

Tect

Web Service Architectures

Judith M. Myerson

© Tect. All rights reserved.

The author and publisher have made every effort in the preparation of this document to ensure the accuracy of the information. However, the information contained in this document is sold without warranty, either express or implied. Neither the authors, Tect, nor its dealers or distributors will be held liable for any damages caused or alleged to be caused either directly or indirectly by this document.

Published by Tect,
29 South LaSalle St.
Suite 520
Chicago
Illinois
60603
USA

Contents

WebServices.Org	3
Service Negotiation	4
Workflow, Discovery, Registries	4
Service Description Language.....	4
Messaging	5
Transport Protocols	5
Business Issues	5
The Stencil Group	5
IBM	5
W3C	6
Wire Stack.....	7
Description Stack	7
Discovery Stack	7
Microsoft	8
Sun	10
Oracle	10
Hewlett-Packard	13
BEA Systems	13
RPC-Style Web Services.....	14
Message-Style Web Services.....	14
Borland	14
Conclusion	14
About the Author.....	15

Web Service Architectures

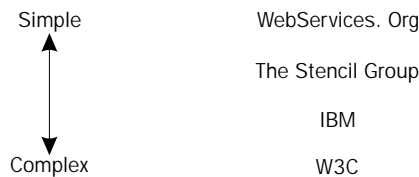
With Web Services, information sources become components that you can use, re-use, mix, and match to enhance Internet and intranet applications ranging from a simple currency converter, stock quotes, or dictionary to an integrated, portal-based travel planner, procurement workflow system, or consolidated purchase processes across multiple sites. Each is built upon an architecture that is presented in this paper as an illustrated stack of layers, or a narrative format.

Each vendor, standards organization, or marketing research firm defines Web Services in a slightly different way. Gartner, for instance, defines Web Services as "loosely coupled software components that interact with one another dynamically via standard Internet technologies." Forrester Research takes a more open approach to Web Services as "automated connections between people, systems and applications that expose elements of business functionality as a software service and create new business value."

For these reasons, the architecture of a Web Services stack varies from one organization to another. The number and complexity of layers for the stack depend on the organization. Each stack requires Web Services interfaces to get a Web Services client to speak to an Application Server, or Middleware component, such as Common Object Request Broker Architecture (CORBA), Java 2 Enterprise Edition (J2EE), or .NET. To enable the interface, you need Simple Object Access Protocol (SOAP), SOAP with Attachments (SwA), and Java Remote Method Invocation (RMI) among other Internet protocols.

Although we have a variety of Web Services architectures, Web Services, at a basic level, can be considered a universal client/server architecture that allows disparate systems to communicate with each other without using proprietary client libraries, according to the WebMethods whitepaper, Implementing Enterprise WebServices with the WebMethods Integration Platform (December 2001). The whitepaper points out that "this [architecture] simplifies the development process typically associated with client/server applications by effectively eliminating code dependencies between client and server" and "the server interface information is disclosed to the client via a configuration file encoded in a standard format (e.g.WSDL)." Doing so allows the server to publish a single file for all target client platforms.

For the purposes of this paper, we present the architecture stacks starting with the most simple, proceed to the more complex ones, and then compare them. After this, we will cover other architecture types from Microsoft, Sun ONE, Oracle, Hewlett-Packard, BEA Systems, and Borland.



Based on initial findings or the current state of implementations, IBM's architecture is most acceptable. All architectures will eventually come into one umbrella, as there is a risk that if companies go away and keep on building their own extensions to the basic architecture stack, the promise of Web Services could be lost. The IBM versions, current and future, could serve as an industry-wide Standard Stack model, after W3C accepts new standards resulting from, for example, the convergence of IBM WSFL and Microsoft XLANG on workflow processes.

WebServices.Org

The following is the Web Services stack from WebServices.Org.

Layer	Example
Service Negotiation	Trading Partner Agreement
Workflow, Discovery, Registries	UDDI, ebXML registries, IBM WSFL, MS XLANG
Service Description Language	WSDL/WSCL
Messaging	SOAP/XML Protocol
Transport Protocols	HTTP, HTTPS, FTP, SMTP
Business Issues	Management, Quality of Service, Security, Open Standards

Service Negotiation

The business logic process starts at the Services Negotiation layer (the top) with, say, two trading partners negotiating and agreeing on the protocols used to aggregate Web Services. This layer is also referred to as the Process Definition layer, covering document, workflow, transactions, and process flow.

