Enabling Service-Oriented Architecture
[REV. DRAFT] UDDI Executive White Paper

THE STENCIL GROUP

The Stencil Group, Inc.
912 Cole Street, PMB 287
San Francisco, California 94117

www.stencilgroup.com

August 2004

STATUS OF THIS DOCUMENT

This is DRAFT 2 of an updated UDDI Executive White Paper. It updates the previous draft (of August 8, 2004) to reflect feedback from several readers on the UDDI committees.

Please note that this document is a work-in-progress and should not be distributed outside the UDDI member committees, except to solicit comments before publication.

REVISION HISTORY

Draft 2: August 31, 2004

- Clarified analogies with ORB, DNS, etc.
- Clarified discussion of UBR.
- Added explicit reference to "taxonomy" in discussion of registry role.
- Added explicit reference to "governance" in discussion of UDDI's role in service-oriented infrastructure.
- Expanded executive summary.
- Corrected various minor typos and diction choices.
- Added companion slide presentation.

Draft 1: August 6, 2004

- Initial version.

EXECUTIVE SUMMARY

Question: how explicitly do you want to counter criticisms that UDDI is an "ugly duckling" or "MIA" among web services standards? I have had conflicting input from committee members on this point. If we want it explicit, this is the place to do it.

The Universal Description, Discovery, and Integration (UDDI) protocol is a key member of the group of interrelated standards that comprise the web services stack. It defines a standard method for publishing and discovering the network-based software components of a service-oriented architecture (SOA).

The UDDI registry model is a central element of the service-oriented approach to software design. By enabling policy-based distribution and management of enterprise web services, a UDDI registry delivers significant business value. It helps ensure that the convenience of developers, the requirements of enterprise architects, and the underlying business policies are not in opposition; in fact, it brings all of these needs into closer alignment by increasing software flexibility, reuse, and control.

The current 3.0 specification represents a significant milestone in UDDI's evolution. It provides key capabilities for enterprise-level deployment and is a mature, well-supported standard. Its development is led by the OASIS consortium of enterprise software vendors and customers.

This paper discusses the strategic rationale for UDDI and analyzes its enabling role in the context of today's enterprise web services applications. In a companion white paper, we provide a concise, technical overview of the UDDI standard and describe key architectural changes in the recent Version 3 specification.

THE SERVICE-ORIENTED IMPERATIVE

The success of IT organizations increasingly is measured by how well the systems they manage adapt to business change. IT leaders must identify and plan for an architecture that provides not only scalability and "five nines" reliability, but also the ability to add new application components or to reorient and coordinate existing functions nimbly and rapidly.

Yet, most incumbent enterprise applications were not designed for flexibility and with presumption of rapid change. The layering of several generations of computing technology has resulted in a complex infrastructure that is difficult to integrate and that often limits IT's options. It is no wonder that the financial and opportunity costs of integration-related projects are so high.

To address these challenges, IT leaders increasingly recognize they must begin to think of IT systems in terms of malleable services, not static assets. Although this service-oriented approach to enterprise software design represents a much-needed solution to the complexity and costs of many legacy practices, it also reflects several well-tested antecedents. Indeed, progressive enterprise software architects have long advocated methods in which applications are designed with modular, loosely coupled interfaces that hide the complexity of the underlying systems.

Yet, because of a lack of universal standards, many earlier approaches were not practical for solving broad-based enterprise software needs. This obstacle was particularly difficult in environments where all endpoint components could not be controlled fully, such as when business processes crossed organizational boundaries to include other corporate divisions or external trading partners.

It is in this context that web services—a group of interrelated standards based upon the Extensible Markup Language (XML) that define an open, loosely coupled, and simplified framework for integrating enterprise software applications—has been embraced by software vendors and customers alike. Indeed, over the past several years, the basic vocabulary of web services has become familiar to every IT executive. The model's pragmatic business benefits—better integration and coordination among systems, increased flexibility of IT assets, and reduced development costs—

are compelling.

