

# It's Time to Take Notice of Web Services

By BILL SHELDON

*Web services, the latest major trend in computing, stand ready to transform how we interact with computers. Briefly, Web services—which let people access and integrate information from throughout the Web—are a set of function calls or method calls. Because these method calls provide data based on a standard description, they are ideal for programmatic use. The calling interface uses XML-based descriptions, which means that they provide true interoperability across different operating environments.*



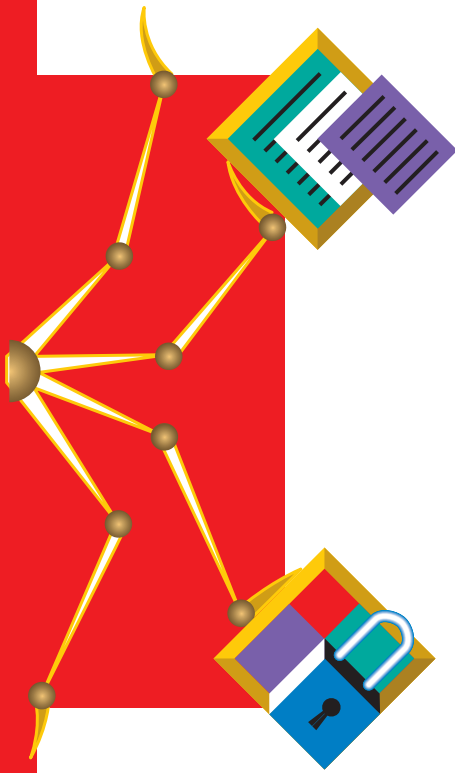
Although Web services are a relatively new concept that has taken much of computing by storm, the sudden adoption of this technology may not be as surprising or as revolutionary as you might think. Web services are the natural evolution of key concepts of enterprise software development. These concepts include the Internet, multi-tier distributed computing, and XML. Web services help bring these concepts

together in such an easy-to-understand way that software developers are trying to show how they have mastered distributed computing's newest paradigm.

## A Brief History

To better understand Web services, let's first consider the Internet and the introduction of COM and Common Object Request Broker Archi-

itecture (CORBA). A few years ago, these two distributed technologies were battling for the enterprise as heavily as Netscape and Microsoft Internet Explorer (IE) were battling for control of the Web. These distributed technologies followed the basic premise that, in the world of business-to-business communication, the ability to simply make data available isn't sufficient. Enterprise software must enable companies to



take advantage of both business logic and business data.

The growth of the Internet led to the thought that businesses located hundreds of miles apart should have access to the same applications: not simply a copy of the same software installed in two locations, but literally sharing one implementation of the same business logic. COM and CORBA provided an implementation-specific way of creating objects that could originate from remote systems. The underlying premise of these implementations was the Interface Definition Language (IDL), the simple description of an action and its associated data. Unfortunately, a fairly significant impediment existed between the implementation of the two types of IDL.

At the same time that COM and CORBA were fighting for acceptance, the Internet was spawning another technology: XML. I don't mean to downplay the importance of HTML as the forbear of XML's

self-describing data structure, but the move from a language that describes how data is displayed to one that describes the data makes XML special. The use of XML has led to technologies such as Extensible Style Language Transformations (XSLT) and XML Data Reduced (XDR) that expand on XML in ways that go beyond the scope of this article.

In addition, a group of individuals and companies wanted to see how they could leverage XML for distributed computing. The goal was to find a simple way to allow a business object in one implementation to access other objects regardless of protocol. The challenge was that each protocol had its own IDL. By using XML to describe an interface, you could create a Web Services Definition Language (WSDL). Then, by combining this new form of IDL with a basic set of support protocols, you could create a set of distributed services that run on the Web.

### Creating the Rosetta Stone

Before you can use WSDL, you need to have a universal method of describing and discovering the services you want to integrate. Universal Description, Discovery, and Integration (UDDI), the first step in working with Web services, is a set of platform-independent standards for describing and registering available services. As the initial means of linking different services, UDDI provides the common grammar that lets different organizations identify the types of services they provide. The UDDI project is described in full at <http://www.uddi.org>.

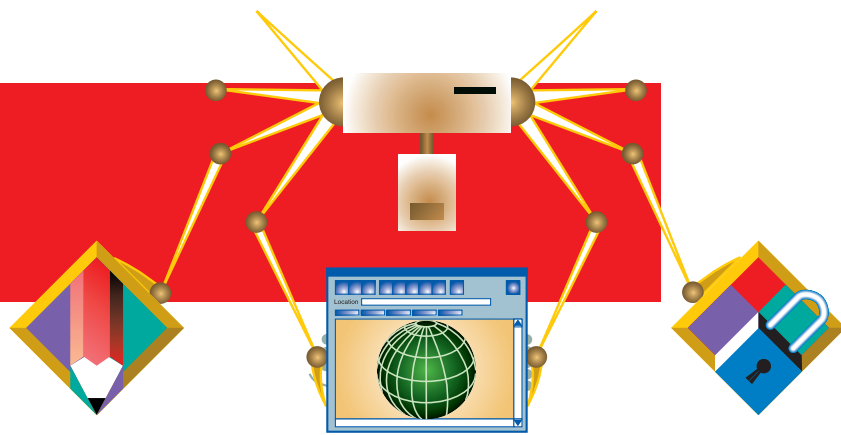
By using UDDI, organizations can publicly or privately advertise the interfaces available from their systems. This information neither