Workflow, Discovery, Registries

The stack then moves to the next layer to establish workflow processes using Web Services Flow Language (WSFL) and MS XLANG, which is an XML language to describe workflow processes and spawn them. Microsoft previously achieved recognition for WSDL by working with IBM. History may repeat itself since IBM now has a similar technology to XLANG. In April 2001, IBM published WSFL. Gartner expected IBM and Microsoft to jointly agree to submit a proposal to W3C to combine XLANG and WSFL by the end of 2001. Yet, the W3C Web site has not indicated whether it has received the proposal for consideration. If it did, the proposal has not yet been posted on the Web site (February 2002).

WSFL specifies how a Web Service is interfaced with another. With it, you can determine whether the Web Services should be treated as an activity in one workflow or as a series of activities. While WSFL complements WSDL (Web Services Definition Language) and is transition-based, XLANG is an extension of WSDL and block-structured based. WSFL supports two model types: **flow** and **global** models. The flow model describes business processes that a collection of Web Services needs to achieve. The global model describes how Web Services interact with one another. XLANG, on the other hand, allows orchestration of Web Services into business processes and composite Web Services. WSFL is strong on model presentation while XLANG does well with the long-running interaction of Web Services.

You may declare a Web Service as private, meaning that it cannot expose details of what it does to public applications. You can create Web Services with the WebMethod Attribute in Visual Basic.NET, or with EJB wrappers for existing J2EE applications in either the Internet (public) or intranet (private) environment. You may declare them as public or private methods when you code.

Among the software supporting WSFL is IBM MQ Series Workflow (now known as WebSphere Process Manager) that automates business process flows, optimizes Enterprise Application Integration (EAI) with people workflow, provides scalability, and complies with the Workflow Coalition and multi-platform capabilities. MS XLANG is the language implemented in BizTalk.

Web Services that can be exposed may, for example, get information on credit validation activities from a public directory or registry, such as Universal Description, Discovery and Integration (UDDI). The ebXML, E-Services Village, BizTalk.org, and xml.org registries and Bowstreet's (a stock service brokerage) Java-based UDDI (jUDDI) are other directories that could be used with UDDI in conjunction with Web Services for business-to-business (B2B) transactions in a complex EAI infrastructure under certain conditions. Web Services is still primarily an interfacing architecture, and needs an integration platform to which it is connected. Such an integration platform would cover the issue of integrating an installed base of applications that cannot work as Web Services yet.

The first release of UDDI's Business Registry became fully operational in May 2001, enabling businesses to register and discover Web Services via the Internet. Its original intent was to enable electronic catalogues in which businesses and services could be listed. The UDDI specification defines a way to publish and discover information about services. In November 2001, the UDDI Business Registry v2 beta became publicly available.

Hewlett Packard Company, IBM, Microsoft, and SAP launched beta implementation of their UDDI sites that have conformed to the latest specification, including enhanced support for deploying public and private Web Service registries, and the interface (SOAP/HTTP API) that the client could use to interact with the registry server. In addition to the public UDDI Business Registry sites, enterprises can also deploy private registries on their intranet to manage internal Web Services using the UDDI specification. Access to internal Web Service information may also be extended to a private network of business partners.

Service Description Language

As you move further down the stack, you need WSDL to connect to a Web Service. This language is an XML format for describing network services. With it, service requesters can search for and find the information on services via UDDI, which, in turn, returns the WSDL reference that can be used to bind to the service.

Web Service Conversational Language (WSCL) helps developers use the XML Schema to better describe the structure of data in a common format (say, with new data types) the customers, Web browsers, or indeed any XML enabled software programs can recognize. This protocol can be used to specify a Web Service interface and to describe service interactions.

Messaging

Now, we get to the Messaging layer in the stack where SOAP acts as the envelope for XML-based messages, covering message packaging, routing, guaranteed delivery and security. Messages are sent back and forth regarding the status of various Web Services as the work progresses (say, from customer order to shipping product out of the warehouse).

Transport Protocols

When a series of messages completes its rounds, the stack goes to its last layer: the transport layer using Hypertext Transfer Protocol (HTTP), Secure HTTP (HTTPS), Reliable HTTP (HTTPR) File Transfer Protocol (FTP) or Standard Mail Transfer Protocol (SMTP). Then, each Web Service takes a ride over the Internet to provide a service requester with services or give a status report to a service provider or broker.

Business Issues

Finally, the Business Issues row in the table lists other key areas of importance to the use and growth of Web Services. Without consideration to these points, Web Services could quickly become objects of ridicule.

