

serving up the proper balance of web services

CREATE, DEPLOY AND MAINTAIN YOUR LEGACY APPS AND DATA WITHOUT WRITING A SINGLE LINE OF CODE USING THE WEB SERVICES COMPONENTS INCLUDED IN NETWARE 6.5

**BY PETER CLEGG
PHOTOGRAPH BY ERIC TUCKER**



There are web services and then there are Web Services. If the proliferation of Internet idioms has you hazily categorizing Web services as...well, as “just a service that runs across the Web” then you need to read on. Web services is an ancient concept (at least in computer years) that is forcibly coming of age with the convergence of several technologies. Web services are evolving as the holy grail that allows disparate systems in geographically distributed networks to discover, interact and communicate integrated information and services—all without human interaction. And, the bonanza for Novell environments is that NetWare 6.5 includes everything required to create, deploy and host Web services using Novell exteNd technologies. This article provides an overview of Web services, the exteNd product features, and a brief look at how Web services can be applied to create cross platform services, integrate legacy systems or aggregate dissimilar applications. (For more information on Novell Web services, read the white paper at: www.novell.com/products/netware/whitepapers.html)

WHAT ARE WEB SERVICES?

Web services is the emerging category of Web applications that are self-contained, self-describing, modular and can be published, located and invoked across the Web. A Web service can be anything from a simple request for data to a complicated business analysis process. The concept of Web services isn't new—it's just that with the advent of the Internet and the evolution of a few standards, they are now easily possible.

Think of a Web service as that number you call occasionally to get the correct time. Somewhere there's an atomic clock linked to a voice processor that when the phone rings, tells you the time. The spoken time is the “service.” In order to receive the service, you needed to use a communications infrastructure—the phone system. It may be that you had to find the time service through a directory. And, you may also need to get the time in a language other than English—say Chinese or Russian. With the communication infrastructure, a directory and translations, the same service could be used all over the world. Add to the time service other voice

services, such as bank balances, stock quotes and telephone menu processes and you start to see how the number and usefulness of applications increases.

In the world of Web services, the communications infrastructure is the Internet. Using Hypertext Transport Protocol (HTTP or HTTPS) as a Web communications protocol, any service can access any other service across the Web. Each service can publish information about itself, its interface and integration requirements through a standardized format, Web services Description Language (WSDL). Service information can be published to directories using Universal Directory and Discovery Integration (UDDI), a standardized directory format. And all services can communicate using a common language, eXtensible Markup Language (XML). In addition, there may be instances where one service uses another service as a source of information. For service-to-service or process-to-process communications there is Simple Object Access Protocol (SOAP), an XML-based message protocol. (For more information, see *Web Services Components* on page 25.)

The real power of Web services comes with process-to-process aggregation and integration. Because Web services are based on open standards, there are no dependencies on platform, application vendor, access device type, or location. Any system can interact with any other system using a common infrastructure and translations. The results of these interactions can be aggregated for immediate use, or processed and repackaged as another published service. Here are a couple of common Web services examples:

Your bank has information about your savings and checking accounts. Other financial institutions such as mortgage companies, stock brokerages, life insurance companies, credit unions, credit card companies and financial managers have other financial information that is valuable to you. Wouldn't it be nice to see it all in one comprehensive view? Each of these companies no doubt has dissimilar computing systems, but using Web services, each institution could securely provide this information for your aggregated access. Using XML, you can create a service or application to provide individual information. The details of how this service can be accessed or consumed is exposed through

You can use Web services to integrate supply chains, provide customers with comprehensive views of their accounts, give employees the ability to self-service benefits, or create any type of business-to-business integration.

WSDL. You publish the availability of the service using UDDI. Communication takes place over HTTP. And the actual binding or secure interaction between the consumer and the service is accomplished through SOAP. (For more information, see *Web Services Components* on page 25.) An enterprising bank could use Web services to provide customers with a single comprehensive view of all their financial information. Another Web service could be created to analyze this information providing asset and liability statements or total net worth calculations. The collection of information and interaction of services is completely automated and is dynamically current each time you log in.

