UDDI

web services

tutorialspoint

SIMPLYEASYLEARNING

About the Tutorial

UDDI is an XML-based standard for describing, publishing, and finding Web services. In this tutorial, you will learn what is UDDI and why and how to use it.

Audience

This tutorial has been designed for beginners interested in learning the basic concepts of UDDI.

Prerequisites

Since UDDI is an XML-based standard, all that you need to have is a basic understanding of XML to make the most of this tutorial.

Copyright & Disclaimer

Table of Contents

1. OVERVIEW

What is UDDI?

UDDI is an XML-based standard for describing, publishing, and finding web services.

- UDDI stands for **Universal Description, Discovery, and Integration**.
- UDDI is a specification for a distributed registry of web services.
- UDDI is a platform-independent, open framework.
- UDDI can communicate via SOAP, CORBA, Java RMI Protocol.
- UDDI uses Web Service Definition Language (WSDL) to describe interfaces to web services.
- UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
- UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet.

UDDI has two sections:

- A registry of all web service's metadata, including a pointer to the WSDL description of a service.
- A set of WSDL port type definitions for manipulating and searching that registry.

History of UDDI

- UDDI 1.0 was originally announced by Microsoft, IBM, and Ariba in September 2000.
- Since the initial announcement, the UDDI initiative has grown to include more than 300 companies including Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP, and Sun.
- In May 2001, Microsoft and IBM launched the first UDDI operator sites and turned the UDDI registry live.
- In June 2001, UDDI announced Version 2.0.
- As the time of writing this tutorial, Microsoft and IBM sites had implemented the 1.0 specification and were planning 2.0 support in the near future.
- Currently UDDI is sponsored by OASIS.

Partner Interface Processes

Partner Interface Processes (PIPs) are XML-based interfaces that enable two trading partners to exchange data. Dozens of PIPs already exist. Some of them are listed here:

- **PIP2A2**: Enables a partner to query another for product information.
- **PIP3A2**: Enables a partner to query the price and availability of specific products.
- **PIP3A4**: Enables a partner to submit an electronic purchase order and receive acknowledgment of the order.
- **PIP3A3**: Enables a partner to transfer the contents of an electronic shopping cart.
- **PIP3B4**: Enables a partner to query the status of a specific shipment.

Private UDDI Registries

As an alternative to using the public federated network of UDDI registries available on the Internet, companies or industry groups may choose to implement their own private UDDI registries.

These exclusive services are designed for the sole purpose of allowing members of the company or of the industry group to share and advertise services amongst themselves.

Regardless of whether the UDDI registry is a part of the global federated network or a privately owned and operated registry, the one thing that ties them all together is a common web services API for publishing and locating businesses and services advertised within the UDDI registry.

2. ELEMENTS

A business or a company can register three types of information into a UDDI registry. This information is contained in three elements of UDDI.

These three elements are:

- White Pages,
- Yellow Pages, and
- Green Pages.

White Pages

White pages contain:

- Basic information about the company and its business.
- Basic contact information including business name, address, contact phone number, etc.
- Unique identifiers for the company tax IDs. This information allows others to discover your web service based upon your business identification.

Yellow Pages

- Yellow pages contain more details about the company. They include descriptions of the kind of electronic capabilities the company can offer to anyone who wants to do business with it.
- Yellow pages use commonly accepted industrial categorization schemes, industry codes, product codes, business identification codes and the like to make it easier for companies to search through the listings and find exactly what they want.

Green Pages

Green pages contain technical information about a web service. A green page allows someone to bind to a web service after it's been found. It includes:

- The various interfaces
- The URL locations
- Discovery information and similar data required to find and run the web service.

NOTE: UDDI is not restricted to describing web services based on SOAP. Rather, UDDI can be used to describe any service, from a single webpage or email address all the way up to SOAP, CORBA, and Java RMI services.

3. TECHNICAL ARCHITECTURE

The UDDI technical architecture consists of three parts:

UDDI Data Model

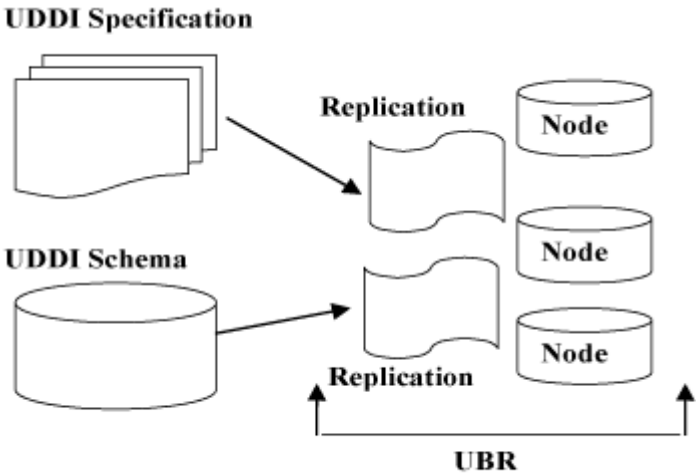
UDDI Data Model is an XML Schema for describing businesses and web services. The data model is described in detail in the "UDDI Data Model" chapter.

UDDI API Specification

It is a specification of API for searching and publishing UDDI data.

UDDI Cloud Services

These are operator sites that provide implementations of the UDDI specification and synchronize all data on a scheduled basis.



UDDI Technical Architecture

The UDDI Business Registry (UBR), also known as the Public Cloud, is a conceptually single system built from multiple nodes having their data synchronized through replication.

The current cloud services provide a logically centralized, but physically distributed, directory. It means the data submitted to one root node will automatically be replicated across all the other root nodes. Currently, data replication occurs every 24 hours.

UDDI cloud services are currently provided by Microsoft and IBM. Ariba had originally planned to offer an operator as well, but has since backed away from the commitment. Additional operators from other companies, including Hewlett-Packard, are planned for the near future.

It is also possible to set up private UDDI registries. For example, a large company may set up its own private UDDI registry for registering all internal web services. As these registries are not automatically synchronized with the root UDDI nodes, they are not considered as a part of the UDDI cloud.

4. DATA MODEL

UDDI includes an XML Schema that describes the following data structures:

- businessEntity
- businessService
- bindingTemplate
- tModel
- publisherAssertion

businessEntity Data Structure

The business entity structure represents the provider of web services. Within the UDDI registry, this structure contains information about the company itself, including contact information, industry categories, business identifiers, and a list of services provided.

Here is an example of a fictitious business's UDDI registry entry:

```
<businessEntity businessKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40"
        operator="http://www.ibm.com"
        authorizedName="John Doe">
    <name>Acme Company</name>
    <description>
        We create cool Web services
    </description>
    <contacts>
        <contact useType="general info">
            <description>General Information</description>
            <personName>John Doe</personName>
            <phone>(123) 123-1234</phone>
            <email>jdoe@acme.com</email>
        </contact>
    </contacts>
  <businessServices>
  ...
```

```
  </businessServices>
  <identifierBag>
     <keyedReference
        tModelKey="UUID:8609C81E-EE1F-4D5A-B202-3EB13AD01823"
        name="D-U-N-S"
        value="123456789" />
  </identifierBag>
  <categoryBag>
     <keyedReference
        tModelKey="UUID:C0B9FE13-179F-413D-8A5B-5004DB8E5BB2"
        name="NAICS"
        value="111336" />
   </categoryBag>
</businessEntity>
```

businessService Data Structure

The business service structure represents an individual web service provided by the business entity. Its description includes information on how to bind to the web service, what type of web service it is, and what taxonomical categories it belongs to.