The Stencil Group

Now, let's take a look at the Stencil Group's Web Services technology stack. It is similar to that of WebServices.Org with three exceptions:

1. The WebServices.Org stack does not divide the layers into emerging and core components. Not doing so could confuse the reader as to which standards are emerging. What is now an emerging standard would become a core standard at a future date.
2. The Stencil Group does not apply Management, Quality of Service, Open Standards, and Security to any layer. The reader could get the wrong impression that they are proprietary and treated as not important. When this happens, the reader will opt for another architecture stack that has these features.
3. The Stencil Group starts the stack with undefined business rules while WebServices.Org begins with a clearly defined business process such as service agreement. The reader could get confused on what undefined business rules are, and how many would eventually be defined.

Layer	Type
Other Business Rules (undefined)	Emerging Layers
Web Services Flow Language (WSFL)	
Universal Description, Discovery and Integration (UDDI)	
Web Services Description Language (WSDL)	
Simple Object Access Protocol (SOAP)	Core Layers
Extensible Markup Language (XML)	
Common Internet Protocols (TCP/IP, HTTP)	

IBM

The IBM Conceptual Web Services stack is part of their Web Services Conceptual Architecture (WSCA) 1.0 (<http://www-4.ibm.com/software/solutions/Webservices/pdf/WSCA.pdf>). It is presented in a slightly different way than that of the first two stacks, by starting with Web Services tools and then showing what each layer is used for.

Tools	Layer		
TPA (Trading Partner Agreement)	Service Negotiation		
WSFL	Service Flow		
UDDI+WSEL	Service Description	Service Publication (Direct UDDI) Service Directory (Static UDDI)	Endpoint Description
WSDL	Service Interface Service Implementation		
SOAP	XML-Based Messaging		
HTTP, FTP, email, MQ, IIOP	Network		
Quality of Service, Management, Security	Business Issues		

The IBM Web Services stack does not show WSCL and ebXML, included in the WebServices.Org stack. It associates the Network layer with IBM MQSeries (now called WebSphere MQ) messaging systems and the Internet Inter-ORB Protocol (IIOP) - a protocol CORBA uses to transmit data, information, and messages between applications. They do not appear in either that of WebServices.Org or The Stencil Group. IBM considers WSDL as a description of the service endpoints where individual business operations can be accessed. WSFL uses WSDL for the description of service interfaces and their protocol bindings. WSFL also relies on WSEL (Web Services Endpoint Language), an endpoint description language to describe non-operational characteristics of service endpoints, such as quality-of-service properties.

Together, WSDL, WSEL, and WSFL provide the core of the Web Services computing stack. IBM perceives UDDI in two categories: static and direct. Static UDDI refers to the Service Directory established after applying WSFL to Service Flow, while direct UDDI pertains to the Service Publication of directory items. Similar to the WebServices.Org stack, the IBM stack applies QoS, management, and security to all layers.

As of May 2001, IBM announced software and tools that enable businesses to create, publish, securely deploy, host, and manage Web Services applications, using the IBM Web Services stack as the framework. They include WebSphere Application Server Version 4.0, WebSphere Studio Technology Preview for Web Services, WebSphere Business Integrator, DB2 Version 7.2, Tivoli Web Services Manager (to monitor performance of all aspects of the Web Services environment), and Lotus software suite (to enable Web collaboration, knowledge management, and distance learning). WebSphere was originally the collective name of IBM's J2EE application server family. It has since been stretched to include most of their middleware and application development offerings, such as MQSeries Workflow (now known as WebSphere Process Manager). IBM currently offers a Web Services Toolkit (WSTK) to help in designing and executing Web Service applications, and enabling them to find one another and collaborate in business transactions without programming requirements or human intervention.

W3C

The W3C Web Services Workshop, led by IBM and Microsoft, has agreed that the architecture stack consists of three components: Wire, Description, and Discovery.

Wire Stack

The following table shows what layers constitute the Wire Stack.

Other "extensions"	
Attachments	Routing
Security	Reliability
SOAP/XML	
XML	

As you will notice, this Wire Stack has extensions to two layers: SOAP and XML. This means whenever the SOAP is used as the envelope for the XML messages, they must be attached, secure, reliable, and routed to the intended service requester or provider. In the stacks of other organizations, SOAP and XML are not treated as "extensions." IBM, for instance, refers to SOAP as a tool for its stack layer, "XML-Based Messaging."