A second, more common example is what you see when you visit a retailer's Web site. Online successes like Overstock.com have worked with suppliers to automate and integrate thousands of products from hundreds of locations to appear as if available from a single location. Suppliers publish refurbished, seconds or overstock inventory items with associated content and images through XML. Overstock.com uses Web services to pull from dissimilar systems and sources, aggregating and formatting content to create an online warehouse of overstocked and discontinued items. Suppliers may be running Windows, Unix, mainframes or even Macs but as long as they have an Internet connection and the ability to support open standards, the exchange is seamless and transparent. Everything shows up on the Overstock.com catalog page as if it were available from a single supplier.

You can use Web services to integrate supply chains, provide customers with comprehensive views of their accounts, give employees the ability to self-service benefits, or create any type of business-to-business integration. The primary benefit of Web services is that connectivity and interaction are negotiated in real time, without the need for human interaction. Applications can be provided as a service without the requirements for download, installation or platform restrictions.

WEB SERVICES WITH NETWARE 6.5

With NetWare 6.5, you've got Web services—great Web services capabilities. In July 2002, Novell acquired SilverStream Software, a

leader in Web services-oriented application development. Novell now offers Novell exteNd solutions for any platform and has also integrated and optimized four Web services solutions for NetWare. NetWare 6.5 includes a state-of-the-art J2EE Web services application server (Novell exteNd Application Server 5.0) and an intuitive integrated development environment (IDE) in which to develop and deploy Web services (Novell exteNd Workbench 4.1.1) at no charge. Novell exteNd Director and Novell exteNd Composer are available separately. Here's a summary list of the exteNd technologies and features:

The Novell exteNd Workbench includes enhanced J2EE component and Web service-creation wizards, visual designers, archive-based projects and one-button deployment to J2EE application servers. Novell exteNd Workbench is a J2EE-oriented IDE that providers can use to create, deploy and maintain Web services. Workbench includes the following tools, wizards and capabilities:

- Component wizards: Create J2EE components such as JSP pages, EJBs, servlets, Java classes, JavaBeans and tag libraries.
- Web service facilities: Use the included wizard for creating Java-based Web service components, a SOAP runtime environment, and a registry manager for searching and publishing to Web service registries.
- Graphical and text-based editors: Edit Java files, JSP files, XML files, WSDL documents, deployment descriptors and plain text files. Editor formats include open source NetBeans or native Workbench (Deployment Description Editor, Deployment Plan Editor, WSDL). Image and class viewers are also included.
- Project views: Show the structure of a project's source files and the structure of a project's generated archives.
- Project tools: Assist in building projects, creating and validating J2EE archives, and deploying archives to J2EE application servers.
- Deployment tools: Deploy applications to all leading J2EE application servers with one click. Hot-deployment automatically and immediately deploys to IBM WebSphere, BEA WebLogic, Oracle or Novell exteNd Application Server.

The primary benefit of Web services is that connectivity and interaction are negotiated in real time without the need for human interaction. Applications can be provided as a service without the requirements for download, installation or platform restrictions.

- Version control integration: Access third-party version control systems directly from Workbench.
- Migration wizard: Automatically update deployment descriptors and other definitions from J2EE 1.2 to J2EE 1.3.
- UDDI tools: Use the included test-bed UDDI server, a UDDI browser and WSDL editor.
- jBroker Web: Core technologies for exteNd Web service support including compilers and SOAP runtime based on JAX-RPC.
- Web service wizard: Invoke jBroker Web compilers to generate Java classes and WSDL files, plus convert Java to WSDL and vice versa.
- Debugger: Invoke the exteNd Debugger directly from Workbench to debug server-based applications. The test-bed client allows running of sample code for accuracy.
- Registry Manager: Define profiles, search registries, view business and service information, and publish new services to a registry.
- Integration with other IDEs: Use best-of-breed J2EE development tools such as WebGain Visual Café, Borland Jbuilder, InLine Standard Edition, and Macromedia Dreamweaver or Novell exteNd Designer in conjunction with exteNd Workbench.

