



## Tracking the evolving specification

[Tom Bellwood](#) ([bellwood@us.ibm.com](mailto:bellwood@us.ibm.com))

 Senior Technical Staff Member, IBM  
 July 2002

The Universal Description, Discovery, and Integration (UDDI) project continues to enrich the toolset available to businesses to represent and model Web services in the UDDI business registry. This article introduces UDDI and the enabling role it serves in the growth of Web services. You can learn how UDDI works, as well as discover new and upcoming features for the UDDI specification.

## What is UDDI?

The UDDI project encourages the inter operability and adoption of Web services. It is a partnership among industry and business leaders and was founded by IBM, Ariba, and Microsoft. Now, over 300 companies participate. UDDI provides a standards-based set of specifications for service description and discovery, as well as a set of Internet-based implementations. UDDI continues to rapidly evolve and to gain industry support. The specification has developed quickly because it is backed with rapid implementation, which proves the concepts and provides a rich experience base for further refinement of the specification.

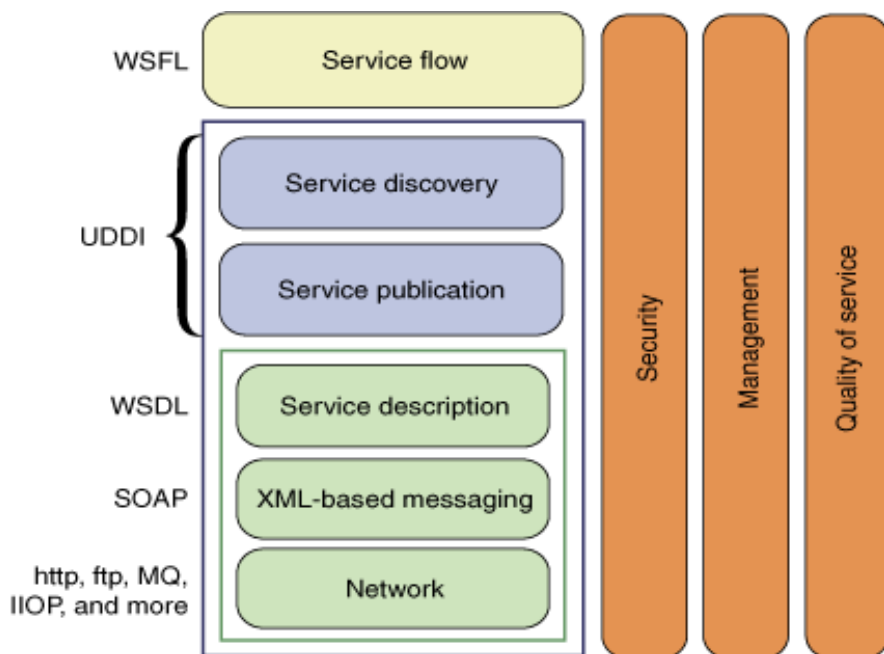
UDDI addresses a number of business problems. First, it helps broaden and simplify business-to-business (B2B) interaction. For the manufacturer who needs to create many relationships with different customers, each with its own set of standards and protocols, UDDI supports a highly flexible description of services using virtually any interface. For the flower shop in Australia that wants to get plugged in to every marketplace in the world but does not know how, UDDI provides a way to do this. The specifications allow the efficient and simple discovery of their business and the services they offer by publishing them in the registry.

For the B2B marketplace provider that needs to get catalog data for the suppliers in its industry along with connections to billing services, packers, shippers, insurers, and so on, UDDI allows dynamic discovery and integration of relevant Web services into an aggregate business process. UDDI provides one-stop shopping for information on businesses and electronic services. Publishing business and service information in UDDI makes it broadly accessible to others.

UDDI is based on existing standards, such as Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP). All compliant implementations of UDDI support the UDDI specification. The public specification is developed in an open, inclusive process by the organization's members. The intention is to produce and implement three successive versions of the specification before turning ownership for future development over to an independent standards body. The UDDI Version 1 specification was published in September 2000, and Version 2 was published in June 2001. Version 3 is under development with availability expected in mid 2002. Version 1 established a base for the registry, while Version 2 added features like business relationships. Version 3 continues to address areas important to the ongoing development of Web services, such as security, improved internationalization, registry interoperability, and various API improvements to enable further tooling improvements.