Description Stack

The Description Stack, the most important component, consists of five layers:

Business Process Orchestration		
Message Sequencing		
Service Capabilities Configuration		
Service Description (WSDL)	Service Interface	WSDL
	Service Description	
XML Schema		

This stack starts with orchestration of business processes from which the messages are sequenced, depending on how service capabilities are configured. Whatever comes out of the proposal on combining WSFL and MS XLANG that IBM and Microsoft submitted to the W3C last year will be the tool for the Business Process Orchestration Layer. What needs to be resolved is to consider what parts of WSFL and MS XLANG are more open than the other: transition-based versus block-structured based control flow, and extending versus complementing WSDL among others.

The Service Capabilities Layer is similar to IBM's WSEL as mentioned in IBM WSCA 1.0 and WSFL 1.0 (<http://www-4.ibm.com/software/solutions/WebServices/pdf/WSFL.pdf>). Like IBM, the W3C uses WSDL to describe service interface and service implementation, neither of which is explicitly highlighted in other stacks. WSDL may use a hefty chunk of XML Schema and takes advantage of SOAP/HTTP bindings to WSDL. SMTP, Reliable HTTP (HTTPR), and HTTP Get are other possible bindings that could be used.

Discovery Stack

As the name implies, the Discovery Stack involves the use of UDDI, allowing businesses and trading partners to find, discover, and inspect one another in a directory over the Internet, as follows:

Directory (UDDI)
Inspection

The Inspection Layer refers to WSIL (Web Services Inspection Language) and WS-Inspection specifications. Please refer to the Microsoft section below for more information on WSIL.

Putting all three stack-components together, we have the Architecture Stack.

Wire Stack	Other "extensions"	
	Attachments	Routing
	Security	Reliability
	SOAP/XML	
	XML	
Description Stack	Business Process Orchestration	
	Message Sequencing	
	Service Capabilities Configuration	
	Service Description (WSDL)	Service Interface
		Service Description
	WSDL	
	XML Schema	
Discovery Stack	Directory (UDDI)	
	Inspection	

The remaining part of this paper covers Web Services architectures from Microsoft, Sun, Oracle, BEA Systems, Hewlett-Packard, and Borland.

Microsoft

On May 24, 2000, Microsoft announced it had posted three additional specifications on its XML Web Service Web site: XLANG, SOAP-Routing, and Direct Internet Message Encapsulation (DIME) protocol, a method of packaging up attachments in SOAP messages. SOAP-Routing along with DIME, however, were not included in .NET My Services (formerly codenamed "HailStorm"). This was because the Global XML Web Services Architecture (GXA) replaced SOAP-Routing with WS-Routing which supports one- and two-way messaging, including peer-to-peer conversation and long-running dialogues.

The global architecture builds upon the foundation of baseline specifications - SOAP, UDDI, and WSDL among others. In April 2001, Microsoft and IBM co-presented their vision of this architecture to the W3C Workshop on Web Services. Both vendors contributed to the development and implementation of the W3C Web Services Architecture Stack - more complex than their own versions.

Layer	Type
WS-Inspection, WS-License, WS-Referral, WS-Routing, WS-Security	Global Architecture
UDDI, WSDL, XML, SOAP	Baseline Architecture

The GXA is modular, meaning that you can use one specification with another to address a set of specific requirements. For example, **WS-Referral** does not explicitly specify security, but rather relies on other specifications in the architecture to enable the routing strategies used by the SOAP nodes in a message path.

Other specifications for the global architecture are:

- **WS-Inspection** assisting in the inspection of a site for available services. Due to the decentralized nature of Web Services, this specification doesn't work well if the communication partner is unknown. In such cases, you would be better off with UDDI. This specification (also known as WSIL) was announced by both IBM and Microsoft to the W3C in November 2001. It is another service discovery mechanism (<http://www-106.ibm.com/developerworks/library/ws-wslover/index.html>) and is complementary to UDDI (<http://www.webservicesarchitect.com/content/articles/modi01.asp>).
- **WS-License** describing a set of commonly used-license types and how they can be placed within the WS-Security "credential" tags, such as X.509 certificates and Kerberos tickets.
- **WS-Routing** referring to a simple, stateless, SOAP-based protocol for controlling the route of SOAP messages in an asynchronous manner over a variety of transports such as TCP, UDP, and HTTP.
- **WS-Security** describing enhancements to SOAP-messaging to provide three capabilities: credential exchange, message integrity, and message confidentiality, each of which you could use by itself or in combination with another as a way of contributing to a security model.

Additional specifications will become available as Microsoft releases them for public review.

To accommodate the architecture, Microsoft offers the .NET Framework as a platform for building, deploying, and running XML Web Services and applications. It allows a Web Service consumer send and receive information in a loosely coupled manner, including a description of the Web Services that it and other consumers offer. SOAP is supported by XML Schema Datatypes (XSD), WSDL, XML, and HTTP.