The Novell exteNd Application Server is a comprehensive, J2EE-certified platform for building and deploying enterprise-class Web applications. It supports the full Java 2 Enterprise Edition (J2EE) standard including:

- JavaServer Pages (JSP)
- Enterprise JavaBeans (EJB)
- standards-based programming interfaces for databases (Java Database Connectivity—JDBC)
- directories (Java Naming and Directory Interface—JNDI)
- messaging (Java Messaging Service—JMS)
- transactions (Java Transaction API—JTA)
- authorization (Java Authentication and Authorization Service—JAAS)
- Java Messaging Service (JMS)
- XML parsing (Java API for XML Parsing—JAXP)
- other services such as CORBA and JavaMail

Other features of Novell exteNd Application Server include:

- jBroker MQ: A full implementation of Java Message Service (JMS) 1.0.2 with features for point-to-point messaging and publish-subscribe communications.
- jBroker ORB: This provides CORBA 2.3 services as well as the RMI-IIOP protocol. Its features include forward (IDL to Java) and reverse (JavaRMI to IIOP) compilers, Portable Object Adapter (POA), Java Objects by Value, Server Activation, IIOP Connection Concentrator, Pluggable Authentication support, IIOP/SSL, Multicast invocations, COS Name Service and Object Transaction Service pluggability.
- Enterprise JavaBeans (EJB 1.1): A full-featured EJB server with support for session beans and complex container-managed entity beans. EJBs enable deployment of object-oriented, distributed enterprise-class applications.
- Servlets 2.2: Enables server-based dynamic HTML. Servlets are instantiated once and reused with caching for better performance. It supports WARs both as a packaging mechanism and as an application context at runtime.
- JavaServer Pages (JSP 1.1): These are also for dynamic HTML through the use of embedded Java tags. It includes a JSP-to-servlet compiler for faster compilation and error reporting.
- Other J2EE Platform Services:
 - JNDI 1.2 (Java Naming and Directory Interface): standardizes access to a variety of naming and directory services.
 - JDBC 2.0 (Java Database Connectivity): provides access to relational databases and other data repositories.
 - JavaMail 1.1: provides the ability to send and receive e-mail messages to the server.
 - JTA 1.0 (Java Transaction API): provides a way for J2EE components and clients to manage their own transactions, and for multiple components to participate in a single transaction.
 - XML (eXtensible Markup Language): provides data definitions as well as messaging and communication hierarchies.
 - Enterprise Data Connectors: These enable connectivity to non-relational databases and packaged applications such as SAP,

PeopleSoft and Lotus Notes (connectors are available with Novell exteNd Composer).

- Data Source Object: This enables automatic data binding where client-side, data-aware controls such as text boxes, list boxes and drop downs can be bound to columns of a Data Source Object without writing code. It also interacts with transaction processing monitors and servers such as CICS, Tuxedo, Microsoft Transaction Server, etc.

Novell exteNd Composer provides a drag-and-drop environment for developing Web services that extract resources from a broad range of legacy systems. Whether from mainframes, commercial systems, Web applications or anything in between, data is easily transformed into XML components. Composer enables organizations to Web service-enable mainframes, AS/400s, legacy VAX/VMS, packaged vendor applications including SAP and Peoplesoft, UNIX applications, EDI transactions, SQL databases and others.

exteNd Workbench provides a rich graphical interface for Web service creation, editing and deployment.

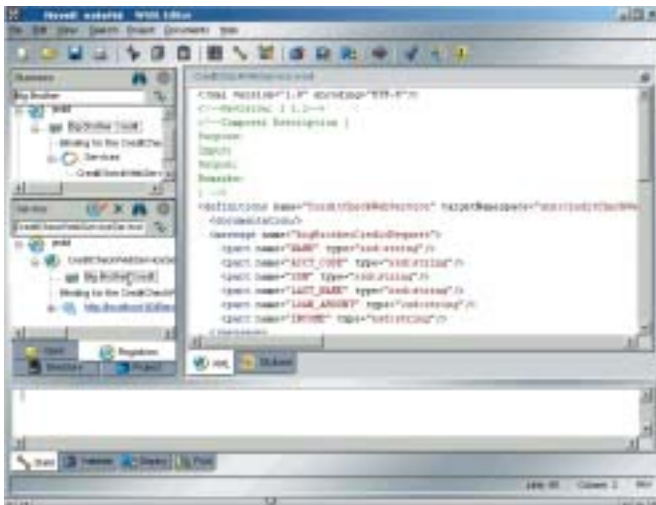


FIGURE 1

Composer also includes a robust business process modeling (BPM) tool that can be used to graphically design complex business-to-business processes. The BPM consists of a visual process designer, a modeling engine, a process server and an easy-to-understand process administrator. End-to-end Animation™ of business processes enables rapid development and troubleshooting during production. The Composer process modeling tool includes a powerful workflow engine based on IBM's Web services Flow Language (WSFL).