Here is an example of a business service structure for the Hello World web service.

```
<businessService serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
        businessKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">
     <name>Hello World Web Service</name>
     <description>A friendly Web service</description>
     <bindingTemplates>
     ...
     </bindingTemplates>
     <categoryBag />
</businessService
```

Notice the use of the Universally Unique Identifiers (UUIDs) in the *businessKey* and *serviceKey* attributes. Every business entity and business service is uniquely identified in all the UDDI registries through the UUID assigned by the registry when the information is first entered.

bindingTemplate Data Structure

Binding templates are the technical descriptions of the web services represented by the business service structure. A single business service may have multiple binding templates. The binding template represents the actual implementation of the web service.

Here is an example of a binding template for Hello World.

```
<bindingTemplate serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
         bindingKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">
     <description>Hello World SOAP Binding</description>
     <accessPoint URLType="http">
         http://localhost:8080
     </accessPoint>
     <tModelInstanceDetails>
         <tModelInstanceInfo
               tModelKey="uuid:EB1B645F-CF2F-491f-811A-4868705F5904">
         <instanceDetails>
              <overviewDoc>
                 <description>
                     references the description of the
                     WSDL service definition
                 </description>
                 <overviewURL>
                      http://localhost/helloworld.wsdl
                 </overviewURL>
              </overviewDoc>
         </instanceDetails>
         </tModelInstanceInfo>
      </tModelInstanceDetails>
</bindingTemplate>
```

As a business service may have multiple binding templates, the service may specify different implementations of the same service, each bound to a different set of protocols or a different network address.

tModel Data Structure

tModel is the last core data type, but potentially the most difficult to grasp. tModel stands for technical model.

tModel is a way of describing the various business, service, and template structures stored within the UDDI registry. Any abstract concept can be registered within the UDDI as a tModel. For instance, if you define a new WSDL port type, you can define a tModel that represents that port type within the UDDI. Then, you can specify that a given business service implements that port type by associating the tModel with one of that business service's binding templates.

Here is an example of a tModel representing the Hello World Interface port type.

```
<tModel tModelKey="uuid:xyz987..."
           operator="http://www.ibm.com"
           authorizedName="John Doe">
    <name>HelloWorldInterface Port Type</name>
    <description>
           An interface for a friendly Web service
    </description>
    <overviewDoc>
           <overviewURL>
                http://localhost/helloworld.wsdl
           </overviewURL>
     </overviewDoc>
</tModel>
```

publisherAssertion Data Structure

This is a relationship structure putting into association two or more businessEntity structures according to a specific type of relationship, such as subsidiary or department.

The publisherAssertion structure consists of the three elements: fromKey (the first businessKey), toKey (the second businessKey), and keyedReference.

The keyedReference designates the asserted relationship type in terms of a keyName - keyValue pair within a tModel, uniquely referenced by a tModelKey.

```
<element name="publisherAssertion" type="uddi:publisherAssertion" />
<complexType name="publisherAssertion">
  <sequence>
```

```
    <element ref="uddi:fromKey" />
    <element ref="uddi:toKey" />
    <element ref="uddi:keyedReference" />
  </sequence>
</complexType>
```

5. INTERFACES

A registry is of no use without some way to access it. The UDDI standard version 2.0 specifies two interfaces for service consumers and service providers to interact with the registry.

Service consumers use **Inquiry Interface** to find a service, and service providers use **Publisher Interface** to list a service.

The core of the UDDI interface is the UDDI XML Schema definitions. These define the fundamental UDDI data types through which all the information flows.

The Publisher Interface

The Publisher Interface defines sixteen operations for a service provider managing its entries in the UDDI registry:

- **get_authToken**: Retrieves an authorization token. All of the Publisher interface operations require that a valid authorization token be submitted with the request.
- **discard_authToken**: Tells the UDDI registry to no longer accept a given authorization token. This step is equivalent to logging out of the system.
- **save_business**: Creates or updates a business entity's information contained in the UDDI registry.
- **save_service**: Creates or updates information about the web services that a business entity provides.
- **save_binding**: Creates or updates the technical information about a web service's implementation.
- **save_tModel**: Creates or updates the registration of abstract concepts managed by the UDDI registry.
- **delete_business**: Removes the given business entities from the UDDI registry completely.
- **delete_service**: Removes the given web services from the UDDI registry completely.
- **delete_binding**: Removes the given web services' technical details from the UDDI registry.
- **delete_tModel**: Removes the specified tModels from the UDDI registry.
- **get_registeredInfo**: Returns a summary of everything the UDDI registry is currently keeping track of for the user, including all businesses, all services, and all tModels.

- **set_publisherAssertions**: Manages all of the tracked relationship assertions associated with an individual publisher account.
- **add_publisherAssertions**: Causes one or more publisherAssertions to be added to an individual publisher's assertion collection.
- **delete_publisherAssertions**: Causes one or more publisherAssertion elements to be removed from a publisher's assertion collection.
- **get_assertionStatusReport**: Provides administrative support for determining the status of current and outstanding publisher assertions that involve any of the business registrations managed by the individual publisher account.
- **get_publisherAssertions**: Obtains the full set of publisher assertions that is associated with an individual publisher account.

The Inquiry Interface

The inquiry interface defines ten operations for searching the UDDI registry and retrieving details about specific registrations:

- **find_binding**: Returns a list of web services that match a particular set of criteria based on the technical binding information.
- **find_business**: Returns a list of business entities that match a particular set of criteria.
- **find_ltservice**: Returns a list of web services that match a particular set of criteria.
- **find_tModel**: Returns a list of tModels that match a particular set of criteria.
- **get_bindingDetail**: Returns the complete registration information for a particular web service binding template.
- **get_businessDetail**: Returns the registration information for a business entity, including all services that entity provides.
- **get_businessDetailExt**: Returns the complete registration information for a business entity.
- **get_serviceDetail**: Returns the complete registration information for a web service.
- **get_tModelDetail**: Returns the complete registration information for a tModel.
- **find_relatedBusinesses**: Discovers businesses that have been related via the uddi-org:relationships model.

6. USAGE EXAMPLE

Consider a company XYZ wants to register its contact information, service description, and online service access information with UDDI. The following steps are necessary:

- Choose an operator with which to work. Each operator has different terms and conditions for authorizing access to its replica of the registry.
- Build or otherwise obtain a UDDI client, such as those provided by the operators.
- Obtain an authentication token from the operator.
- Register information about the business. Include as much information as might be helpful to those searching for matches.
- Release the authentication token.
- Use the inquiry APIs to test the retrieval of the information, including binding template information, to ensure that someone who obtains it can use it successfully to interact with your service.
- Fill in the tModel information in case someone wants to search for a given service and find your business as one of the service providers.
- Update the information as necessary to reflect the changing business contact information and new service details, obtaining and releasing a new authentication token from the operator each time. Whenever you need to update or modify the data you've registered, you have to go back to the operator with which you have entered the data.

The following examples will show how the XYZ Company would register its information and how a distributor interested in carrying the XYZ's product line might find information about how to contact the company and place an order, using the XYZ.com Web services.