As part of the Microsoft .NET initiative, Microsoft provides a user-centric architecture and a set of XML Web Services, collectively called Microsoft .NET My Services. Using .NET Passport as the basic user credential, the .NET My Services architecture defines identity, security, and data models that are common to all services and can help orchestrate a wide variety of applications, devices, and services - all in one basket.

The initial set of .NET My Services includes:

- **.NET Profile.** Name, nickname, special dates, picture, address.
- **.NET Contacts.** Electronic relationships/address book.
- **.NET Locations.** Electronic and geographical location and rendezvous.
- **.NET Alerts.** Alert subscription, management, and routing.
- **.NET Presence.** Online, offline, busy, free, which device(s) to send alerts to.
- **.NET Inbox.** Inbox for items like e-mail and voice mail, including existing mail systems.
- **.NET Calendar.** Time and task management.
- **.NET Documents.** Raw document storage.
- **.NET ApplicationSettings.** Application settings.
- **.NET FavoriteWebSites.** Favorite URLs and other Web identifiers.
- **.NET Wallet.** Receipts, payment instruments, coupons, and other transaction records.
- **.NET Devices.** Device settings, capabilities.
- **.NET Services.** Services provided for an identity.
- **.NET Lists.** General purpose lists.
- **.NET Categories.** A way to group lists.

Sun

Sun Open Net Environment (Sun ONE) is an open framework to support "smart" Web Services, and in which the Java 2 Platform Enterprise Edition (J2EE) platform plays a fundamental role. The Sun ONE Web Services developer model shows how developers can build Web Services, using XML, servlets, JavaServer Pages, EJB architecture, and Java technologies, as shown below. Note that EJB wrappers allow existing applications to become Web Services.

Java API for XML Processing (JAXP)	Provides a Java interface to DOM, SAX, and XSLT.
Java Architecture for XML Binding (JAXB)	Bind XML data to Java code. A developer uses JAXB to compile XML schema information into Java objects. At runtime, JAXB automatically maps the XML document data to the Java object, and vice versa.
Java API for XML Messaging (JAXM)	Provides a Java interface to XML messaging systems, such as the ebXML Message Service, XMLP, and SOAP
Java API for XML Registries (JAXR)	Provides a Java interface to XML registries and repositories such as the ebXML Registry and Repository, and the UDDI Business Registry.
Java APIs for XML based RPC (JAX/RPC)	Provides direct support for an RPC programming convention for XML messaging systems, such as SOAP and XMLP.

The Sun ONE architecture recommends four types of XML-messaging systems: SOAP, SwA (SOAP with Attachments), ebXML Message Service, and XML Protocol (XMLP). ebXML Message Service extends SwA by adding a QoS network that ensures reliable and secure message delivery. An ebXML message can transport any number of XML documents and non-XML attachments.

Layer	Type
JAXP, JAXB, JAXM, JAXR, JAX/RPC, servlets, JSP, and EJB	Java Technologies
UDDI, ebXML	Registries
WSDL	Service Description Language (may be automatically generated depending on a partner module)
SOAP, SwA, ebXML, XMLP	XML-Messaging Systems

A Web Service example is myServices.ONE that provides a shopping basket spanning multiple sites. Created using iNSight for Forte for Java, this Web Service allows Internet shoppers to view and update their purchases in one basket. Making up myServices.ONE are three basic services:

- **myIdentity** providing identification across the sites. This identifies the users uniquely across multiple sites, so they will not login repeatedly as they go from one site (<http://www.barnesandnoble.com>) to another (<http://www.sears.com>).
- **myBasket** consolidating items from multiple sites. Users can add items to the basket from different sites and view the consolidated list of all items in a central, secure basket.
- **myJeeves** automating the purchase process across various domains. It handles the actual payment and shipping details.

