

# WebServices

.NET J2EE XML JOURNAL

WSJ2.COM

Hewlett-Packard  
Application Server  
Version 8.0  
an hp netaction product



PLUS early access release  
of hp web services products

[www.hpmiddleware.com/download](http://www.hpmiddleware.com/download)

**From the Editor**  
**None of the Above**  
by Sean Rhody pg. 5

**Guest Editorial**  
**The Smart Money's**  
**on OASIS BTP**  
by Jim Webber pg. 7

**Guest Editorial**  
**Asynchronous**  
**Web Services**  
by David Chappell pg. 46

**Industry Commentary**  
**Standard Java APIs**  
**for Web Services**  
by Anne Thomas Manes pg. 66

**News**  
pg. 64



Make products and services  
available more quickly  
than ever before

Toward a  
pragmatic  
approach for  
adopting Web  
Services



Page 20

**Peer to Peer:** Web Services Over P2P Networks  **Darren Govoni**  
*How Web services discovery benefits from P2P decentralization* **8**

**Focus:** Apply a Little SOAP to Your Java  **J. Jeffrey Hanson**  
*Dynamically converting existing Java code to a Web service* **14**

**WSJ Feature:** Driving Commerce Value Systems **Tom Tuduc**  
with the Web Services Paradigm *Web services on the move* **28**

**Focus:** JWS: Web Services in Java  **David Bau**  
*Making integration with your enterprise systems easier* **34**

**BPM:** Business Process Management of  **Christen Lee**  
Web Services *Greater potential for return on investment* **42**

**WSJ Feature:** Why Orchestrating Business  **Laury Verner**  
Processes is Key to Web Services *A need for effective integration* **48**

**Product Review:** Orbix E2A Web Services  **Joe Mitchko**  
Integration Platform *XMLBus Edition 5.0 from IONA Technologies* **54**

**Focus:** Using Web Services with J2EE  **Saurabh Dixit**  
*Moving into a technology-independent world* **56**

# **Sonic Software**

**[www.sonicsoftware.com](http://www.sonicsoftware.com)**

# **Sonic Software**

**[www.sonicsoftware.com](http://www.sonicsoftware.com)**

# SpiritSoft

[www.spiritsoft.com/climber](http://www.spiritsoft.com/climber)

# WebServices

.NET J2EE XML JOURNAL

## INTERNATIONAL ADVISORY BOARD

JEREMY ALLAIRE, ANDREW ASTOR, STEVE BENFIELD, DAVE CHAPPELL, PHILIP DESAUTELS, GRAHAM GLASS, TYLER JEWELL, PAUL LIPTON, DAVID LITWACK, NORBERT MIKULA, FRANK MOSS, BOB SUTOR

## TECHNICAL ADVISORY BOARD

DAVE HOWARD, JP MORGENTHAU, ANDY ROBERTS, DAVID RUSSELL, AJIT SAGAR, SIMEON SIMEONOV, RICHARD SOLEY

EDITOR-IN-CHIEF: SEAN RHODY  
EDITORIAL DIRECTOR: JEREMY GEELAN  
INDUSTRY EDITOR: NORBERT MIKULA  
PRODUCT REVIEW EDITOR: JOE MITCHKO  
.NET EDITOR: DAVE RADER  
EXECUTIVE EDITOR: GAIL SCHULTZ  
MANAGING EDITOR: CHERYL VAN SIE  
EDITOR: M'LOU PINKHAM  
EDITOR: NANCY VALENTINE  
ASSOCIATE EDITORS: JAMIE MATUSOW  
JEAN CASSIDY

## WRITERS IN THIS ISSUE

DAVID BAU, DAVID CHAPPELL, SAURABH DIXIT, DARREN GOVONI, JEFF HANSON, CHRISTEN LEE, ANNE THOMAS MANES, MURTHY MANIHA, JOE MITCHKO, SEAN RHODY, TOM TUDUC, LAURY VERNER, JIM WEBBER