Creating Registry

After obtaining an authentication token from one of the operators – Microsoft, for example – the XYZ.com developers decide what information to publish to the registry and use one of the UDDI tools provided by Microsoft. If necessary, the developers can also write a Java, C#, or VB.NET program to generate the appropriate SOAP messages. Here is an example.

```
POST /save_business HTTP/1.1
Host: www.XYZ.com
Content-Type: text/xml; charset="utf-8"
```

```
Content-Length: nnnn
SOAPAction: "save_business"
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas/xmlsoap.org/soap/envelope/">
      <Body>
         <save_business generic="2.0" xmlns="urn:uddi-org:api_v2">
         <businessKey="">
         </businessKey>
         <name>
             XYZ, Pvt Ltd.
         </name>
         <description>
             Company is involved in giving Stat-of-the-art....
         </description>
         <identifierBag> ... </identifierBag>
         ...
         </save_business>
      </Body>
</Envelope>
```

This example illustrates a SOAP message requesting to register a UDDI business entity for XYZ Company. The key element is blank, because the operator automatically generates the UUID key for the data structure. Most fields are omitted for the sake of showing a simple example.

Company XYZ can always execute another save_business operation to add to the basic information required to create a business entity.

Retrieving Information

After XYZ Company has updated its UDDI entry with the relevant information, companies that want to become XYZ distributors can look up contact information in the UDDI registry and obtain the service descriptions and the access points for the two Web services that XYZ.com publishes for online order entry: preseason bulk orders and in-season restocking orders.

This example illustrates a sample SOAP request to obtain business detail information about the XYZ Company. Once you know the UUID, or key, for the specific business that's been registered, you can use it in the get_businessDetail API to return specific information about that business.

```
POST /get_businessDetail HTTP/1.1
Host: www.XYZ.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "get_businessDetail"
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas/xmlsoap.org/soap/envelope/">
   <Body>
      <get_businessDetail generic="2.0" xmlns="urn:uddi-org:api_v2">
          <businessKey="C90D731D-772HSH-4130-9DE3-5303371170C2">
      </businessKey>
      </get_businessDetail>
   </Body>
</Envelope>
```

7. UDDI WITH WSDL

The UDDI data model defines a generic structure for storing information about a business and the web services it publishes. The UDDI data model is completely extensible, including several repeating sequence structures of information. However, WSDL is used to describe the interface of a web service. WSDL is fairly straightforward to use with UDDI.

- WSDL is represented in UDDI using a combination of *businessService, bindingTemplate,* and *tModel* information.
- As with any service registered in UDDI, generic information about the service is stored in the *businessService* data structure, and information specific to how and where the service is accessed is stored in one or more associated *bindingTemplate* structures. Each *bindingTemplate* structure includes an element that contains the network address of the service and has associated with it one or more *tModel* structures that describe and uniquely identify the service.
- When UDDI is used to store WSDL information, or pointers to WSDL files, the *tModel* should be referred to by convention as type *wsdlSpec*, meaning that the *overviewDoc* element is clearly identified as pointing to a WSDL service interface definition.
- For UDDI, WSDL contents are split into two major elements: the interface file, and the implementation file.

The Hertz reservation system web service provides a concrete example of how UDDI and WSDL works together. Here is the <tModel> for this web service:

```
<tModel authorizedName="..." operator="..." tModelKey="...">
  <name>HertzReserveService</name>
       <description xml:lang="en">
          WSDL description of the Hertz reservation service interface
       </description>
       <overviewDoc>
         <description xml:lang="en">
                WSDL source document.
            </description>
         <overviewURL>
                http://mach3.ebphost.net/wsdl/hertz_reserve.wsdl
            </overviewURL>
```

```
        </overviewDoc>
        <categoryBag>
            <keyedReference
               tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
               keyName="uddi-org:types" keyValue="wsdlSpec"/>
        </categoryBag>
</tModel>
```

The key points are:

- The overviewURL element gives the URL to where the service interface definition WSDL file can be found. This allows humans and UDDI/WSDL aware tools to locate the service interface definition.
- The purpose of the keyedReference element in the categoryBag is to make sure that this tModel is categorized as a WSDL specification document.

8. IMPLEMENTATIONS

A number of UDDI implementations are currently available. These implementations make it easier to search or publish UDDI data, without getting mired in the complexities of the UDDI API. Here is a brief synopsis of the main UDDI implementations available.

Java Implementations

There are two UDDI implementations for Java.

- **UDDI4J (UDDI for Java)**: UDDI4J was originally created by IBM. In January 2001, IBM turned over the code to its own open source site. UDDI4J is a Java class library that provides an API to interact with a UDDI.
- **jUDDI:** jUDDI is an open source Java implementation of a UDDI registry and a toolkit for accessing UDDI services.

Perl Implementation

- **UDDI::Lite**: It provides a basic UDDI client for inquiry and publishing.

Ruby Implementation

- **UDDI4r**: It provides a basic UDDI client for inquiry and publishing.

Python Implementation

- **UDDI4Py**: UDDI4Py is a Python package that allows the sending of requests to, and processing of responses from the UDDI Version 2 APIs.

9. SPECIFICATIONS

The UDDI project also defines a set of XML Schema definitions that describe the data formats used by the various specification APIs. These documents are all available for download at www.uddi.org. The current version of all specification groups is Version 2.0.

The specifications include the following:

- UDDI Replication,
- UDDI Operators,
- UDDI Programmer's API, and
- UDDI Data Structures.

UDDI Replication

This document describes the data replication processes and interfaces to which a registry operator must conform to achieve data replication between sites. This specification is not a programmer's API; it defines the replication mechanism used among UBR nodes.

UDDI Operators

This document outlines the behavior and operational parameters required by the UDDI node operators. This specification defines data management requirements to which operators must adhere.

UDDI Programmer's API

This specification defines a set of functions that all UDDI registries support for inquiring about services hosted in a registry and for publishing information about a business or a service to a registry. This specification defines a series of SOAP messages containing XML documents that a UDDI registry accepts, parses, and responds to. This specification, along with the UDDI XML API schema and the UDDI Data Structure specification, makes up a complete programming interface to a UDDI registry.

UDDI Data Structures

This specification covers the specifics of the XML structures contained within the SOAP messages defined by the UDDI Programmer's API. This specification defines five core data structures and their relationships with one another.

The UDDI XML API schema is not contained in a specification; rather, it is stored as an XML Schema document that defines the structure and datatypes of the UDDI data structures.

10. SUMMARY

We have learned UDDI and its elements in this tutorial and have also seen the complete architecture and the data model of UDDI.

We have learned about the two UDDI interfaces: Publisher's Interface and Enquiry Interface. We have also learned how to register and search for web services with UDDI.

What's Next?

The next step is to learn about SOAP, WSDL, and Web Services.

SOAP

SOAP is a simple XML-based protocol that allows applications to exchange information over HTTP.

If you want to learn more about SOAP, please visit our SOAP tutorial.

WSDL

WSDL is the standard format for describing a web service in XML format.

WSDL is an integral part of UDDI.

If you want to learn more about WSDL, please visit our WSDL Tutorial.

Web Services

Web services can convert your applications into web-applications.

If you want to learn more about web services, please visit our Web Services tutorial.

11. API QUICK REFERENCE

Here is the complete reference of the UDDI Enquiry APIs and the UDDI Publishing APIs.

