UDDI Version 3 Features List

This version:

http://uddi.org/pubs/uddi_v3_features.htm

Latest version:

http://uddi.org/pubs/uddi_v3_features.htm

Editors (alphabetically):

Karsten Januszewski, Microsoft Corporation

Ed Mooney, Sun Microsystems, Inc.

Contributors (alphabetically):

Richard Harrah, Hewlett-Packard Company

Sam Lee, Oracle Corporation

Joel Munter, Intel Corporation

Claus Von Riegen, SAP AG

Content

1 Introduction

UDDI Version 3 builds on the vision of UDDI: a "meta service" for locating web services by enabling robust queries against rich metadata. Expanding on the foundation of Versions 1 and 2, Version 3 offers the industry a specification for building flexible, interoperable XML Web services registries useful in private as well as public deployments.

With a vast array of enhancements – including multi-registry topologies, increased security features, improved WSDL support, a new subscription API and core information model advances – the Version 3 specification offers clients and implementers a comprehensive and complete blueprint of a description and discovery foundation for a diverse set of Web services architectures.

References to the UDDI Version 3.0 Specification have been provided at the end of each section.

2 Publisher Assigned Keys

Each entity in a UDDI registry is assigned a key, which uniquely identifies that entity within the registry. In prior versions of the UDDI Specification, the behavior in which a publisher wished to copy the entirety of a UDDI registry entity from one registry to another while preserving the identical key was explicitly not allowed.

Version 3 of UDDI approaches the issue of key generation in a significantly different fashion and, as such, the possibility of publishing an entity to another UDDI registry while preserving the key is allowed. This behavior is known as *entity promotion*. With this version of UDDI, a publisher is permitted to propose a new key for an entity, and, given the policies of a registry, that key and the entity associated with that key may be inserted into the registry.

The ability to share data among registries puts broadly distributed environments within UDDI's scope.

2.1 Support for Multi-Registry Environment

In order to support this more broadly distributed environment, UDDI Version 3 introduces the notions of *root* and *affiliate* registries as part of its guidance on inter-registry associations. The existence of a root registry enables its affiliates to share data with the root registry and among themselves with the knowledge that keys remain unique. The notion of registry topologies is thus enabled – multiple UDDI registries that are interrelated in complex ways.

For more information, see Chapter 8, *Publishing Across Multiple Registries*.

2.2 Human-friendly, URI-based keys

Alongside the ability to promote keys across registries is a new format for UDDI keys. Prior versions of UDDI mandated that keys had to be a formatted Universally Unique Identifier (UUID). Version 3 removes this restriction and recommends the usage of a key scheme based on DNS names. This allows publishers to establish a key partition from a DNS record and then generate keys based on that partition. For example, a valid Version 3 key might look as follows:

```
uddi:example.com:1
uddi:example.com:sales-division:53
```

These human-friendly keys allow organizations to manage their own key space using internally established conventions and structures.

For more information, see Section 4.4, *About uddiKeys* and Section 5.2.2, *Publishing Entities with Publisher Assigned Keys*.

2.3 The Universal Business Registry (UBR) as Root

In order to enable this new scenario of multiple UDDI registries sharing keys, the establishment of a root registry is critical. An important example of a root registry is the UDDI Business Registry (UBR), which today has a set of policies in place to generate unique keys as well as in the future will implement policies to validate key partitions through signatures that correlate with DNS records. These policies ensure the uniqueness of keys within the UDDI Business Registry, thus establishing the UBR as a root registry of many purposes.

By acknowledging the UBR as a root, an affiliate registry can establish inter-registry communication policies and procedures with both the UBR and any other registry which is an affiliate of the UBR.

For more information, see Chapter 8, *Publishing Across Multiple Registries*.

3 Support for digital signature

A major advancement in the Version 3 specification is the support for digital signatures. By allowing UDDI entities to be digitally signed, a new level of data integrity and authenticity is delivered by UDDI.

Inquirers of a registry can now filter their queries, only requesting data that has in fact been signed. When an inquirer then retrieves and verifies data from a registry, the inquirer can be confident that the data is *exactly* as the publisher intended it.

Publishers to a registry now have the assurance that they are not being misrepresented by someone claiming to own a UDDI entity. Once publishers have signed data, they can have confidence in the integrity of that data.

These assurances for both the inquirer and publisher are transitive: given V3's support of entity promotion, data can be copied between registries and guaranteed to not have changed during the process of being copied. As such, the multiple registry environment discussed above can be achieved with a high level of data integrity.

Ultimately, digital signatures improve the data quality within UDDI and allow both the protection and non-repudiation required for e-commerce and other Web services scenarios that call for a high degree of trust.