Oracle

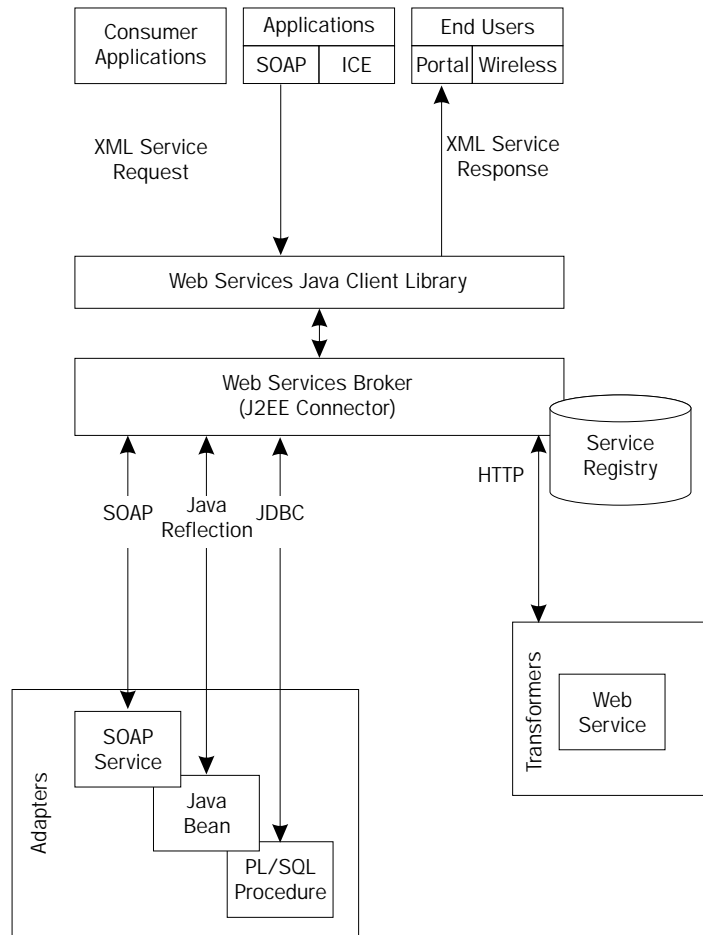
The core of the Oracle9i Web Services framework is the Web Service Broker, a J2EE execution engine deployed in Oracle9i Application Server. Application developers can access the engine using the Oracle Web Services Java client APIs that provide a level of abstraction over the communication protocol to connect to the execution engine (direct Java method calls, PL/SQL calls, HTTP, HTTPS, or JMS (Java Message Service) messages).

Oracle's Web Services framework is based on open industry standards, defining four requirements: description, discovery, request/response, and transport.

Requirement	Standard	Remarks
Description	WSDL	<p>All service attributes, including input/output parameters, version, provider, and copyright/licensing information, are stored in registries. The Oracle9i Web Services framework supports two registries:</p> <ul style="list-style-type: none"> • Service Registry: stores service definitions (called service descriptors). • Application Profile Registry: stores information about the consumer applications that are allowed to access services.
Discovery	UDDI LDAP	<p>Search registries for services with the desired characteristics.</p> <ul style="list-style-type: none"> • Development-time: Service descriptors can be published in UDDI registries (natively or in WSDL). • Run-time: Service descriptors can be stored in Oracle Internet Directory (OID) for security, centralized management, and lookup (via LDAP).
Request/Response	XML	Request and response formats are defined per service in XML documents and XML Schema documents
Transport	SOAP ICE (See Note 1)	Send requests to services and receive responses. Oracle9i Web Services framework includes adapters for common transport protocols and supports custom adapters.

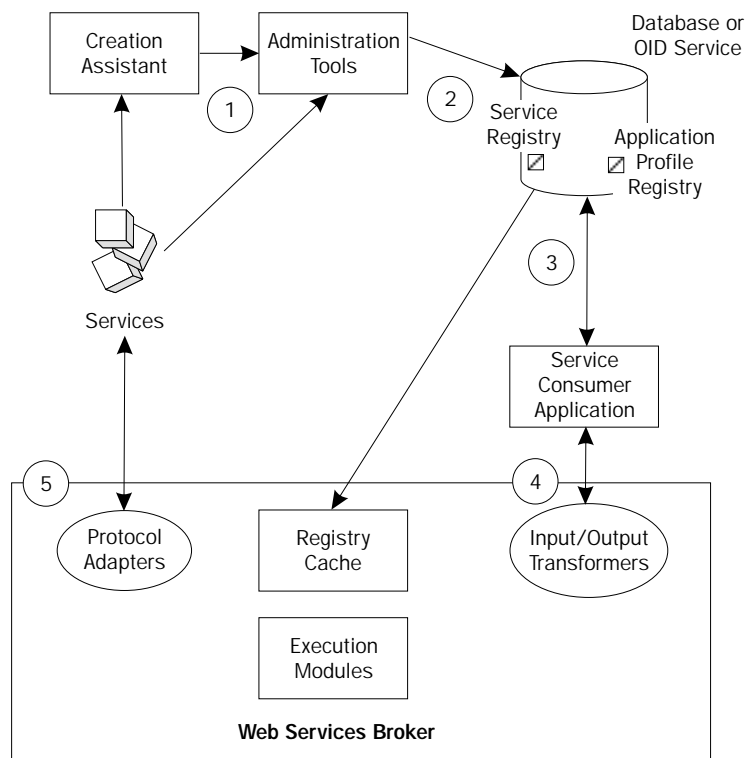
Note 1: Information and Content Exchange. A syndication protocol that standardizes interaction between information publishers and subscribers via the Web