The UDDI Inquiry APIs

API Name	Description	V1.0	V2.0
find_binding	Searches for template bindings associated with a specified service.	Y	Y
find_business	Searches for business that matches the specified criteria.	Y	Y
find_relatedBusinesses	Discovers business that have been related via the uddi-org:relationships model.		Y
find_service	Searches for service associated with a specified business.	Y	Y
find_tModel	Searches for tModel records that matches the specified criteria.	Y	Y
get_bindingDetail	Retrieves the complete bindingTemplate for each specified bindingKey.	Y	Y
get_businessDetail	Retrieves the complete businessEntity for each specified businessKey.	Y	Y
get_businessDetailExt	Retrieves the extended businessEntity for each specified businessKey.	Y	Y
get_serviceDetail	Retrieves the businessService record for each specified serviceKey.	Y	Y
get_tModelDetail	Retrieves the tModel record for each specified tModelKey.	Y	Y

The UDDI Publishing APIs

API Name	Description	V1.0	V2.0
get_authToken	Retrieves an authorization token. All of the Publisher interface operations require that a valid authorization token be submitted with the request.	Y	Y
discard_authToken	Tells the UDDI registry to no longer accept a given authorization token. This step is equivalent to logging out of the system.	Y	Y
save_business	Creates or updates a business entity's information contained in the UDDI registry.	Y	Y
save_service	Creates or updates information about the web services that a business entity provides.	Y	Y
save_binding	Creates or updates the technical information about a web service's implementation.	Y	Y
save_tModel	Creates or updates the registration of abstract concepts managed by the UDDI registry.	Y	Y
delete_business	Removes the given business entities from the UDDI registry completely.	Y	Y
delete_service	Removes the given web services from the UDDI registry completely.	Y	Y
delete_binding	Removes the given web service technical details from the UDDI registry.	Y	Y
delete_tModel	Removes the specified tModels from the UDDI registry.	Y	Y

get_registeredInfo	Returns a summary of everything the UDDI registry is currently keeping track of for the user, including all businesses, all services, and all tModels.	Y	Y
set_publisherAssertions	Manages all of the tracked relationship assertions associated with an individual publisher account.		Y
add_publisherAssertions	Causes one or more publisherAssertions to be added to an individual publisher's assertion collection.		Y
delete_publisherAssertions	Causes one or more publisherAssertion elements to be removed from a publisher's assertion collection.		Y
get_assertionStatusReport	Provides administrative support for determining the status of current and outstanding publisher assertions that involve any of the business registrations managed by the individual publisher account.		Y
get_publisherAssertions	Obtains the full set of publisher assertions that is associated with an individual publisher account.		Y

find_binding

Description: The find_bindings function searches for template binding records associated with a specified service and the specified tModel record(s).

The response includes a root bindingDetail element and one bindingTemplate element for each matching binding. If the UDDI operator returns only a partial list of matching results, the bindingDetail element's truncated attribute will be set to true. If no matches are found, a bindingDetail element with zero sub elements is returned.

Version 2.0 Syntax

```
<find_binding serviceKey="uuid_key" [maxRows="nn"] generic="2.0"
xmlns="urn:uddi-org:api_v2">
   [<findQualifiers/>]
   <tModelBag/>
</find_binding>
```

Arguments

serviceKey: Required uuid_key attribute specifying the associated businessService.

maxRows: Optional attribute to specify the maximum number of rows to be returned; if maxRows is exceeded, the bindingDetail element's truncated attribute will be set to true.

findQualifiers: Optional element to override the default search functionality.

tModelBag: Required uuid_key element to specify tModel records. If more than one tModel is specified, the search is performed via a logical AND.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: It signifies that the uuid_key value passed did not match with any known serviceKey or tModelKey values. The error structure will signify which condition occurred first, and the invalid key will be indicated clearly in text.

E_unsupported: It signifies that one of the findQualifier values passed was invalid. The invalid qualifier will be indicated clearly in text.

find_business

Description: The find_business function searches for businesses that match the specified criteria.

The response includes a root businessList element, and one businessInfo element for each matching company. If the UDDI operator returns only a partial list of matching results, the businessList element's truncated attribute will be set to true. If no matches are found, a businessList element with zero sub elements is returned.

Version 2.0 Syntax

```
<find_business generic="2.0" [maxRows="nn"]
```

```
     xmlns="urn:uddi-org:api_v2">
     [<findQualifiers/>]
     [<name/> [<name/>]...]
     [<discoveryURLs/>]
     [<identifierBag/>]
     [<categoryBag/>]
     [<tModelBag/>]
</find_business>
```

Arguments

maxRows: Optional attribute to specify the maximum number of rows to be returned; if maxRows is exceeded, the bindingDetail element's truncated attribute will be set to true.

findQualifiers: Optional element to override the default search functionality. For example, the find qualifier exactNameMatch will match exact business names.

name: The full or partial name of the business. UDDI 2.0 enables you to specify up to five business names.

discoveryURLs: Optional element to search by discovery URLs. If more than one discoveryURL is specified, the search is performed via a logical OR.

identifierBag: Optional element to search by identifier. If more than one identifier is specified, the search is performed via a logical OR.

categoryBag: Optional element to search by category. For example, you can search by NAICS codes. If more than one category is specified, the search is performed via a logical AND.

tModelBag: Optional element to search by tModel records. If more than one tModel is specified, the search is performed via a logical AND.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: It signifies that the uuid_key value passed did not match with any known serviceKey or tModelKey values. The error structure will signify which condition occurred first, and the invalid key will be indicated clearly in text.

E_unsupported: It signifies that one of the findQualifier values passed was invalid. The invalid qualifier will be indicated clearly in text.

E_tooManyOptions: It signifies that the implementation defined limit on the number of name arguments was exceeded.

find_relatedBusinesses

Description: The find_relatedBusinesses function is used to discover businesses that have been related via the uddi-org:relationships model.

The response includes a root relatedBusinessesList element. If the UDDI operator returns only a partial list of matching results, the relatedBusinessesList element's truncated attribute will be set to true. If no matches are found, a relatedBusinessesList element with zero sub elements is returned.

Version 2.0 Syntax

```
<find_relatedBusinesses generic="2.0" xmlns="urn:uddi-org:api_v2">
     [<findQualifiers/>]
     <businessKey/>
     [<keyedReference/>]
</find_relatedBusinesses>
```

Arguments

findQualifiers: Optional element to override the default search functionality. For example, the find qualifier exactNameMatch will match exact business names.

businessKey: Required uuid_key specifying the businessEntity.

keyedReference: Optional element used to specify a uddi-org:relationship value. A keyedReference requires three attributes: tModelKey, keyName, and keyValue.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: It signifies that the uuid_key value passed did not match with any known serviceKey or tModelKey values. The error structure will signify which condition occurred first, and the invalid key will be indicated clearly in text.

E_unsupported: It signifies that one of the findQualifier values passed was invalid. The invalid qualifier will be indicated clearly in text.

find_service

Description: The find_service function searches for services associated with a specified business.

The response includes a root serviceList element, and one serviceInfo element for each matching company. If the UDDI operator returns only a partial list of matching results, the serviceList element's truncated attribute will be set to true. If no matches are found, a serviceList element with zero sub elements is returned.