PUBLISHER, PRESIDENT, AND CEO: FUAT A. KIRCAALI  
VP, PRODUCTION & DESIGN: JIM MORGAN  
SENIOR VP, SALES & MARKETING: CARMEN GONZALEZ  
VP, SALES & MARKETING: MILES SILVERMAN  
VP, EVENTS: CATHY WALTERS  
VP, BUSINESS DEVELOPMENT: GRISHA DAVIDA  
CHIEF FINANCIAL OFFICER: BRUCE KANNER  
ASSISTANT CONTROLLER: JUDITH CALNAN  
ACCOUNTS PAYABLE: JOAN LAROSE  
ACCOUNTS RECEIVABLE: JAN BRAIDECHE  
ACCOUNTING CLERK: BETTY WHITE  
ADVERTISING ACCOUNT MANAGERS: MEGAN RING  
ROBYN FORMA

ASSOCIATE SALES MANAGERS: CARRIE GEBERT  
ALISA CATALANO  
KRISTIN KUHNLE  
LEAH HITTMANN

CONFERENCE MANAGER: MICHAEL LYNCH  
SALES EXECUTIVES, EXHIBITS: MICHAEL PESICK  
RICHARD ANDERSON

SHOW ASSISTANT: NIKI PANAGOPOULOS  
ART DIRECTOR: ALEX BOTERO

ASSOCIATE ART DIRECTORS: CATHRYN BURAK  
LOUIS CUFFARI

ASSISTANT ART DIRECTORS: RICHARD SILVERBERG  
AARATHI VENKATARAMAN  
TAMI BEATY

WEBMASTER: ROBERT DIAMOND  
WEB DESIGNERS: STEPHEN KILMURRAY  
CHRISTOPHER CROCE  
BILL DARRON

CONTENT EDITOR: LIN GOETZ  
JDSTORE.COM: ANTHONY D. SPITZER  
CUSTOMER SERVICE MANAGER: ANTHONY D. SPITZER

## SUBSCRIPTIONS

FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS, PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT.

SUBSCRIPTION HOTLINE: [SUBSCRIBE@SYS-CON.COM](mailto:SUBSCRIBE@SYS-CON.COM)

COVER PRICE: \$6.99/ISSUE

DOMESTIC: \$69.99/YR (12 ISSUES)

CANADA/MEXICO: \$99.99/YR OVERSEAS: \$129.99/YR  
(U.S. BANKS OR MONEY ORDERS)

BACK ISSUES: \$10 EA, INTERNATIONAL \$15 EA

[SUBSCRIBE@SYS-CON.COM](mailto:SUBSCRIBE@SYS-CON.COM)

## EDITORIAL OFFICES

SYS-CON PUBLICATIONS, INC.  
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645  
TELEPHONE: 201 802-3000 FAX: 201 782-9637  
WEB SERVICES JOURNAL (ISSN# 1535-6906)  
IS PUBLISHED MONTHLY (12 TIMES A YEAR) BY  
SYS-CON PUBLICATIONS, INC., 135 CHESTNUT RIDGE ROAD,  
MONTVALE, NJ 07645

PERIODICALS POSTAGE PENDING

POSTMASTER: SEND ADDRESS CHANGES TO:

WEB SERVICES JOURNAL, SYS-CON PUBLICATIONS, INC.  
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645

## © COPYRIGHT

2002 by SYS-CON Publications, Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system, without written permission. For promotional reprint, contact reprint coordinator, SYS-CON Publications, Inc., reserves the right to review, republish and authorize the readers to use the articles submitted for publication.

All brand and product names used on these pages are trade names, service marks or trademarks of their respective companies. SYS-CON Publications, Inc., is not affiliated with the companies or products covered in Web Services Journal.

# None of the Above

Written by  
Sean Rhody



**Author Bio:**  
*Sean Rhody is the editor-in-chief of Web Services Journal. He is a respected industry expert and a consultant with a leading Internet service company.*  
[SEAN@SYS-CON.COM](mailto:SEAN@SYS-CON.COM)

One of the ingrained attitudes in the United States is the desire to simplify choices to a binary decision. Black or white. Coffee or tea. Winner or loser. Even our expressions reflect this – “two-horse race,” “two-party system.” And it’s no different in the world of Web services. On the one hand, we have the entire world of Microsoft’s .NET initiative. On the other hand we have the many-headed beast known as Java. Java or .NET. Seems simple enough.