**Figure 1. The layered Web services stack with UDDI**
**Contents:**

- [What is UDDI?](#)
  - [How UDDI works](#)
  - [The UDDI specification at a glance](#)
  - [What's new in UDDI version 2?](#)
  - [Modeling support](#)
  - [Powerful categorization](#)
  - [Enhanced inquiry](#)
  - [Internationalization](#)
  - [Replication](#)
  - [What's ahead](#)
  - [Conclusion](#)
  - [Resources](#)
  - [About the author](#)
  - [Rate this article](#)
- 
- Related content:**
- [Programming with UDDI4J version 2](#)
  - [Subscribe to the developerWorks newsletter](#)



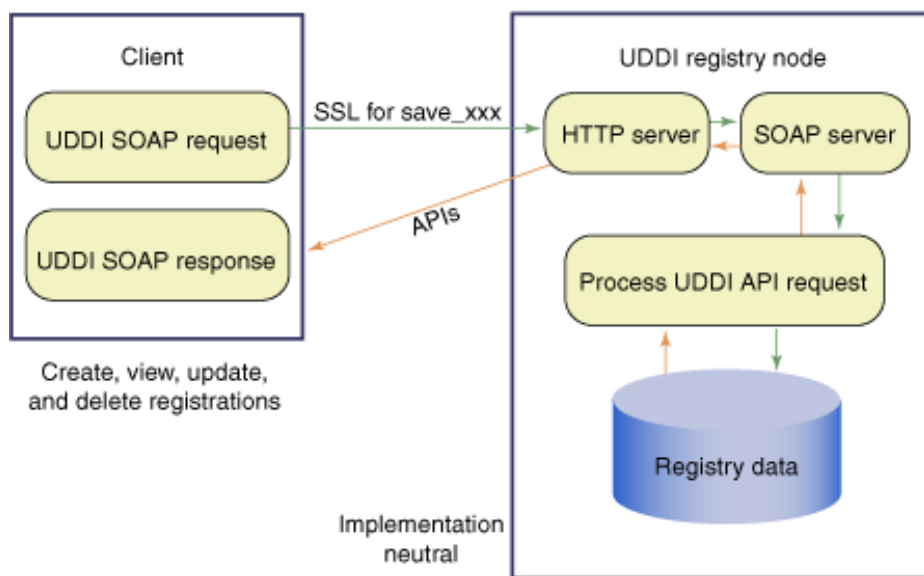
As shown in [Figure 1](#), UDDI fits into an overall Web services stack and is a key component of its foundation, enabling the creation, specification, discovery, and invocation of Web services.

UDDI builds on a network transport layer and a SOAP-based XML messaging layer. Service description languages, such as Web Services Description Language (WSDL), provide a uniform XML vocabulary (similar to an Interactive Data Language (IDL)) for use in describing Web services and their interfaces. You can build upon this overall foundation by adding layered capabilities, such as Web services workflow descriptions using Web Services Flow Language (WSFL), security, management, and quality-of-service features, which address system reliability and availability.