Version 2.0 Syntax

```
<find_service businessKey="uuid_key" generic="2.0" [maxRows="nn"]
     xmlns="urn:uddi-org:api_v2">
     [<findQualifiers/>]
     [<name/> [<name/>]...]
     [<categoryBag/>]
     [<tModelBag/>]
</find_service>
```

Arguments

businessKey: Required uuid_key attribute specifying the associated businessEntity.

maxRows: Optional attribute to specify the maximum number of rows to be returned. If maxRows is exceeded, the serviceList element's truncated attribute will be set to true.

findQualifiers: Optional element to override the default search functionality. For example, the find qualifier exactNameMatch will match exact business names.

name: The full or partial name of the service. UDDI 2.0 allows you to specify up to five service names.

categoryBag: Optional element to search by category. If more than one category is specified, the search is performed via a logical AND.

tModelBag: Optional element to search by tModels. If more than one tModel is specified, the search is performed via a logical AND.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: It signifies that the uuid_key value passed did not match with any known serviceKey or tModelKey values. The error structure will signify which condition occurred first, and the invalid key will be indicated clearly in text.

E_tooManyOptions: Too many search options were specified.

E_unsupported: The specified findQualifier is not supported.

find_tModel

Description: The find_tModel function searches for tModel records that match the specified criteria.

The response includes a root tModelList element and one tModelInfo element for each matching company. If the UDDI operator returns only a partial list of matching results, the tModelList element's truncated attribute is set to true. If no matches are found, a tModelList element with zero sub elements is returned.

Version 2.0 Syntax

```
<find_tModel generic="2.0" [maxRows="nn"]
        xmlns="urn:uddi-org:api_v2">
    [<findQualifiers/>]
    [<name/>]
    [<identifierBag/>]
    [<categoryBag/>]
</find_tModel>
```

Arguments

maxRows: Optional attribute to specify the maximum number of rows to be returned. If maxRows is exceeded, the serviceList element's truncated attribute is set to true.

findQualifiers: Optional element to override the default search functionality. For example, the find qualifier exactNameMatch will match exact business names.

name: The full or partial name of the service. UDDI 2.0 allows you to specify up to five service names.

identifierBag: Optional element to search by identifier. If more than one identifier is specified, the search is performed via a logical OR.

categoryBag: Optional element to search by category. If more than one category is specified, the search is performed via a logical AND.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_unsupported: It signifies that one of the findQualifier values passed was invalid. The invalid qualifier is clearly indicated in the error text.

get_bindingDetail

Description: The get_bindingDetail function retrieves the complete bindingTemplate for each specified bindingKey.

The response includes a root bindingDetail element, and one bindingTemplate element for each matching binding. If the UDDI operator returns only a partial list of matching results, the bindingDetail element's truncated attribute will be set to true. If no matches are found, an E_invalidKeyPassed error is returned.

Version 2.0 Syntax

```
<get_bindingDetail generic="2.0" xmlns="urn:uddi-org:api_v2" >
     <bindingKey/>
     [<bindingKey/> ...]
</get_bindingDetail>
```

Arguments

businessKey: Required uuid_key attribute specifying the associated businessEntity.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: An invalid bindingKey was specified.

get_businessDetail

Description: The get_businessDetail function retrieves the complete businessEntity for each specified businessKey.

The response includes a root businessDetail element, and one businessEntity element for each matching business. If the UDDI operator returns only a partial list of matching results, the businessDetail truncated attribute is set to true. If no matches are found, an E_invalidKeyPassed error is returned.

Version 2.0 Syntax

```
<get_businessDetail generic="2.0" xmlns="urn:uddi-org:api_v2">
     <businessKey/>
     [<businessKey/> ...]
</get_businessDetail>
```

Arguments

businessKey: Required uuid_key specifying the businessEntity. You can specify multiple businessKeys.

Error Returned

If any error occurs in processing this API call, a dispositionReport element is returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: An invalid bindingKey was specified.

get_businessDetailExt

Description: The get_businessDetailExt function retrieves the extended businessEntity for each specified businessKey.

The response includes a root businessDetailExt element, and one businessEntityExt element for each matching business. If the operator returns only a partial list of matching results, the businessDetailExt element's truncated attribute is set to true. If no matches are found, an E_invalidKeyPassed error is returned. This function is useful for querying external UDDI registries that are not part of the UDDI cloud services and that may contain extra business registration information. When querying a UDDI operator site, this method returns the exact same results as get_businessDetail.

Version 2.0 Syntax

```
<get_businessDetailExt generic="2.0" xmlns="urn:uddi-org:api_v2" >
     <businessKey/>
     [<businessKey/> ...]
</get_businessDetailExt>
```

Arguments

businessKey: Required uuid_key specifying the businessEntity. You can specify multiple businessKeys.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: An invalid bindingKey was specified.

E_unsupported: The query is not supported. If this occurs, use the get_businessDetail query.

get_serviceDetail

Description: The get_serviceDetail function retrieves the businessService record for each specified serviceKey.

The response includes a root serviceDetail element, and one businessService element for each matching service. If the UDDI operator returns only a partial list of matching results, the serviceDetail element's truncated attribute will be set to true. If no matches are found, an E_invalidKeyPassed error is returned.

Version 2.0 Syntax

```
<get_serviceDetail generic="2.0" xmlns="urn:uddi-org:api_v2">
     <serviceKey/>
     [<serviceKey/> ...]
</get_serviceDetail>
```

Arguments

serviceKey: Required uuid_key specifying the serviceDetail. You can specify multiple serviceKeys.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: An invalid serviceKey was specified.

get_tModelDetail

Description: The get_tModelDetail function retrieves the tModel record for each specified tModelKey.

The response includes a root tModelDetail element, and one tModel element for each matching tModel. If the UDDI operator returns only a partial list of matching results, the tModelDetail element's truncated attribute will be set to true. If no matches are found, an E_invalidKeyPassed error is returned.

Version 2.0 Syntax

```
<get_tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2">
     <tModelKey/>
     [<tModelKey/> ...]
</get_tModelDetail>
```

Arguments

tModelKey: Required uuid_key specifying the tModel. You can specify multiple tModelKeys.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: An invalid tModelKey was specified.

get_authToken

Description: The get_authToken API call is used to obtain an authentication token. Authentication tokens are opaque values that are required for all other publisher API calls. This API is provided for implementations that do not have some other method of obtaining an authentication token or certificate, or that choose to use user ID and password based authentication.

This function returns an authToken message that contains a valid authInfo element that can be used in subsequent calls to publisher API calls that require an authInfo value.

Version 2.0 Syntax

```
<get_authToken generic="2.0" xmlns="urn:uddi-org:api_v2"
    userID="someLoginName"
    cred="someCredential" />
```

Arguments

userID: This required attribute argument is the user identifier that an individual authorized user was assigned by an Operator Site.

cred: This required attribute argument is the password or credential that is associated with the user.

Error Returned

If any error occurs in processing this API call, a dispositionReport element is returned to the caller within a SOAP Fault. The following error number information is relevant:

E_unknownUser: It signifies that the Operator Site that received the request does not recognize the userID and/or cred argument values passed as valid credentials.

discard_authToken

Description: The discard_authToken API call is used to inform an Operator Site that the authentication token is to be discarded, effectively ending the session. Subsequent calls that use the same authToken will be rejected. This message is optional for Operator Sites that do not manage session state or that do not support the get_authToken message.