The illustration below shows how various parts of the Oracle9i Web Services framework are related to one another. Starting in the upper-left, Consumer Applications send XML Service Requests to the Web Services Client Library, using SOAP and ICE. The Client Library provides Java and PL/SQL interfaces to the Web Services Broker. Interacting with the Broker for Web Services and Database Services is accomplished through SOAP, Java reflection, or JDBC via software components called adapters. When the Broker returns results to the Services, it dispatches them to the Consumer applications for display to end-users. Software components called transformers allow the framework to support several output formats, including HTML pages, and pages formatted for wireless and mobile devices.



The Oracle9i Web Services framework insulates developers from the complexity of interacting with multiple information sources, protocols, and delivery channels. It is component-based to maximize re-use. It includes tools for creating, managing, and monitoring services.

The following figure gives a developer's view of how various parts of the framework interact.



Working clockwise from the left, the steps are:

1. A Service Provider can start by reusing an existing Web or database application - ideally, one that returns results in XML. If not, Oracle9i Web Services includes utilities that map HTML and other data sources to XML. Oracle9i Web Services also includes a Creation Assistant that generates a simple service from a Web page.
2. Next, a Service Administrator uses a tool of choice (command-line utility or graphical Oracle Enterprise Manager) to register the service, making it available to consumers.
3. Service Consumer Applications query the Service Registry to get the data required to find and invoke a service. Data about Service Consumer Applications, including access privileges, is stored and maintained in the Application Profile Registry.
4. Then, the Service Consumer Application interacts with the Web Services Broker, which uses an input transformer, if needed, to convert the Consumer's request to a format it can use internally. When the service returns a result, the Web Services Broker applies an output transformer, if needed, and dispatches the data to the Service Consumer application. The Service Consumer Application can display the data to end-users, or use it in the flow of some business logic.
5. The Web Services Broker invokes the service via an adapter appropriate for the service's protocol (HTTP, SMTP, etc.).

Hewlett-Packard

In May 1999, HP was the first to develop a Web Services platform under the name E-speak. Hewlett-Packard kept quiet as Microsoft, Oracle, and IBM responded, until March 2001 when it put a large number of software products together in two groups. The first of these groups was the NetAction suite, which contained products for Web Services, with 25 new products, including Bluestone. Bluestone software, which runs transactions for e-commerce, took an important role. Also brought under NetAction were E-speak, Chai (HP's version of embedded Java), and Open Call, an API bundle for call centers, and HP Security. Analysts say HP has possibilities as major player.

The Hewlett Packard Web Services Platform supports both Web Service interactions and Web Service implementation bindings via an architecture that addresses three key infrastructure services, as shown in the following illustration of the HP Web Services Platform. In addition, supporting functions that handle transactional semantics, security, availability, scalability, monitoring, and management are provided by the underlying HP Total-e-Server platform.

Application Processing (Workflow, servlet, EJB, JSP, Cocoon)	Security Transactions Availability Scalability Monitoring Management Tools
Interaction Control (Envelope Processing, Dispatch to Application Components)	
Messaging (Transports, Listeners, Content Format Handlers)	

BEA Systems

BEA Systems develop Web Services on the J2EE platform using the SOAP protocol. J2EE applications expose EJBs and JMS destinations as Web Services. Private registries (possibly based on UDDI) are used to integrate with partners by some applications. Typical enterprise application integration is based on the J2EE Connector Architecture (JCA). Shawn Willet, principal analyst at Sterling, commented, "The JCA technology is a bit immature and a lot of enterprise users may want to go with a ... more mature tool." BEA views their application platform as an integration platform.

Unlike other vendors, BEA (and Borland) uses Business Transaction Protocol (BTP) - an XML dialect for orchestrating inter-enterprise business transactions that address the unique business-to-business (B2B) requirements. This protocol is stack agnostic, so it can be easily implemented in conjunction with other standards such as ebXML or SOAP. For example, a header can be added to the ebXML message envelope to carry the transaction context defined by BTP.

BEA offers two types of Web Services: remote procedure call (RPC)-style and message-style. The first type supports simple Web Services (like stock quotes), is synchronous and is often given by vendors, while the second type is targeted toward a loosely coupled, asynchronous model and is a key requirement for enterprise-class Web Services.