Novell exteNd Director is an interaction and portal server that assists developers and users in rapidly building interactive applications based on user need and device type. Director is a framework with programmable sub-systems that enables developers to quickly build applications using intuitive tools and interfaces. Its primary function is to facilitate the integration of disparate, third-party systems using a collection of pre-built subsystems. Director subsystems include portal, wireless device transcoding, workflow, rules engine, content management, user management, security, directory authentication, search and a common framework for caching, localization and utilities. Every subsystem is plug-and-play with the modular, J2EE-compatible architecture of exteNd Director. The open architecture provides developers with the flexibility to build solutions with diverse technologies such as portal, JSP (including Struts), Java clients, or even Microsoft .NET using Web services. Director includes a set of business drivers that provide user customization, personalization, profiling and rules to quickly and easily build sophisticated J2EE and Web services-based applications.

CREATING A WEB SERVICES APPLICATION

Creating a published, deployed, interactive Web Service can be nontrivial depending on the complexity of the project. For simple illustration purposes, we'll step through a few of the basics. For in-depth tutorials on developing Web services and Web applications using Novell exteNd Workbench, visit:

www.novell.com/documentation/lg/workbench41.

The Workbench interface is graphically rich and intuitive. You can

create projects, edit code and deploy services using tools that are straightforward and visually rich with information. Different panes provide navigation, editing and output options. Workbench uses the Model-View-Controller (MVC) architecture which separates user interface from business logic and manages the application flow with a controller servlet.

PROVIDERS AND CONSUMERS - In most cases, there are two components to every Web service—a provider and a consumer. You create providers and consumers in much the same way distinguishing them by the direction of operations between them. These two

components are also referred to as servers (providers) and clients (consumers). The steps to create Web services servers and clients are nearly the same and both rely on registries.

REGISTRY - Web services rely on registries or registry information which describes what a service does, where it is located, who provides it, how to communicate with it and what variables or parameters are required. Servers use registries to publish information about themselves. Clients use registries to locate servers and interpret how interaction will be established. Registries are often published to Web pages that are available for query. A registry can be created using Workbench's WSDL editor or can be located and queried using Workbench.

CREATING A WEB SERVICE - You can create Web services—server or client—using any one of several options (Enterprise Archive, JavaBean Archive, Resource Adapter Archive, Web Archive, Java Archive, etc.). With Workbench, you create a Web service first by creating a directory for service components and then creating a Web Archive (WAR) project.

After you create the service project, add source code for the

Workbench is used to locate and examine registry information.

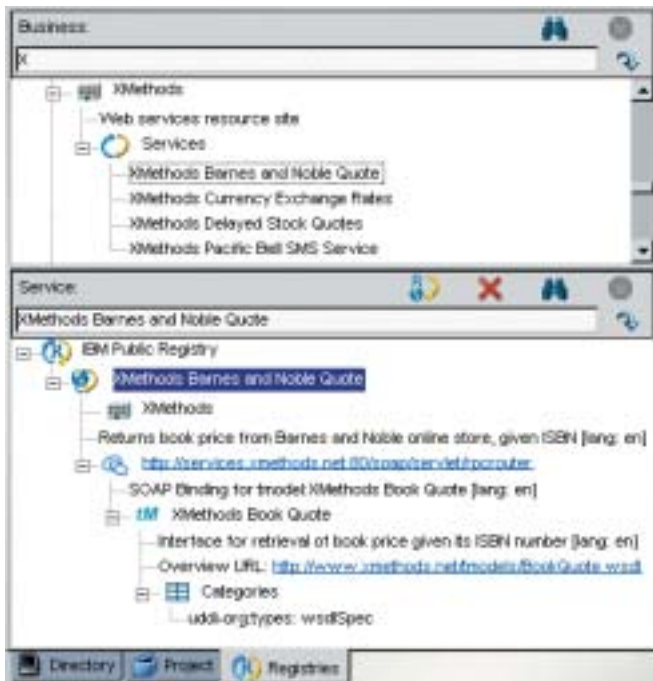


FIGURE 2

Sample WSDL code showing registry name, location and binding information.