For more information, see Appendix I, *Support For Digital Signatures*.

4 Policy

The Version 3 specification acknowledges and enables UDDI to be employed in a variety of different milieux. Because of the diverse set of environments for different UDDI registry implementations – internet, extranet, intranet, development, test, production, etc. – there is a need to provide flexibility for implementations to support vastly different operational policies. With this in mind, significant work was undertaken to identify all the policy decisions that each UDDI registry and/or node must make. Using the policy guide that is now part of the Version 3 specification, different UDDI implementations can mold a particular registry given its context.

The Version 3 specification also outlines mechanisms for representing these policy decisions both through a new UDDI policy schema and through normative modeling behaviors within UDDI itself. As such, policies can be established and consistently determined by inquirers of a UDDI node.

Some UDDI aspects that have been identified as policy decisions include the following: authorization models, data custody and confidentiality, key generation, value set validation, subscription, user publication limits, and audit policy. The specification outlines the goal for each policy and provides

recommendations for each policy decision.

The work undertaken on policy represents a significant step forward for enabling UDDI registries to flourish in a wide variety of contexts.

For more information, see Chapter 9, *Policy*.

5 Information Model Improvements

In response to a number of different use cases, the UDDI Information Model has been enhanced with several improvements.

5.1 categoryBags on bindingTemplates

The ability to classify bindingTemplates with categoryBags now allows metadata to be attributed directly to the technical details of a Web service, enabling more granular searches to be performed on the specific technical metadata for a given service.

For more information, see Section 3.5, *bindingTemplate Structure*.

5.2 Operational information

In Version 3, the separation of the data published into a UDDI entity versus the metadata associated with that data is critical. This separation is particularly important with regard to the ability to calculate a digital signature.

With this separation in mind, Version 3 adds a new data structure, operationalInfo. The operationalInfo structure is used to convey the operational information for the UDDI core data structures (businessEntity, businessService, bindingTemplate and tModel). Such operational information includes the date and time that the data structure was created and modified, the identifier of the UDDI node at which the publish operation took place, and the identity of the publisher.

For more information, see Section 3.8, *operationalInfo Structure.*

5.3 Multiple Overview Documents

The ability to define multiple overviewDoc elements for tModels and the instanceDetails structure in bindingTemplates is supported. This provides the capability to specify the description of both Web services and Web service types in different formats.

For more information, see Section 3.5.2.5, *overviewDoc*.

5.4 Support for Complex Categorization

UDDI now provides the ability to use complex categorization in attributing UDDI entities. For example, geodetic typing can now be modeled through the use of geographical coordinates in order to specify where a specific business or service is physically located. Geographical latitudes and longitudes can be grouped together in a keyedReferenceGroup.

Also, UDDI now provides the ability to extend categorization systems through derivation. In such a way, the same value set might be reused in different contexts. For example, a category system used to classify the *service area* of a business might be re-used to classify the *physical location* that business.

For more information, see Section 3.3.2.14, *keyedReferenceGroup* and Appendix F, *Using Categorization*.

5.5 Information Model Extensibility

UDDI Version 3 allows publishers to use XML Schema's derivation mechanism to extend the information

model. It provides clear rules and guidelines regarding the treatment of extended data structures.

For more information, see Appendix H, *Extensibility*.

5.6 Schema Enhancements

UDDI Version 3 makes extensive uses of the XML Schema features to unambiguously define the syntactic requirements of various API sets and data structures, reducing the possibility of human misinterpretation of the specification. Use of XML Schema also allows any schema-aware software and tools to process and validate UDDI messages more easily.

The XML Schema features used include:

- `minLength`/`maxLength` for string data field limits
- `choice` for defining the conditions that at least one of the multiple elements must present
- `value="collapse"` of whiteSpace facet for whitespace handling
- `default` qualifier of optional attributes for defining the behavior when optional attributes are omitted
- `final="restriction"` constraints for formalizing the schema extensibility allowed

For more information, see Chapter 2, *UDDI Schemas*.

5.7 Internationalization and language support

As part of its aim of providing a registry for *universal* description, discovery and integration, the UDDI specification includes support for internationalization features. Enhancements to this support falls into two broad groups.

First, the specification provides increased support for multi-regional businesses, organizations, and other Web service providers to describe their operations across international or inter-region units.

Second, the specification provides increased support for internationalization of UDDI data and services such as use of multiple languages or multiple scripts of the same language; mechanisms to specify additional language-specific sorting order; and consistent search results independent of language of information being searched.

For more information, see Appendix D, *Internationalization*.