RPC-Style Web Services

You use a stateless session EJB to implement an RPC-style Web Service. When clients, for example, invoke the Web Service specific to a service, they send parameter values to the Web Service, which executes the required methods, and then sends back the return values. RPC-style Web Services are synchronous, meaning that when a client sends a request, it waits for a response before doing anything else. This means that they are tightly coupled with a resemblance to traditional distributed object paradigms, such as RMI and DCOM. One example is a computer screen showing stock quote ticker with an input block, with which the user can get current information on a list of stock during trading hours.

Message-Style Web Services

This type of Web Service is loosely coupled and document-driven rather than being associated with a service-specific interface. When a client invokes a message-style Web Service, the client typically sends it an entire document, such as a purchase order, rather than a discrete set of parameters. The Web Service accepts the entire document, processes it, and may or may not return a result message, such as a manager's acknowledgment of the order. This means the client does not wait for the response before it can do something else. It can wait for it hours, days, or even weeks unless the system has some kind of mechanism to alert the manager to respond within a time frame.

You can also use a message-style Web Service to request a record with the information you need all at once in an XML message - last name, first name, social security number, and so on. This coarse-grained communication example is far better than making three or more separate calls to get the record.

Borland

Borland offers cross-platform development by, for example, using Delphi 6 to create Web Services running on IIS for Windows and then take that same code to Linux, recompile using Kylix 2, and deploy on Apache Web servers. The Internet protocols they use to expose internal Web Services and access external Web Services include: SOAP, WSDL, UDDI, BTP, and ebXML.

Both Delphi 6 and Kylix 2 provide three SNAP™ families you need to build and deploy Web Services. They are:

- *BizSnap* that simplifies e-business integration by creating and using XML/SOAP based Web Services. It interacts with Microsoft's BizTalk.
- *WebSnap* a component-based Web application development framework that supports leading Web Application Servers, including Apache, Netscape, and Microsoft Internet Information Services (IIS).
- *DataSnap* a Web Service-enabled database middleware that enables any client application or service to easily connect with any major database over the Internet. It supports all major database servers such as Oracle, MS-SQL Server, Informix, IBM DB2, Sybase, and InterBase. Client applications connect to DataSnap servers through industry standard SOAP/XML HTTP connections over the Internet without bulky database client drivers and complex configuration requirements; DCOM, CORBA, and TCP/IP connections are also supported.

Conclusion

The W3C Architecture Stack is complex as compared to the stacks presented by other organizations. While this stack is theoretically correct, it may be inconsistent from a customer or end user's perspectives. The W3C's Wire Stack is at the top; however, the SOAP/XML protocols are usually found at the bottom stack layer, such as the WebServices.Org Transport Protocol, the Stencil Group's Common Internet Protocols and IBM Web Services Network. While the W3C's Discovery Stack is at the bottom, UDDI is at the WebServices.Org second layer, The Stencil Group's third layer and IBM's third layer that combines with WSEL. While the WebServices.Org Business Processes and The Stencil Group's Other Business Rules are similar to the Business Process Orchestration in W3C's Description Stack, IBM's stack lacks a layer on business processes. IBM, a primary member of the W3C Web Services Workshop, presented its simpler version, and recently included TPA as its top layer.

Microsoft, another primary member of the workshop, presented the GXA apparently derived from the W3C Architecture Stack. Its global architecture is restricted to Microsoft operating systems, while Sun's ONE Architecture Web Services run on a wider range of platforms. In the near future, we will see .NET on non-Microsoft operating systems, as a result of the ongoing efforts to make .NET Web Services available to non-Microsoft users in all five categories: consumers, service providers, independent software vendors, managed service providers, and corporate application developers. When this

happens, vendors' future market share of Web Services may change today's landscape. Around that time, an architecture stack (similar to IBM's) will rise on the horizon - as the industry-wide standard.

About the Author

Judith M. Myerson is a Systems Engineer/Architect with a Master's Degree in Engineering. A noted columnist and writer with over 150 articles/reports published, she is the editor of Enterprise Systems Integration, 2nd Edition, and the author of The Complete Book of Middleware, and articles in New Directions in Internet Management - all by Auerbach Publishers. In addition to Web Services, her area of interest covers enterprise-wide systems, databases, enabling technologies, application development, network management, distributed systems, component-based technologies, and project management among others. She can be reached at jmyerson@bellatlantic.net.

For the latest information on the impact of Web Services on both Business and Technology, visit:
<http://www.webservicesarchitect.com>