### How UDDI works

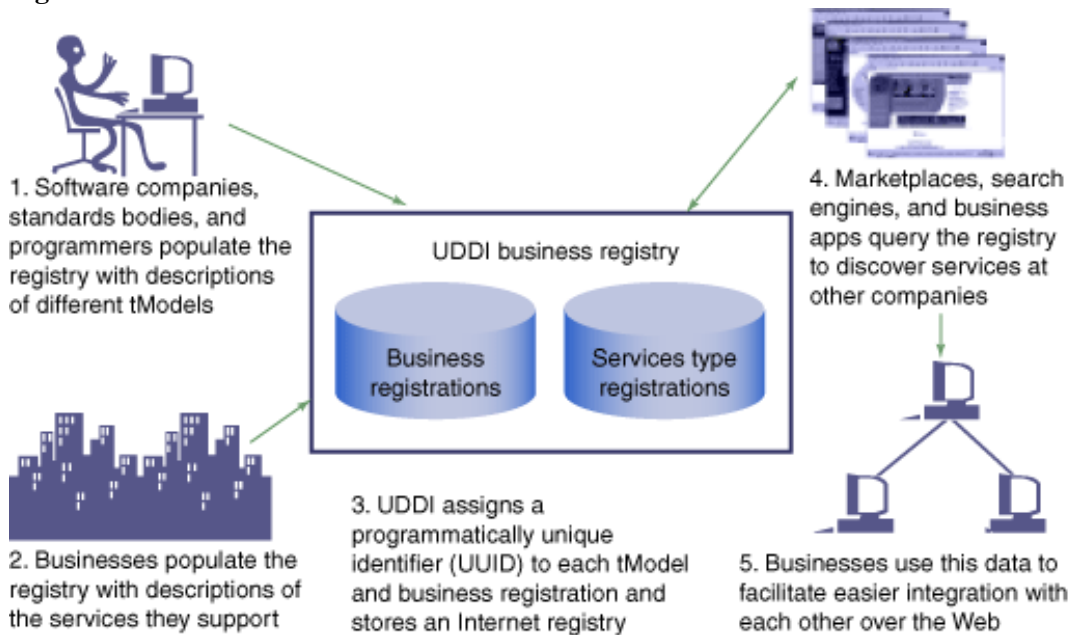
A UDDI registry contains programmatically accessible descriptions of businesses and the services they support. It also contains references to industry-specific specifications that a Web service might support, taxonomy definitions (used for meaningful categorization of businesses and services), and identification systems (used for meaningful identification of businesses). UDDI provides a programming model and schema, which define the rules for communicating with the registry. All APIs in the UDDI specification are defined in XML, wrapped in a SOAP envelope, and sent over HTTP.

**Figure 2. Flow of UDDI messages between Client and Registry**  
**UDDI and SOAP**



[Figure 2](#) illustrates the UDDI message transport, from the client's SOAP request over HTTP to a registry node and back. The registry server's SOAP server handles the UDDI SOAP message, processes it, and returns a SOAP response to the client. As a matter of registry policy, client requests that entail modifying data are required to be secured and authenticated transactions.

**Figure 3. How UDDI works**



[Figure 3](#) illustrates how a UDDI registry is populated with data and how customers discover and use this information. A UDDI registry is built on the data provided by its customers. There are several steps to making data useful in UDDI. As shown in step 1, publishing useful information to the registry begins when software companies and standards bodies define specifications relevant to an industry or business, which they register in UDDI. These are known as technical models, or more commonly as *tModels*.

In step 2, companies also register descriptions of their businesses and the services they offer. A UDDI registry keeps track of all these entities by assigning each one a programmatically unique identifier, known as a Unique Universal Identifier (UUID) key as shown in step 3. A UUID key is guaranteed to be unique and never changes within a UDDI registry. These keys look like a formatted random hexadecimal string (for example, C0B9FE13-179F-413D-8A5B-5004DB8E5BB2). They may be used to reference the entity with which they are associated. UUID keys created in a registry are only meaningful within the context of that registry.

Other clients, such as e-Marketplaces, search engines, and business applications (for example, a workflow-based aggregation of Web services) in step 4, use a UDDI registry to discover services of interest. In turn, other businesses may invoke these services, allowing simple and dynamic integration as illustrated in step 5.

Data in a UDDI registry can be conceptually divided into four categories, each of which represents a top-level entity in UDDI. Every such entity is assigned its own UUID and always can be located in the context of that UDDI registry with this identifier:

- Technical models
- Businesses
- Business services
- Service bindings

Registry information for businesses and services can be thought of in three groups: white, yellow, and green pages.

White pages represent basic information on a business, such as its name, description of the business it does, contact information, and the like. It also includes any identifiers for that business, such as a Dun & Bradstreet D-U-N-S® Number.

Yellow page information extends your ability to find a business or service within the registry by supporting classification using various taxonomy systems for categorization. Such classifications may be associated not only with businesses and their services, but with tModels. If only white, yellow, or both page data is provided, a registry entry is of limited value in the programmatic discovery and use of services. To do that, information on how and where to programmatically invoke a service is needed, and the green pages provide this information.

Green pages are the binding information associated with services and provide references to the technical specifications those services implement, as well as pointers to various file and URL-based discovery mechanisms.

A UDDI registry is composed of one or more implementations of the UDDI specification that interoperate to share registry data. One special UDDI registry is composed of a set of publicly accessible UDDI implementations called nodes. These interoperate to share registry data and, taken together, form the UDDI Business Registry. This registry is provided at no charge to the public. All UDDI Business Registry entries exist redundantly on all Operator sites, but entries may be changed only at the site where they were created.