5.8 Update to Access Point behavior and Improved WSDL Support

The UDDI accessPoint has been updated to support a more extensible typing ability through the addition of the useType attribute. This now enables WSDL files to be more easily registered and consumed from UDDI. It also simplifies the indirection mechanisms (hostingRedirector) that were explained in prior versions.

For more information, see Section 3.5.2.1, *accessPoint* and Appendix B, *Using and Extending the useType Attribute*.

6 Extended Discovery Features

The UDDI Inquiry API has been significantly enhanced to support a range of additional use cases and scenarios.

6.1 Support for previous multi-step queries into single-step complex queries

UDDI now provides the ability to nest sub-queries within a single query, reducing the number of round trips a client must make to a UDDI registry. By allowing nested queries for tModels within queries for services, clients can narrow in on the types of service they are searching for much more efficiently.

For more information, see Section 5.1.9, *find_binding*, 5.1.10, *find_business*, and 5.1.12, *find_service*.

6.2 New and extensible find qualifiers and sort orders

UDDI has created new normative find qualifiers to improve the effectiveness of queries. Version 3 also recognizes that different registries might support different find qualifiers and sort orders for a given API. With this in mind, a given registry can create and model new find qualifiers and sort orders.

For more information, see Section 5.1.4, *Find Qualifiers* and Section 11.4, *findQualifier tModels*.

6.3 Extended Wildcard support

UDDI has standardized on SQL-like approximate matching and has expanded the different queries in which wildcards can be used. This increases the kinds of queries that can be issued to UDDI.

For more information, see Section 5.1.6, *About Wildcards* and Appendix G, *Wildcards*.

6.4 Management of large results sets

UDDI now provides a mechanism to enable paging through large result sets. This effectively allows the data to be "chunked" into multiple response messages from the server.

For more information, see Section 5.1.5, *use of listDescription*.

7 Subscription API

The new subscription API includes robust support for notification with synchronous and asynchronous alternatives. The subscription API set also provides for tracking registry activity and has been updated to support multi-registry environments. In such a way, users can establish a subscription based on a specific query or set of entities that the user is interested in. In the case of a query-based subscription, if the result set changes within a given time span, the user is notified. In the case of entity-based subscription, if the contents of one of those entities were to change, the user is notified.

Some of the use cases enabled by subscription include notification of new businesses or services that are registered; monitoring of existing businesses or services; obtaining registry data for use in a private registy; and obtaining data for use in a marketplace or portal registry.

For more information, see Section 5.5, *Subscription API Set* and Appendix C, *Supporting Subscribers.*

8 Registry Management Improvements

New and revised features in UDDI Version 3 have improved registry management. These include improvements to external validation and the replication protocol, and the addition of a public API for custody transfer.

8.1 Custody and Ownership Transfer API

In publishing to a node, a publisher grants custody of its data to the node. UDDI Version 3 defines a new API that helps manages this relationship. It enables two publishers and two nodes in a registry to cooperatively transfer custody and ownership of one or more existing businessEntity or tModel structures from one node to another and at the same time to transfer ownership of the entities from one publisher to another. The Version 3 custody transfer API uses the replication stream to complete the custody

transfer process.

For more information, see Section 5.4, *Custody and Ownership Transfer API Set.*

8.2 External Validation

Registries that support external validation now have the option of caching external value set values. This helps minimize the number of calls to external validation web services.

UDDI Version 3 gives registries two options for obtaining the set of valid values. One is to accumulate the valid values from successful calls to validate_values. The other is to obtain the set of valid values with a call to get_allValidValues, where supported by the validation service. The get_allValidValues API is new with UDDI Version 3.

For more information, see Section 5.6, *Value Set API Set* and Section 6.4, *Checked Value Set Validation*.

8.3 Replication

The XML Replication Schema for UDDI V3 has been improved. Some elements have been removed and some added, matching the descriptions below. A minor attribute/facet redundancy within the acknowledgement processing was corrected and several important element values now have their types derived directly from the uddi_v3.xsd schema.

Specific corrections were made to the Replication API Set, including: enhancements to and requiring a response from notify_ChangeRecordsAvailable; the addition of a new get_HighWaterMarks message; the addition of metadata (i.e., operationalInfo) to the replication stream; the additional of a new change record payload type to support generator keys; the elimination of the indeterminate changeRecordSetAssertion message; and last but certainly not least, the addition of digital signature information.

For more information, see Chapter 7, *Inter-node Operation*.

9 Document Makeover

Version 3 consolidates the UDDI specification into one document, the result of which is a more readable document, in which everything necessary to UDDI exists in a single document. A glossary has been added to improve overall terminology consistency. Lastly, the specification has been released as HTML on the Web.