includes, nor is concerned with, the actual implementation behind the interface. After UDDI's role is complete, WSDL-the Web service description-takes over. WSDL defines specific details of a service, including interface names, input parameters, return values, output parameters, and supported protocols for access to the service.

UDDI is primarily a design-time protocol oriented toward identifying services; WSDL is both a design-time and a runtime tool that describes in detail what a service provides. WSDL uses XML to create a standard definition of the interface methods, messages, and data. The WSDL 1.1 specification, originally developed through collaboration between Microsoft and IBM to resolve differences in their initial contracting languages for network services, is under review by the World Wide Web Consortium (W3C) standards committee (<http://www.w3.org/tr/wsdl>). A WSDL document consists of five primary elements, which combine to provide the description of a service:

- **Types** - defines the various data types used as part of the interface.
- **Message** - describes the actual messages (i.e., method request and response) that are supported.
- **Port Type** - identifies the method calls and which messages are associated with each method.
- **Binding** - builds on the Port Types by associating the protocol details and message implementations.
- **Service** - groups a set of related "ports" together.

These primary elements let you describe how to communicate with a Web service. In addition, WSDL relies on two elements that function within these outer elements. The first nested element, Operation, describes the actions that the service supports; Operation functions as part of the



definition of a Port Type. The second nested element, Port, is part of the Service element and describes one method based on binding and a network address.

Thus, WSDL defines the elements of the interface (Types, Messages, and Port Types), as well as which protocols (Binding and Services) are supported for these interfaces. Web services rely on three primary binding protocols for communication. The first two, HTTP Get and HTTP Post, provide a generic method for a Web browser to communicate with a Web service. Not every Web service supports these protocols, however, because some interfaces are more complex than these relatively basic protocols. The third protocol, Simple Object Access Protocol (SOAP), is accepted universally by Web services.

### Cleaning up Interoperability

SOAP, like WSDL, is a joint industry effort. The current SOAP 1.1 protocol definition resides at <http://www.w3.org/TR/SOAP/>. As the primary protocol for a Web service, SOAP is not bound to a single implementation. SOAP does not specify the underlying transport protocol (e.g., RPC, HTTP, SMTP); instead, it specifies a generic transport for information.

At its core, SOAP is an XML-based protocol. Typically, a SOAP message

consists of three parts: the envelope, the header, and the body. The envelope, the top element of the XML document, wraps the SOAP message. The message header, which lets you define encoding rules and other message features, is optional; if defined as part of an interface, the instructions the message header contains may be designated as optional or mandatory for successful handling of the message.

The mandatory SOAP message body contains the remote procedure to be executed. The body of the message contains the information related to the method, the parameter values, and the return values. The normal response from a SOAP request is encoded using the SOAP message format, with the body of the message indicating the method that was executed and the results of the method call.

The key to SOAP interoperability is that you can transmit the same SOAP message as part of an HTTP request, or through a remote procedure call (RPC), and expect the same results-as long as the recipient supports both the underlying transport mechanism and the SOAP protocol. This ability to submit requests regardless of the underlying implementation enables entirely different systems to communicate.

### Working with Web Services

Web services, however, are not simply a set of dry interface descrip-

tions; nor are they a set of concepts. Web services are a method of implementing complex interfaces in a standard way. Every major software vendor, including IBM, Microsoft, Sun Microsystems, BEA Systems, and Oracle, supports the protocols involved in Web services.

Microsoft is tying its next-generation development suite, .NET, to Web services. But the same is true for IBM's WebSphere, Oracle's 9i Application Server, and BEA's WebLogic. These companies and others believe in the future of Web services and they have created an alliance—the Web Services Interoperability Organization (WSIO)—aimed at creating a Web services standard. The alliance hopes to make it easier for businesses and consumers to do business over the Web using lightweight, XML-based Web services and several computing platforms.

The members of WSIO are not only building Web services, but they are also providing tools to let developers build Web services. These tools, which let you take advantage of Web services with minimal work, are the final piece in the Web services equation. The market for Web services is here now. If you aren't integrating Web services as part of your Internet offering, you should probably start doing so today. ■

### About the Author

#### Bill Sheldon

([bsheldon@interknowlogy.com](mailto:bsheldon@interknowlogy.com)) is a principal software engineer and architect with InterKnowlogy in Carlsbad, California. Bill, who holds an MCSD, designs software solutions and develops enterprise infrastructure components for extranets, intranets and the Internet.