Both IBM and Microsoft operate Version 1 nodes in the UDDI Business Registry, and they, as well as both HP and SAP, currently operate beta sites that support much of the Version 2 UDDI specification. All four plan to support Version 2 production registries in mid-2002. Each operator supports the SOAP APIs defined by the UDDI specification. Compliance is enforced through a business contract. Operators are free to offer additional services beyond those required by the specification, such as browser-based user interfaces (which all of the operators have done).

### The UDDI specification at a glance

The UDDI specification is composed of several documents. The API specification describes the SOAP APIs, which allow you to perform discovery and publishing operations. Request/response semantics and error handling are described. There is also substantial information on conventions and usage. Companion documents include the Data Structure specification and the API Schema, which define the message and data semantics.

The UDDI APIs fall into either the Inquiry or Publishing category. Version 1 supports the API operations, as shown in .

### Listing 1. Summary of UDDI V1 APIs

Inquiry Operations:	Publishing Operations:
Find	Save
find_business	save_business
find_service	save_service
find_binding	save_binding
find_tModel	save_tModel
Get details	Delete
get_businessDetail	delete_business
get_serviceDetail	delete_service
get_bindingDetail	delete_binding
get_tModelDetail	delete_tModel
get_registeredInfo	get_registeredInfo
	Security
	get_authToken
	discard_authToken

The Inquiry APIs lend themselves to three query patterns:

- The browse pattern entails the use of find operations, which allow you to browse for entries using a variety of criteria, such as taxonomy categorizations, identifiers, or partial name information using the `find_XXX` APIs.
- The drill down pattern involves obtaining detailed information about an entity that you already have found. The `get_XXX` APIs support this capability.
- The invocation pattern is the last pattern. Invoking services requires using the binding template information, which typically is cached by the client for repeated use without the need to return to the registry for the same information each time the client needs it. If the binding information changes, failure to acquire and use the service should cause clients to return to the registry to refresh this information. This is known as the invocation pattern.

Saving and deleting (with `save_XXX` and `delete_XXX` APIs) can be accomplished against each of the top-level entities and is mostly self-explanatory, but it should be noted that the behavior of save operations in UDDI is generally destructive. An example is resaving the same service with different information, which results in the complete

### UDDI Version 2.0

UDDI defines four core data elements within the data model:

- `businessEntity` (modeling business information)
- `businessService` (describing a service)
- `tModel` (describing specifications, classifications, or identifications)
- `binding Template` (mapping between a `businessService` and the set of `tModels` that describe its technical

replacement of the prior entry for that service.

The operations associated with `authTokens` require you to preregister with a particular UDDI Business Registry node and provide credentials necessary to establish a publisher's identity. These credentials are used to obtain an `authToken` for purposes of performing a publishing operation (with the `save_XXX` APIs). This `authToken` may be reused for a short time during successive publishing operations provided it has not expired. Operator policy determines both the content and life of an `authToken`.

### What's new in UDDI version 2?

Version 2 of UDDI introduces some key features that improve the quality and efficiency of using a UDDI registry from the Version 1 specifications. The following sections describe the new features in Version 2:

- Modeling support for complex organizations
- More powerful categorization and identifier support for clients
- Enhanced inquiry
- Internationalization features
- Peer-based replication

### Modeling support

The updates associated with modeling support are targeted primarily at helping large organizations more efficiently model their businesses and services. Many companies, from large conglomerates involved in different ventures to focused ones that wish to partition their Web offerings by the geographies they serve, may wish to represent themselves on the Web as separate but related businesses. In Version 1 of UDDI, the only choice for these situations was to maintain separate and unrelated businesses. Version 2 allows you to define relationships between businesses, such as *parent-child*, *peer-peer*, and *identity*. This gives you the flexibility to model a business with subsidiaries, external business partners, or various internal divisions. Relationships may be formed between any two businesses (as defined by their unique business keys). The new canonical tModel for Business Relationship types supports this capability, together with several new publishing APIs that allow you to define, delete, and request the status of a relationship, as shown in [Listing 2](#).

A new inquiry API called `find_relatedBusinesses` enables anonymous searches of relationships involving a given company. The other new publishing APIs in [Listing 2](#) all pertain to the relationships that a particular publisher created and owns and are not accessible through anonymous inquiry. A business relationship consists of a pair of identical relationship assertions, each of which ties two businesses together and identifies the specific relationship being established. The owners of both of the two businesses involved must each declare the same relationship assertion in order to form an externally visible business relationship. Only these matched relationship assertions are returned as the result of a `find_relatedBusinesses()` inquiry.