A STANDARDS-BASED WEB SERVICES REGISTRY

UDDI is an important enabling element of the service-oriented approach to software design. The standard specifies protocols for accessing a registry for web services, methods for controlling access to the registry, and a mechanism for distributing or delegating records to other registries. In short, a UDDI registry provides a standards-based approach to locate a software service, to invoke that service, and to manage metadata about that service.

Rather than forcing applications to include hard-wired information about an external service's application programming interface (API), UDDI registries provide this binding information dynamically, at run-time. The benefits of this approach immediately become clear should some details—even the location of the service—change. Moreover, the UDDI registry can provide different responses depending upon the security, transport, or quality of service as defined by arbitrary business rules (or, as they often are described in the UDDI documentation, "taxonomies.")

To further illustrate the concept of a UDDI web services registry, consider the important roles similar systems have played in other distributed application architectures. The Domain Name System (DNS) controls the Internet's network addresses, CORBA implemented its Trading and Naming services to help direct the flow of system calls, and Microsoft Windows uses an eponymous Windows Registry to manage the interactions of COM/DCOM components. Although UDDI goes beyond the simple location transparency of most of these systems by providing an advanced framework for defining and querying services by taxonomy, the essential value of a UDDI registry is similar: it provides a mechanism for managing an otherwise ad hoc, chaotic, and unscalable series of interactions.

In fact, many adopters of web services methods today are facing this very challenge. Many software developers within IT organizations have begun to take advantage of a wide range of tools that simplify incorporating the SOAP interfaces that are the basic building blocks of web service interactions. While development managers and enterprise software architects often encourage this "organic" growth of web service implementations, they also are acutely aware of their enterprises' needs to provide an infrastructure that systematically addresses needs such as discovery, manageability, and security.

UDDI and other standards that have begun to flesh out the web services "stack" ensure that the convenience of developers, the requirements of enterprise architects, and the underlying business policies are not in opposition. In fact, a UDDI registry can make the jobs of all these groups of users significantly easier.

UDDI's Role in Web Services Development

Benefits such as standards-based interoperability that are provided to programmers by web services are clear. Nonetheless, when development teams begin to build web services interfaces into the their applications, they soon face issues all too familiar to developers who work in any programming environment: code reuse, ongoing maintenance, and documentation. Moreover, as the number of web services created within an IT organization grows, the need to manage these services can increase exponentially.

For development teams, registries based upon UDDI help answer needs such as:

- How can development managers systematically organize and manage web services across multiple systems and development teams?
- How can developers systematically manage the process of moving services through each phase of development: from coding to testing to public deployment?
- How can programmers document interface specifications, message transports, and authentication mechanisms with other developer groups? As the services change over time, how can external applications accommodate the changes?

When developing and deploying web services applications, UDDI registries help drive better code reuse and developer productivity. A UDDI registry provides an interoperable, standards-based approach for systematically documenting and publishing web services, regardless of development environment or platform. It can help developers—even across functional groups—to find a shared service and use that service within his or her own application.

UDDI's Role in Service-Oriented Infrastructure

Issues of services development point to the larger question of how to design an IT infrastructure that is supports web services development efforts. Although questions such as how best to conceive shared services, to design identity management and authentication mechanisms, and so on are beyond the scope of the UDDI standard (and this paper!), UDDI registries represent an important element of this overall question. In a service-oriented environment, enterprise software architects must consider questions such as:

- How can critical applications be insulated from changes—or failures—in back-end shared services?
- How can an organization share information about services in a controlled way that reflects the business rules and policies of the business?

Compounding these issues, a service-oriented approach implies that these questions must be addressed as a routine aspect of run-time operations, not hard-coded into the applications themselves.

Registries based upon UDDI provide IT administrators a formal layer of indirection necessary for service-oriented application development and management. By providing a sort of firewall between a service and the applications that call it, system administrators more easily can accommodate changes in the life-cycle of specific components—such as for version updates, for policy considerations, or even for service termination.

