for internally storing UDDI registry data. These include BEA, Novell and Sun Microsystems.

DSML

The OASIS DSML Working Group manages this technology, which is an XML vocabulary for working with LDAP directories. The first version, DSMLv1, provided a format for representing the structure of a directory as an XML document, so that clients could read the directory using XML. The current version, DSMLv2, also supplies XML formats for directory queries, protocol bindings (including a formal SOAP request/response binding), operations on a directory and the results of these operations. Since DSMLv2 implements LDAP using XML, it can bring LDAP functionality to clients that are incompatible with LDAP (such as mobile devices) or cannot access LDAP directories through firewalls.

XML/edi
UDDI (Universal Description, Discovery, and Integration) and Web Services: Perspective

XML/edi is a strategy to promote XML for e-business by using it alongside electronic data interchange (EDI). Since EDI’s cost and complexity prevented broad acceptance throughout the business community, XML/edi proponents would like to combine several technologies — EDI, XML, directories, repositories, templates, and agents — into a system that would handle both data and processing information. In this system, XML will supplement and, in some cases, replace existing EDI transactions.

Web Services Inspection Language (WSIL)

WSIL is a joint effort between IBM and Microsoft for decentralized Web services discovery. It provides an XML document format for querying multiple Web sites to see if they have Web services available. In theory, this lets Web services publishers and consumers locate Web services before they are published in UDDI Registries or even bypass UDDI registration entirely. WSIL, which includes bindings to UDDI and WSDL; can complement UDDI as a Web services crawler that queries many different sites. Implementations are currently available through IBM DeveloperWorks and MSDN.

Insight

UDDI shows great potential as a technology- and platform-independent registry for Web Services. It leverages both WSDL and SOAP and provides a lightweight structure for publishing, querying, locating, and invoking all kinds of online (and even offline) services. The latest version, 3.0, provides very rich querying and searching facilities as well as information about establishing private and shared registries, which show great promise for developing and using UDDI. As promised, the UDDI Community transferred the technology to OASIS, which will manage and enhance UDDI now and in the future. There are still a number of issues that must be addressed before UDDI is completely ready for mission-critical, interorganizational applications.