The following example shows how a relationship is formed and then discovered. First, you get an authorization token for publishing:

### Listing 2. Getting an authToken

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/" >
  <Body>
<get_authToken generic="2.0"
  xmlns="urn:uddi-org:api_v2"
  userID="businessA_UserId"
  cred="businessA_Password" />
  </Body>
</Envelope>
```

fingerprint)

Beyond these core elements, Version 2 has added support for modeling relationships between businesses.

The UDDI Specification and the UDDI Business Registry, which provides a set of reference implementations that interoperate to share registrations, enable a programming model that supports either design-time or run-time service discovery, depending on the needs of the client. The just-in-time integration value proposition of the IBM Web Services Initiative combined with other important enabling technologies, such as the WSDL, allows organizations to perform dynamic discovery and binding of Web services at run time.

The continuing evolution of the UDDI Specification further increases the value of UDDI registries by addressing key areas, such as data integrity, access control, identification, and authentication, as well as interoperability, improved data modeling, query, and localization.

Which returns the following:

### Listing 3. The response to get authToken

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
<authToken generic="2.0"
  operator="www.ibm.com/services/uddi"
  xmlns="urn:uddi-org:api_v2" >
  <authInfo>businessA_AuthToken</authInfo>
</authToken>
  </Body>
</Envelope>
```

Next, you establish a business assertion relating two businesses, A and B, whose respective `businessKeys` are simplified here to `businessKeyA` and `businessKeyB` instead of the typical UUID format. The SOAP envelope is not shown, for brevity.

```
<add_publisherAssertions generic="2.0"
  xmlns="urn:uddi-org:api_v2">
  <authInfo>businessA_AuthToken</authInfo>
  <publisherAssertion>
  <fromKey>"businessKeyA"</fromKey>
<toKey>"businessKeyB"</toKey>
  <keyedReference keyValue="parent-child"
  keyName="Subsidiary"
  tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4" />
  </publisherAssertion>
</add_publisherAssertions>
```

Once an identical assertion is established by the owner of business B, a publicly visible business relationship will be visible in the registry. Next, you perform a simple inquiry using the `find_relatedBusinesses()` API:

```
<find_relatedBusinesses generic="2.0"
  xmlns="urn:uddi-org:api_v2">
  <businessKey>"businessKeyA"</businessKey>
</find_relatedBusinesses>
```

[Listing 3](#) shows the returned XML. A more fine-grained inquiry might include a `keyedReference` identifying the specific relationship being sought.

Another important feature added in Version 2 to support the modeling needs of large businesses is called a Service Projection, which allows one business to create a reference to a service owned by another business. This is useful in several scenarios, such as for a company that offers the same service across two or more of its businesses, or for a common service (for example, overnight shipping) whose use many businesses might want to encourage by linking it to the services they offer themselves.

A projected service reference is nothing more than that. The creator of a Service Projection cannot alter the actual service being referenced, but in all other respects, the service behaves as if it were owned by the business creating the projection. The service is returned as part of a `get_businessDetail()` or `get_serviceDetail()` API call. You differentiate a projected service from an actual one through the owning business key associated with the service. The key always matches the actual business that owns the service, rather than the business that created the service projection.

## Powerful categorization

In Version 1, three built-in taxonomies were provided for use in categorizing businesses and services. These were the NAICS industry categorizations, UNSPC project and service categorizations, and ISO-3166-2 geographic taxonomies. Use of these taxonomies is internally checked by the UDDI registry; attempts to save invalid codes are rejected. The importance of meaningful categorization in UDDI cannot be overstated. It is the single most powerful means of finding useful information of interest, and thus improving the ability of industry to create and control new taxonomies continues to be a priority.

Version 2 adds the ability for organizations to define new externally checked taxonomies, which can be offered for public use in UDDI. These external taxonomy suppliers must support a `validate_values` Web service and make it accessible to the UDDI Business Registry to support external checking and validating of taxonomy values that clients wish to associate with their registry entries. This is a controlled process. Registration of an externally validated taxonomy can be accomplished only with the approval of a UDDI Business Registry operator.