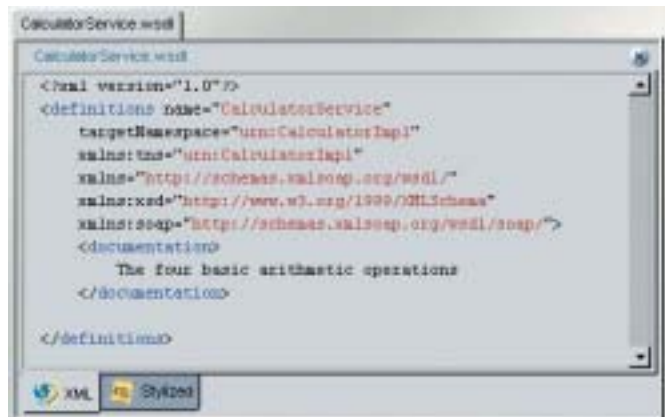


FIGURE 3

Workbench includes powerful deployment capabilities making it possible to deploy Web services to any of the leading application servers. Customize your deployment profiles with server-specific information and save them for future use.

particular service and any related libraries. You can easily make these additions using the graphical tools included with exteNd Workbench and the Web Service Wizard.

Web service providers use skeleton classes to implement remote interfaces. The skeleton receives a SOAP request, translates arguments from XML to Java data types, and calls the business method. The client application uses a stub class that also implements the remote interface. Workbench automatically generates the code for the communication between the Web service and the client application.

Once the code and libraries have been collected, the exteNd Workbench Wizard builds and compiles the project files which include generated code that enables the server to translate XML SOAP requests into method calls for the source object.

Now your project is ready for deployment or placement into operation on an application server. Workbench includes powerful

deployment capabilities making it possible to deploy Web services to any of the leading application servers. Customize your deployment profiles with server-specific information and save them for future use. Servers supported for deployment using Novell exteNd Workbench include: Novell exteNd Application Server, Sun Reference Implementation, Jakarta Tomcat, BEA WebLogic, IBM WebSphere, Oracle9iAS and SilverStream eXtend Application Server.

For Example:

Imperial Sugar Company, one of the largest sugar refiners and processors in the United States, needed to enhance customer service and reduce order processing costs in order to be competitive in an industry with razor-thin margins.

Using Novell exteNd technology, Imperial Sugar XML-enabled sales transactions from a legacy system, assembled them into appropriate business process flows, and exposed them as Web services in an advanced Web services-based portal that allows its customers to place orders electronically and view a real-time picture of their relationship with the company. Novell exteNd's intuitive, visual design environment requires very little training to create services from mainframe applications. Imperial Sugar completed the design and implementation of its Web services architecture in six weeks, followed by two months of testing after which it began roll out to its 10,000 customers. Imperial Sugar uses the Novell exteNd Application Server to manage all of the runtime execution for both its Web services components and the new Imperial extranet portal application. The load balancing features of the exteNd Application Server deliver the scalability and performance necessary for Imperial Sugar's phased rollout of the extranet, and its fault tolerance assures high availability to its customers.

A Web Archive (WAR) project is created for a new provider service.



FIGURE 4

Whether you're looking for a simple way to expose legacy content for broader reuse, or develop a new gee-whiz distributed application, Web services gives you a fast and easy way to do so.

THE ADVANTAGE OF WEB SERVICES

Web services provide two major benefits—flexibility in development and the reuse of existing content. With Web services, the plumbing (network infrastructure, platforms, applications, etc.) is separated from the content and business logic. This simplifies activities such as deploying, integrating, modifying and redeploying making it possible to reuse and repurpose existing applications and content. For many companies, the lifeblood data necessary to sustain business viability already exists in locations throughout the organization. Information may be locked in legacy mainframe and midrange systems, or vendor-packaged applications such as customer resource management (CRM), enterprise resource planning (ERP), human resources (HR) and others. Web services integration exposes the business intelligence contained in these specialized applications for broader application and reuse—without the need to implement new systems or replace existing assets. You can extract, manipulate, reformat, combine and present content to a much larger universe of

people, systems and processes using the same network infrastructure, same personnel and same systems that are already in place.

In addition, applications developed using Web services technology and according to Web services standards are portable across a wide assortment of platforms. There is broad flexibility in assembling component building blocks of functionality that can be tightly integrated, yet spread geographically, organizationally and across disparate systems.

Whether you're looking for a simple way to expose legacy content for broader reuse, or develop a new gee-whiz distributed application, Web services gives you a fast and easy way to do so. And the exteNd technologies in NetWare 6.5 lets you do it without additional expense. Novell exteNd Workbench provides the IDE for development and deployment and exteNd Application server is the platform. You just supply the business rules and logic and you can be providing and consuming Web services in short order. **N**

The Web Service Wizard automatically generates stubs and skeletons.