Upon successful completion, a dispositionReport is returned with a single success indicator. Discarding an expired authToken is processed and reported as a success condition.

Version 2.0 Syntax

```
<discard_authToken generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
</discard_authToken>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

save_business

Description: The save_business API call is used to save or update information about a complete businessEntity element.

This API returns a businessDetail message containing the final results of the call that reflects the new registered information for the businessEntity information provided. These results will include any businessServices that are contained by reference. If

the same entity determined by matching key is listed more than once in the save_business message, it may be listed once in the result for each appearance in the save_business message. If so, the last appearance in the results represents the final saved state.

Version 2.0 Syntax

```
<save_business generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <businessEntity/> [<businessEntity/>.]
</save_business>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

businessEntity: One or more complete businessEntity elements can be passed. These elements can be obtained in advance by using the get_businessDetail API call or by any other means.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

E_invalidKeyPassed: It signifies that the request cannot be satisfied because one or more uuid_key values specified is not a valid key value. This includes any tModelKey references, as well as references to serviceKey or bindingKey values that either do not exist.

E_invalidProjection: It signifies that an attempt was made to save a businessEntity containing a service projection that does not match the businessService being projected. The serviceKey of at least one such businessService will be included in the dispositionReport.

E_userMismatch: It signifies that one or more of the uuid_key values passed refers to data that is not controlled by the individual who is represented by the authentication token. The key causing the error will be clearly indicated in the error text.

E_invalidValue: A value that was passed in a keyValue attribute did not pass validation. This applies to checked categorizations, identifiers and other validated code lists. The error text will clearly indicate the key and value combination that failed validation.

E_requestTimeout: It signifies that the request could not be carried out because a needed validate_values service did not respond in a reasonable amount of time. Details identifying the failing service will be included in the dispositionReport element.

E_valueNotAllowed: Restrictions have been placed by the taxonomy provider on the types of information that should be included at that location within a specific taxonomy. A validate_values service chosen by the Operator Site has rejected this businessEntity for at least one specified category.

E_accountLimitExceeded: It signifies that user account limits have been exceeded.

save_service

Description: The save_service API call adds or updates one or more businessService elements.

This API call returns a serviceDetail message containing the final results of the call that reflects the newly registered information for the effected businessService elements. In cases where multiple businessService elements are passed in the request, the result will contain the final results for each businessService passed and these will occur in the same order as found in the request. If the same entity is listed more than once in the save_service message, it may be listed once in the result for each appearance in the save_service message. If so, the last appearance in the results represents the final saved state.

Version 2.0 Syntax

```
<save_service generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <businessService/> [<businessService/>.]
</save_service>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

businessService: One or more complete businessService elements can be passed. For the purpose of performing round trip updates, this data can be obtained in advance by using the get_serviceDetail API call or by any other means.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

E_invalidKeyPassed: It signifies that the request cannot be satisfied because one or more uuid_key values specified is not a valid key value. This includes any tModelKey references, as well as references to serviceKey or bindingKey values that either do not exist.

E_userMismatch: It signifies that one or more of the uuid_key values passed refers to data that is not controlled by the individual who is represented by the authentication token. The key causing the error will be clearly indicated in the error text.

E_invalidValue: A value that was passed in a keyValue attribute did not pass validation. This applies to checked categorizations, identifiers and other validated code lists. The error text will clearly indicate the key and value combination that failed validation.

E_requestTimeout: It signifies that the request could not be carried out because a needed validate_values service did not respond in a reasonable amount of time. Details identifying the failing service will be included in the dispositionReport element.

E_valueNotAllowed: Restrictions have been placed by the taxonomy provider on the types of information that should be included at that location within a specific taxonomy. A validate_values service chosen by the Operator Site has rejected this businessEntity for at least one specified category.

E_accountLimitExceeded: It signifies that user account limits have been exceeded.

save_binding

Description: The save_binding API call is used to save or update a complete bindingTemplate element. This message can be used to add or update one or more bindingTemplate elements as well as the container/contained relationship that each bindingTemplate has with one or more existing businessService elements.

This API returns a bindingDetail message containing the final results of the call that reflects the newly registered information for the effected bindingTemplate elements. If more than one bindingTemplate is saved in a single save_binding message, the resulting bindingDetail message will return results in the same order that they appeared in the save_binding message. If the same bindingTemplate (determined by matching bindingKey) is listed more than once in the save_binding message, it may

be listed once in the result for each appearance in the save_binding message. If so, the last appearance in the results represents the final saved state.

Version 2.0 Syntax

```
<save_binding generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <bindingTemplate/> [<bindingTemplate/>.]
</save_binding>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

bindingTemplate: One or more complete bindingTemplate elements. To save a new bindingTemplate, pass a bindingTemplate element with an empty bindingKey attribute value. Any bindingTemplate data saved in this way must provide a serviceKey value that references a businessService that is controlled by the same publisher.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

E_invalidKeyPassed: It signifies that the request cannot be satisfied because one or more uuid_key values specified is not a valid key value, or that a hostingRedirector value references a bindingTemplate that itself contains a hostingRedirector value.

E_userMismatch: It signifies that one or more of the uuid_key values passed refers to data that is not controlled by the individual who is represented by the authentication token. The key causing the error will be clearly indicated in the error text.

E_accountLimitExceeded: It signifies that user account limits have been exceeded.

save_tModel

Description: The save_tModel API call adds or updates one or more registered tModel elements.

This API returns a tModelDetail message containing the final results of the call that reflects the new registered information for the effected tModel elements. If multiple tModel elements were passed in the save_tModel request, the order of the response will exactly match the order the elements appeared in the save. If the same tModel, determined by matching key, is listed more than once in the save_tModel message, it may be listed once in the result for each appearance in the save_tModel message. If so, the last appearance in the results represents the final saved state.

Version 2.0 Syntax

```
<save_tModel generic="2.0" xmlns="urn:uddi-org:api_v2" >
     <authInfo/>
     <tModel/> [<tModel/>.]
</save_tModel>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

tModel: One or more complete tModel elements can be passed. If adding a new tModel, the tModelKey value should be passed as an empty element.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

E_invalidKeyPassed: It signifies that the request cannot be satisfied because one or more uuid_key values specified is not a valid key value. This will occur if a uuid_key value is passed in a tModel that does not match with any known tModel key. The key value that causes an error will be indicated clearly in the error text.

E_userMismatch: It signifies that one or more of the uuid_key values passed refers to data that is not controlled by the individual who is represented by the authentication token. The key causing the error will be clearly indicated in the error text.

E_invalidValue: A value that was passed in a keyValue attribute did not pass validation. This applies to checked categorizations, identifiers and other validated code lists. The error text will clearly indicate the key and value combination that failed validation.

E_requestTimeout: It signifies that the request could not be carried out because a needed validate_values service did not respond in a reasonable amount of time. Details identifying the failing service will be included in the dispositionReport element.

E_valueNotAllowed: Restrictions have been placed by the taxonomy provider on the types of information that should be included at that location within a specific taxonomy. A validate_values service chosen by the Operator Site has rejected this businessEntity for at least one specified category.

E_accountLimitExceeded: It signifies that user account limits have been exceeded.

delete_business

Description: The delete_business API call is used to remove one or more business registrations (e.g. registered businessEntity data) and all direct contents from a UDDI registry.

Upon successful completion, a dispositionReport message is returned with a single success indicator.

Version 2.0 Syntax

```
<delete_business generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <businessKey/>
    [<businessKey/>...]
</delete_business>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

businessKey: one or more uuid_key values that represent specific instances of known businessEntity data.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: It signifies that one of the uuid_key values passed did not match with any known businessKey values. No partial results will be returned. If any businessKey values passed are not valid or if the message contained multiple instances of a uuid_key value, this error will be returned. The key that caused the error will be clearly indicated in the error text.