In addition to the fundamental benefits of run-time binding provided by the registry in a service-oriented architecture, administrators often require control of the publication and distribution of information about deployed web services, so that software deployment follows business policy. To facilitate these operational and governance needs, the current version of UDDI adds support for features such as client authentication and publish/subscribe for peer registries.

THE EVOLUTION OF UDDI

When UDDI first was conceived, much of the attention was focused on the "UDDI Business Registry" (UBR), a public implementation of the UDDI standard that represents a master directory of publicly available e-commerce services. Returning to an analogy used earlier in this article, one could consider the role of this public registry as similar to the root node of the DNS database. Although the comparison is imperfect, both represent successful examples of distributed registry infrastructure.

The UBR serves as highly visible reference implementation of the standard. As such, it provides an ideal environment for validating and proposing changes to the specification, as well as for testing applications and tools that make use of UDDI. Although the UBR remains an important part of the UDDI project, it represents only one aspect of the overall effort. Just as the overwhelming majority of DNS activity occurs within the confines of a company's own network, so too do most UDDI implementations support a business' internal web services infrastructure.

This understanding of how web services are most often used today is reflected as the UDDI specification has evolved. Its current implementation recognizes the need for federated control in real-world operational environments and further integrates the standard with other elements of service-oriented infrastructure. Highlights of the standard's progress are shown in the table below.

Figure 1: History of the UDDI Specification

UDDI VERSION	YEAR RELEASED	KEY OBJECTIVE
1.0	2000	Create foundation for registry of Internet-based business services
2.0	2003	Align specification with emerging web services standards and provide support for flexible, external taxonomies
3.0	2004	Support secure interaction of private and public implementations as key element of service-oriented infrastructure

The current 3.0 specification represents a significant milestone in UDDI's evolution. It provides key capabilities for enterprise-level deployment and is a mature, well-supported standard. In fact, a prerequisite of its certification by the OASIS standards group was the existence of several deployed commercial implementations.

HOW TO LEARN MORE

The UDDI specification is managed by OASIS, a member-led, international, non-profit standards consortium that concentrates on structured information and e-business standards. The organization's members include enterprise IT users, vendors, academics, and governments. In addition to UDDI, OASIS is known best for shepherding web services-related protocols such as ebXML, SAML, WS-Security, BPEL, and others.

To learn more about UDDI and OASIS, please visit *www.uddi.org*. In addition to the specification itself, the UDDI web site provides detailed technical notes, best practices, case studies, and information about how to contribute to UDDI's ongoing development. The site also provides links to several commercial and open-source implementations of UDDI registries that are available in the marketplace.

APPENDIX: UDDI USE CASE SCENARIOS

Many practical applications of general web services and specific UDDI concepts exist, and it is not our objective to document them exhaustively here. Instead, we outline two scenarios that are representative of the registry interaction features enabled by Version 3 of the UDDI specification.

Scenario 1: Private Test Registry

Business Scenario

For the past year, the IT organization of a major corporation had begun to explore the possibilities of web services approaches to application development and integration. Using the technology first in pilot projects and other piecemeal efforts, the IT team had skirted around the question of how to deploy and manage its web services applications. As it begins to plan for using web services in the company's mission-critical business processes and to create services that will be available to the rest of the organization, the IT team realizes that it will require a more controlled and systematic approach.

Overview of Issues

- *Need to test real-world conditions.* As software is developed, testing and debugging must occur under conditions as close to real-world production environment as possible and, in fact, incorporate several external, functioning services in the test scenarios. Additionally, it is desirable that as few modifications as possible be made to the component software to switch from "test" to "production" mode.