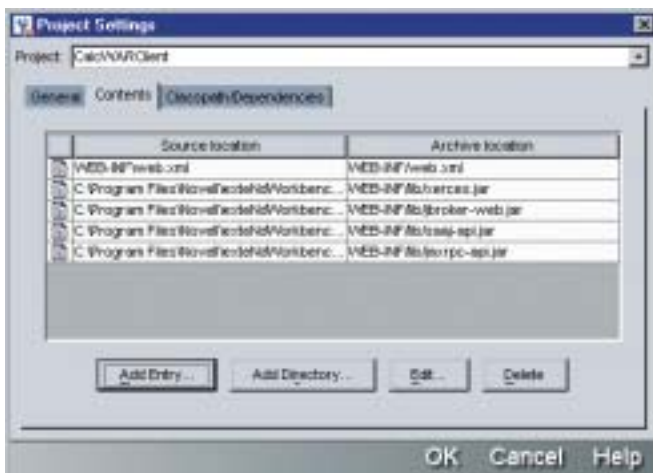


FIGURE 5

Deploying a Web service to Jakarta Tomcat using exteNd Workbench.

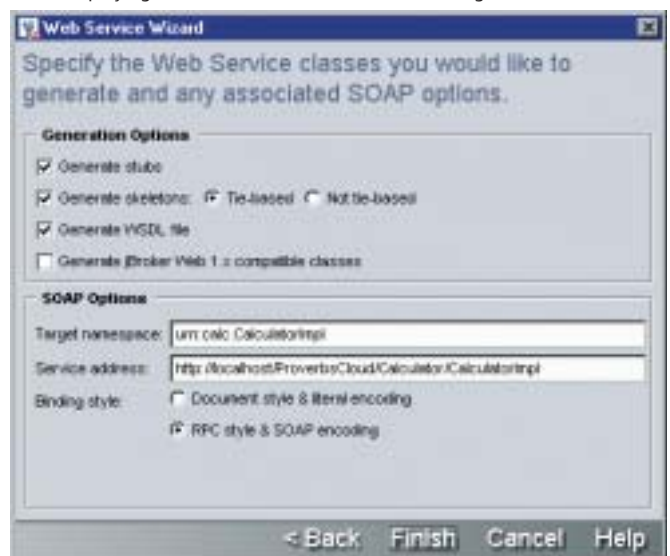


FIGURE 6

Web Services Components

J2EE JAVA 2 PLATFORM, ENTERPRISE EDITION

J2EE is the platform for building distributed enterprise applications using Java technology. The platform consists of an application server or servlet container which manages and coordinates the execution of Java servlets, Java Server Pages (JSP), Enterprise Java Beans (EJB) and other interfaces and services. J2EE is not dependent on the operating system or hardware and provides for distributed processing across locations and service platforms.

XML EXTENSIBLE MARKUP LANGUAGE

XML is a standard for the definition and description of data in context. XML tags describe what a particular selection of text represents and how it is related to other text in the same group or hierarchy. XML makes data and its meaning portable for consumption by other processes and displayable across a wide range of devices in a variety of formats. XML is an important Web services standard as it is used both for SOAP and WSDL.

SOAP SIMPLE OBJECT ACCESS PROTOCOL

SOAP is an XML-based message protocol for invoking Web services in a decentralized, distributed environment. SOAP's text-based format and

XML-like syntax provide a simple mechanism for process-to-process information exchange and invoking services across the Web. SOAP's protocol provides a framework that describes what is in a message (envelope), how to process it (rules for expressing datatypes) and a standard for expressing remote procedure calls.

WSDL WEB SERVICES DESCRIPTION LANGUAGE

WSDL is a protocol for describing the capabilities of a Web service including the protocols and formats used by the service. WSDL is XML-like in format and describes network services with either document- or procedure-oriented information about what a Web service is, how to establish communication and where it is located.

UDDI UNIVERSAL DISCOVERY AND DIRECTORY INTEGRATION

UDDI is a standard for registering and discovering Web services. It defines a directory of contact information and a catalog of available Web services. UDDI defines a framework that enables businesses to discover each other, define how they interact over the Internet, and share information in a global registry. The registry includes business entity information (name, category, location, etc.), service information (category, communication specifications, etc.), and technical information (request/response, security, other metadata).