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

E_userMismatch: It signifies that one or more of the uuid_key values passed refers to data that is not controlled by the individual who is represented by the authentication token. The key causing the error will be clearly indicated in the error text.

delete_service

Description: The delete_service API call is used to remove one or more previously businessService elements from the UDDI registry and from its containing businessEntity parent.

Upon successful completion, a dispositionReport is returned with a single success indicator. If a business service being deleted is the target of a business service projection associated with another businessEntity, that reference relationship will be automatically eliminated as a result of this call. All contained bindingTemplate data will also be removed from the registry as a result of this call. Any references to bindingTemplates so removed will not be affected.

Version 2.0 Syntax

```
<delete_service generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <serviceKey/>
    [<serviceKey/> .]
</delete_service>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

serviceKey: One or more uuid_key values that represent specific instances of known businessService data.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: It signifies that one of the uuid_key values passed did not match with any known serviceKey values. No partial results will be returned. If any serviceKey values passed are not valid or if the message contained multiple instances of a uuid_key value, this error will be returned. The key causing the error will be clearly indicated in the error text.

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

E_userMismatch: It signifies that one or more of the uuid_key values passed refers to data that is not controlled by the individual who is represented by the authentication token. The key causing the error will be clearly indicated in the error text.

delete_binding

Description: The delete_binding API call causes one or more instances of bindingTemplate data to be deleted from the UDDI registry.

Upon successful completion, a dispositionReport is returned with a single success indicator. References to bindingTemplates that are deleted as a result of this call, such as those referenced by other bindingTemplates (in hostingRedirector elements) are not affected.

Version 2.0 Syntax

```
<delete_binding generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <bindingKey/> [<bindingKey/> .]
</delete_binding>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

bindingKey: One or more uuid_key values that represent specific instances of known bindingTemplate data.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: It signifies that one of the uuid_key values passed did not match with any known bindingKey values. No partial results will be returned - if any bindingKey values passed are not valid or if the message contained multiple instances of a uuid_key value, this error will be returned.

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

E_userMismatch: It signifies that one or more of the bindingKey values passed refers to a bindingTemplate that is not controlled by the individual associated with the authentication token.

delete_tModel

Description: The delete_tModel API call is used to logically delete one or more tModel structures. Logical deletion hides the deleted tModels from find_tModel result sets but does not physically delete it. Deleting an already deleted tModel has no effect.

Upon successful completion, a dispositionReport message is returned with a single success indicator.

Version 2.0 Syntax

```
<delete_tModel generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <tModelKey/> [<tModelKey/> .]
</delete_tModel>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

tModelKey: One or more URN qualified uuid_key values that represent specific instances of known tModel data. All tModelKey values begin with a uuid URN qualifier (e.g. "uuid:" followed by a known tModel UUID value.)

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: It signifies that one of the URN qualified uuid_key values passed did not match with any known tModelKey values.

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

E_userMismatch: It signifies that one or more of the tModelKey values passed refers to data that is not controlled by the individual who is represented by the authentication token.

get_registeredInfo

Description: The get_registeredInfo API call is used to get an abbreviated list of all businessEntity and tModel data that are controlled by the individual associated with the credentials passed.

Upon successful completion, a registeredInfo message is returned, listing abbreviated business information in one or more businessInfo elements, and tModel information in one or more tModelInfo elements. This API is useful for determining the full extent of registered business and tModel information controlled by a single user in a single call. This message complements the get_publisherAssertions message, which returns information about assertions managed by an individual publisher account.

Version 2.0 Syntax

```
<get_registeredInfo generic="2.0" xmlns="urn:uddi-org:api_v2" >
     <authInfo/>
</get_registeredInfo>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

Error Returned

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

set_publisherAssertions

Description: The set_publisherAssertions API call is used to manage all of the tracked relationship assertions associated with an individual publisher account.

Upon successful completion, a publisherAssertions message is returned containing all of the relationship assertions currently attributed to the publisher account that is associated with the authInfo data passed.

Version 2.0 Syntax

```
<set_publisherAssertions generic="2.0" xmlns="urn:uddi-org:api_v2" >
     <authInfo/>
     [<publisherAssertion>
     <fromKey/>
     <toKey/>
     <keyedReference/>
     </publisherAssertion>...]
</set_publisherAssertions>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

publisherAssertion: These are zero or more relationship assertions. Relationship assertions consist of a reference to two businessEntity key values as designated by the fromKey and toKey elements, as well as a required expression of directional relationship within the contained keyedReference element.

Error Returned

If any error occurs in processing this API call, a dispositionReport element is returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: It signifies that one of the uuid_key values passed did not match with any known businessKey or tModelKey values. The assertion element and the key that caused the problem will be clearly indicated in the error text.

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

E_userMismatchv: It signifies that neither of the businessKey values passed in the embedded fromKey and toKey elements is controlled by the publisher account associated with the authentication token. The error text clearly indicates which assertion caused the error.

add_publisherAssertions

Description: The add_publisherAssertions API call causes one or more publisherAssertions to be added to an individual publisher's assertion collection.

Upon successful completion, a dispositionReport message is returned with a single success indicator.

Version 2.0 Syntax

```
<add_publisherAssertions generic="2.0" xmlns="urn:uddi-org:api_v2" >
     <authInfo/>
     <publisherAssertion>
     <fromKey/>
     <toKey/>
     <keyedReference/>
     </publisherAssertion>
     [<publisherAssertion/>...]
</add_publisherAssertions>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

publisherAssertion: These are one or more relationship assertions. Relationship assertions consist of a reference to two businessEntity key values as designated by the fromKey and toKey elements, as well as a required expression of directional relationship within the contained keyedReference element.

Error Returned

If any error occurs in processing this API call, a dispositionReport element is returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed: It signifies that one of the uuid_key values passed did not match with any known businessKey or tModelKey values. The key and element or attribute that caused the problem will be clearly indicated in the error text.

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

E_userMismatch: It signifies that neither of the businessKey values passed in the embedded fromKey and toKey elements is controlled by the publisher account associated with the authentication token. The error text will clearly indicate which assertion caused the error.

delete_publisherAssertions

Description: The delete_publisherAssertions API call causes one or more publisherAssertion elements to be removed from a publisher's assertion collection.

Upon successful completion, a dispositionReport message is returned with a single success indicator.