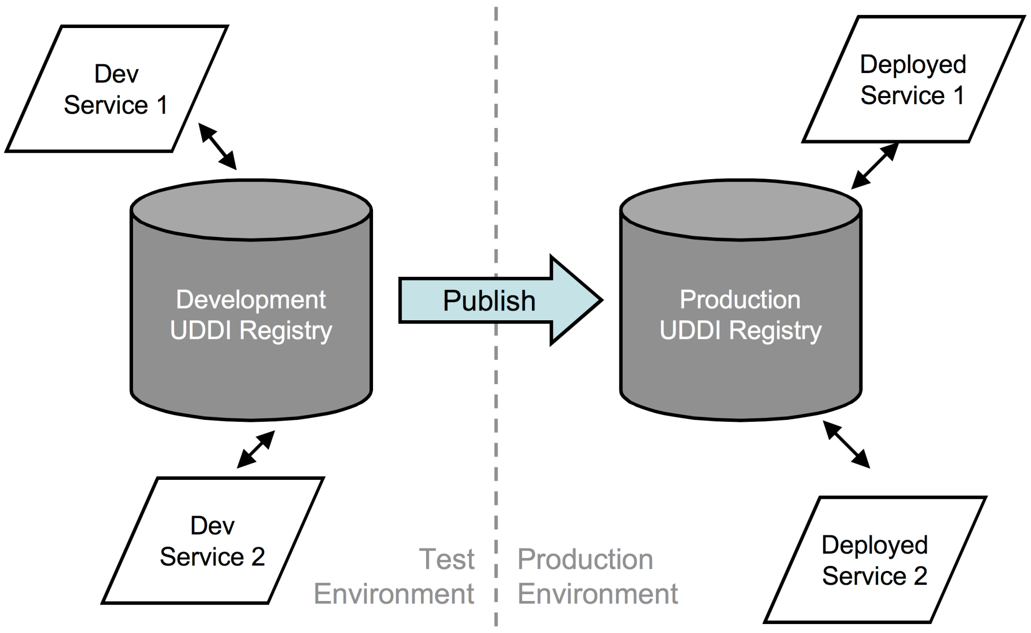
- *Clear separation between production and test systems.* At the same time, development versions of software must not interfere with actual production systems. Because services can be highly distributed and are loosely coupled, maintaining this distinction is paramount to ensure that dependencies are managed systematically.

- *Requirement to support distributed developer base.* Developers using the system may be based world-wide and, in fact, use different platforms and technologies from group to group. As a result, interoperability and support for a variety of network connections is an important functional requirement.

Description of Solution

The IT organization develops a test environment that utilizes a "one-way peering" model of registry interaction to create two overlapping domains for services. Those in the "private" domain can interact with the outside world, but not the other way around. When development versions of software have been fully tested and certified, they are promoted to the production sphere, using the expanded publishing features of the UDDI Version 3 specification.

Figure 2: Illustration of Private Test Registry



Comment: As services are certified and promoted to the production environment, the associated UDDI entities are published from the development registry to the production registry using new features enabled in Version 3 of the UDDI specification.

Scenario 2: Supporting Collaboration among Trading Partners

Business Scenario

A large manufacturer has built a business based on providing "specialty" and custom fabricated plastics components on a spot and contract basis. Its role in the middle of the supply chain—between commodity suppliers like refiners and the plants of manufacturers like consumer packaged goods concerns—requires that the company manage relationships with multiple business partners and even act as an intermediary between its suppliers and customers. In order to increase its value to partners by providing visibility into supply and demand, as well as reduce its own costs of managing inventory and logistics, the company has embarked upon a program of automating a largely manual process of communicating with its suppliers using web services-based interfaces to the key applications.

Overview of Issues

- *Interoperability*. The sources of data for the new system range from internal systems like ERP applications to third-party services like inventory and logistics tracking. Because all of these applications are established, long-running systems, standardizing on one particular platform is not an option.