This new validation feature allows taxonomy providers the flexibility of ensuring that only valid taxonomy values are saved by clients using their taxonomy. When a client requests to use a provider's taxonomy, the provider may choose to perform a contextual validation along with validating that the requester is authorized to use the taxonomy. The more common non-contextual scenarios also support caching taxonomy data within the UDDI Business Registry to reduce dependence on the provider's external taxonomy service.

## Enhanced inquiry

Version 2 builds upon the inquiry capabilities previously provided by adding several powerful features dealing with more complex query requirements. To enhance existing `find_XXX` inquiry APIs, several new filter criteria (find qualifiers) have been added, including `orLikeKeys`, `orAllKeys`, `combineCategoryBags`, `serviceSubset`, and `andAllKeys`. Of these, `combineCategoryBags`, `serviceSubset`, and `orLikeKeys` are of particular interest.

The `combineCategoryBags` qualifier allows you to group all of the taxonomy data associated with a business and all of its contained services (including any service projections) into a single collection that the search is performed against. This is useful because it reduces the steps in finding a business of interest by looking in both the business and its constituent services at the same time.

Similarly, the `serviceSubset` filter allows you to search for businesses using taxonomy criteria, which are tested only against the categorizations associated with a business' constituent services. Categorizations associated with the business itself are not included in the search.

Finally, the `orLikeKeys` qualifier is particularly useful because it enables pseudo-complex queries. For example, you can find businesses located in the United States, Mexico, or Canada that also offer cold storage or generic shipping services. This enables you to search for businesses that are categorized with different levels of specificity, while at the same time allowing a single inquiry to reference multiple different taxonomies.

## Internationalization

Several new features have been added in an ongoing effort to address internationalization improvements in UDDI. Support for multiple names together with an `xml:lang` code has been added to `businessEntity` and `businessService` entities. You must provide at least one name, but you can provide multiple names (each with a different language code).

Another internationalization feature in Version 2 is the addition of a new type of taxonomy called `postalAddress`, which enables you to create a `tModel` describing localized postal addresses that then may be associated with the addresses found in business entities.

## Replication

While not directly visible to UDDI clients, significant improvements were made in replication between registry operators for Version 2. UDDI Version 1 only supported a file-based replication scheme whose complexity grew as  $n^2$  with the number of registry node implementations participating within a UDDI registry. Version 2 addresses the needs of a larger number of participating nodes that replicate with each other. Replication can proceed in a peer-based approach where registry updates from all other nodes can be obtained from any one node. Replication is now further supported by the definition of a set of APIs that allows processing of the changes and management of the process.

## What's ahead

The next version of the UDDI Specification planned for mid-2002 will focus on security, advanced data management, and further

internationalization. Security will be addressed by improving UDDI data integrity through enhanced access control, identity, and authentication. This will include support for existing and emerging security technologies from such organizations as W3C and OASIS. Advanced data management features will provide continued enhancements to the search capability to provide more concise and targeted queries, better interpretation of query results, the ability to capture richer and more meaningful descriptions of businesses and services, and easier management of existing data. Improvements to internationalization will include enhanced support for multinational corporations to describe their global operations across international business units and will address localization of UDDI data and services.

#### Conclusion

UDDI continues to evolve rapidly. Publicly available beta implementations of UDDI Version 2 were provided by multiple UDDI Business Registry operators in 2001. As Version 3 of the specification becomes available in mid-2002, the registry will continue to add features needed for conducting business through the Web, addressing the areas of security, improved internationalization, and additional features to support registry use and interoperability.

#### Resources

- Much more information on this topic is available in the [IBM developerWorks Web services zone](#).
- See also the [UDDI organization](#) home page.
- Learn more about [Web services and UDDI](#).
- Check out the developerWorks article [Programming with UDDI4J version 2](#).
- February 2002 [developerWorks journal](#) article: "Build and deploy your own Web services with UDDI4J"

#### About the author



Tom Bellwood is a Senior Technical Staff Member with IBM and has many years of experience in the technology sector, ranging from the world of semiconductor design and design automation to systems and application development. He is an expert on UDDI and how it supports the growing world of Web services. He is the technical lead for the IBM UDDI Business Registry and has spoken at many technical conferences.



developerWorks

[About IBM](#) | [Privacy](#) | [Legal](#) | [Contact](#)