Version 2.0 Syntax

```
<delete_publisherAssertions generic="2.0" xmlns="urn:uddi-org:api_v2" <
    <authInfo/<
    <publisherAssertion<
    <fromKey/<
    <toKey/<
    <keyedReference/<
    </publisherAssertion<
    [<publisherAssertion/<...]
</delete_publisherAssertions<
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

publisherAssertion: These are one or more publisher assertion structures exactly matching an existing assertion that can be found in the publisher's assertion collection.

Error Returned

If any error occurs in processing this API call, a dispositionReport element is returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_assertionNotFound: It signifies that one of the assertion structures passed does not have any corresponding match in the publisher's assertion collection. This also occurs if a publisher assertion appears multiple times in the message. The assertion that caused the problem will be clearly indicated in the error text.

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

get_assertionStatusReport

Description: The get_assertionStatusReport API call provides administrative support for determining the status of current and outstanding publisher assertions that involve any of the business registrations managed by the individual publisher account. Using this message, a publisher can see the status of assertions that they have made, as well as see assertions that others have made that involve businessEntity structures controlled by the calling publisher account.

Upon successful completion, an assertionStatusReport message is returned containing assertion status information.

Version 2.0 Syntax

```
<get_assertionStatusReport generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    [<completionStatus/>]
</get_assertionStatusReport>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

completionStatus: These are one of the following values.

- **status:complete**: Passing this value causes only the publisher assertions that are complete to be returned. Each businessEntity listed in assertions that are complete has a visible relationship that directly reflects the data in a complete assertion

- **status:toKey_incomplete**: Passing this value causes only those publisher assertions where the party who controls the businessEntity referenced by the toKey value in an assertion has not made a matching assertion to be listed.
- **status:fromKey_incomplete**: Passing this value causes only those publisher assertions where the party who controls the businessEntity referenced by the fromKey value in an assertion has not made a matching assertion to be listed.

Error Returned

If any error occurs in processing this API call, a dispositionReport element is returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidCompletionStatus: It signifies that the completionStatus value passed is unrecognized. The completion status that caused the problem will be clearly indicated in the error text.

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

get_publisherAssertions

Description: The get_publisherAssertions API call is used to obtain the full set of publisher assertions that is associated with an individual publisher account.

This API call returns a publisherAssertions message that contains a publisherAssertion element for each publisher assertion registered by the publisher account associated with the authentication information. Only assertions made by the individual publisher are returned.

Version 2.0 Syntax

```
<get_publisherAssertions generic="2.0" xmlns="urn:uddi-org:api_v2" >
     <authInfo/>
</get_publisherAssertions>
```

Arguments

authInfo: This required argument is an element that contains an authentication token. Authentication tokens are obtained using the get_authToken API call.

Error Returned

If any error occurs in processing this API call, a dispositionReport element is returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_authTokenExpired: It signifies that the authentication token value passed in the authInfo argument is no longer valid because the token has expired.

E_authTokenRequired: It signifies that the authentication token value passed in the authInfo argument is either missing or is not valid.

Error Code Reference

A complete reference of error codes returned by UDDI APIs is as given:

Error Codes

The following list of error codes can be returned in the error code and error number (errCode and errno attributes) within a dispositionReport response to the API calls.

If a V2 registry encounters an error while processing a V1 message it may only return a V1 message.

Non-error conditions are not reported by way of SOAP Faults but are instead reported using the dispositionReport element.

Error Key	Error Number	Description
E_assertionNotFound	30000	Signifies that a particular publisher assertion (consisting of two businessKey values, and a keyed reference with three components) cannot be identified in a save or delete operation.
E_authTokenExpired	10110	Signifies that the authentication token information has timed out.
E_authTokenRequired	10120	Signifies that an invalid authentication token was passed to an API call that requires authentication.
E_accountLimitExceeded	10160	Signifies that a save request exceeded the quantity limits for a given data type.
E_busy	10400	Signifies that the request cannot be processed at the current time.
E_categorizationNotAllowed	20100	Used for UDDI Version 1.0 compatibility. Replaced by E_valueNotAllowed in 2 and higher. Restrictions have been placed on

		the types of information that can be categorized within a specific taxonomy.
E_fatalError	10500	Signifies that a serious technical error has occurred while processing the request.
E_invalidKeyPassed	10210	Signifies that the uuid_key value passed did not match with any known key values. The details on the invalid key will be included in the dispositionReport element.
E_invalidProjection	20230	Signifies that an attempt was made to save a businessEntity containing a service projection that does not match the businessService being projected. The serviceKey of at least one such businessService will be included in the dispositionReport.
E_invalidCategory	20000	Used for UDDI Version 1.0 compatibility only. Replaced by E_invalidValue in version 2 and higher. Signifies that the given keyValue did not correspond to a category within the taxonomy identified by the tModelKey. Used with categorization only.
E_invalidCompletionStatus	30100	Signifies that one of the assertion status values passed is unrecognized. The completion status that caused the problem will be clearly indicated in the error text.
E_invalidURLPassed	10220	Signifies that an error occurred during processing of a save function involving accessing data from a remote URL. The details of the HTTP Get report will be included in the dispositionReport element. Not used in V1 or V2.
E_invalidValue	20200	A value that was passed in a keyValue attribute did not pass validation. This applies to checked categorizations, identifiers and other validated code lists. The error text will clearly indicate the key and value combination that failed validation.
E_keyRetired	10310	Signifies that a uuid_key value passed has been removed from the registry. While the key was once valid as an accessor,

		and is still possibly valid, the publisher has removed the information referenced by the uuid_key passed. V1 errata is not used. Included here for historical code-set completion.
E_languageError	10060	Signifies that an error was detected while processing elements that were annotated with xml:lang qualifiers. Presently, only the description and name elements support xml:lang qualifications.
E_messageTooLarge	30110	Signifies that the message is too large. The upper limit will be clearly indicated in the error text.
E_nameTooLong	10020	Used for UDDI Version 1.0 compatibility only. Signifies that the partial name value passed exceeds the maximum name length designated by the policy of an implementation or Operator Site.
E_operatorMismatch	10130	Signifies that an attempt was made to use the publishing API to change data that is mastered at another Operator Site. This error is only relevant to the public Operator Sites and does not apply to other UDDI compatible registries.
E_publisherCancelled	30220	The target publisher cancelled the custody transfer operation.
E_requestDenied	30210	A custody transfer request has been refused.
E_requestTimeout	20240	Signifies that the request could not be carried out because a needed web service, such as validate_values, did not respond in a reasonable amount of time. Details identifying the failing service will be included in the dispositionReport element.
E_secretUnknown	30230	The target publisher was unable to match the shared secret and the five (5) attempt limit was exhausted. The target operator automatically cancelled the transfer operation.
E_success	0	Signifies no failure occurred. This return code is used with the dispositionReport for

		reporting results from requests with no natural response document.
E_tooManyOptions	10030	Signifies that too many or incompatible arguments were passed. The error text will clearly indicate the nature of the problem.
E_transferAborted	30200	Signifies that a custody transfer request will not succeed.
E_unrecognizedVersion	10040	Signifies that the value of the generic attribute passed is unsupported by the Operator Instance being queried.
E_unknownUser	10150	Signifies that the user ID and password pair passed in a get_authToken message is not known to the Operator Site or is not valid.
E_unsupported	10050	Signifies that the implementer does not support a feature or API.
E_userMismatch	10140	Signifies that an attempt was made to use the publishing API to change data that is controlled by another party.
E_valueNotAllowed	20210	Signifies that a value did not pass validation because of contextual issues. The value may be valid in some contexts, but not in the context used. The error text may contain information about the contextual problem.
E_unvalidatable	20220	Signifies that an attempt was made to reference a taxonomy or identifier system in a keyedReference whose tModel is categorized with the unvalidatable categorization.