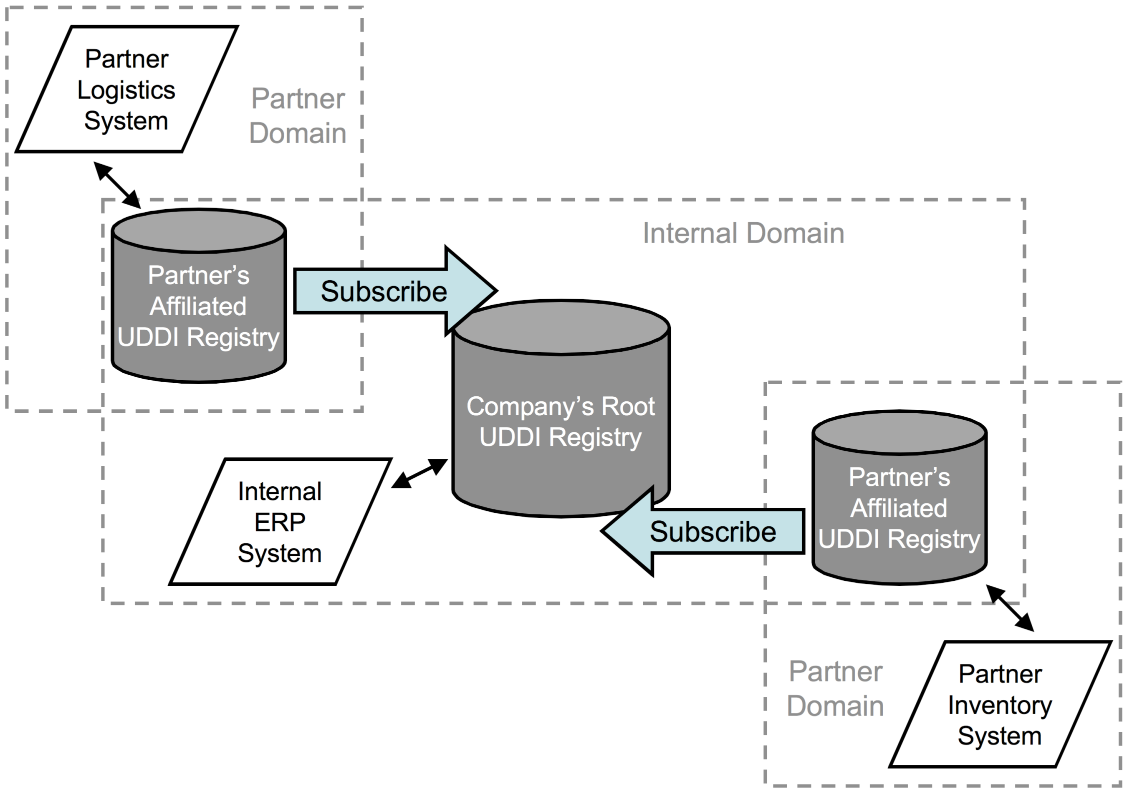
- *Decentralization and collaboration*. The company's business relationships are highly customized, and as a result, the integration infrastructure must be significantly decentralized. In fact, many of the business processes in question cannot be controlled by any single organization but, rather, require the cooperation of all parties involved.

- *Security*. Many of the systems in question are highly strategic, and information about these systems—even where they exist—may be highly sensitive and should not be shared with other companies in the network.

Description of Solution

As part of an overall web services solution, the company implements a service broker using a UDDI registry as a central element. By deploying it within the boundaries of a "DMZ" trusted environment, the company can both isolate interactions from its internal network, as well as limit the exposure of the registry to the outside world. In addition, by establishing subscription-based relationships with partners' registries in the trading network, the company can ensure that information is fully, but safely, distributed among trading partners. The registry also implements the XML Digital Signatures support in Version 3 of the UDDI specification to ensure the integrity and authenticity of exchanged data.

Figure 3: Illustration of Trading Partner Collaboration



Comment: Partners use UDDI Version 3's new subscription features to monitor the company's root registry. They gain visibility to only a desired subset of all of the services available, as defined in the company's business policies.