But like most simplifications, this one is not entirely accurate, and in fact ignores an entire gamut of choices that exist, and are valid. If Web services was the political system, these choices would be the Independents, the Ross Perot Party.

The simplification that is made with Web services is that in order to do Web services, you need to pick an overall architecture. Naturally, Microsoft, Sun, IBM, and BEA would love for you to continue in this line of thinking because it sells product for them, and not just in Web services.

Not that an architecture platform isn’t important. Without it, your chances of ever doing anything meaningful are pretty limited. But most of the time, you have already selected your architecture. And it may or may not match the platform that the two big camps may want you to use.

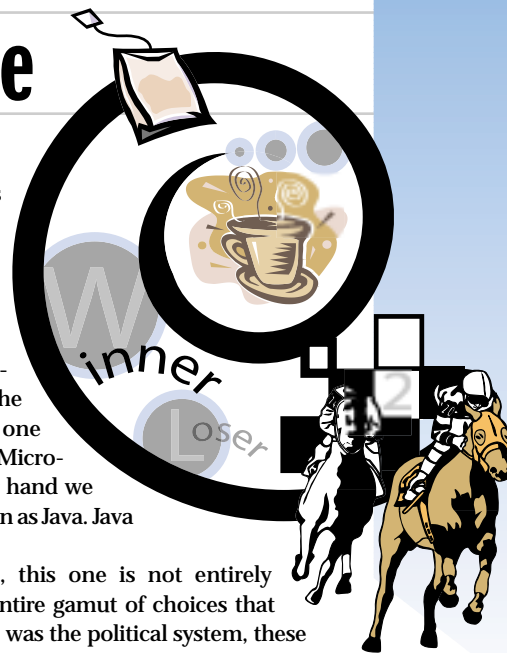
And that’s part of the problem with Web services – we tend to define it as a platform, and assume the installation of a particular platform in order to actually implement Web services.

But that’s a fallacy. In reality, Web services exists as a separate entity, distinct from these platform choices, although perhaps intimately tied to platforms. If we look closely at the Web services stack (stay tuned for a poster from **WSJ** illustrating this for our devoted readers) we can see that at its most basic, Web services consists of UDDI, WSDL, and XML (and possibly, probably SOAP). Now, excluding SOAP for the moment, all of these protocols are neutral to the platform. That’s not to say that one or another doesn’t run better on a particular platform, or that in your architecture it doesn’t belong on a particular platform – of course it does. But what it does say is that the choice is up to you, not to Microsoft, BEA, IBM, or anyone else. You get to choose where UDDI runs within your company. You get to choose where WSDL resides. You get to set your XML schema, and even with SOAP, you still have more than a single option.

The reality, of course, is that there are more options. Companies like Sonic, Tibco, Cape Clear, and Kenamea, to name a few, present a third alternative – the Agnostic Approach. These companies take the road that says there’s nothing wrong with selecting J2EE or .NET – but there’s nothing sacred about them either. And if you think about it, what good would Web services be if we were only able to use them with Java and Windows. For Pete’s sake – the biggest advantage to Web services is that we can wrap CICS and COBOL/Fortran/PL1/whatever on a mainframe and let it look like any other Web service.

Now, my background is clear – remember that I used to edit **Java Developer’s Journal**, and before that **PowerBuilder Developer’s Journal**. I’m clearly a PC kind of guy, so don’t think that **WSJ** is turning into some IBM 390 magazine. But at the same time, isn’t it something special to be able to include all those platforms? Isn’t it about time we were able to create a solution that included the mainframe, UNIX platforms, Windows, and Macs? I think so. I think that in reality, that’s what Web services is all about.

In other countries, one of the options when voting is “None of the Above” – a refusal to select any of the major candidates. While .NET and J2EE are certainly strong options, we need that third choice – None of the Above. ☺

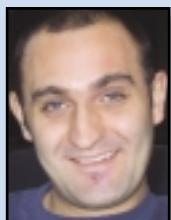


# Actional

[www.actional.com/soap2](http://www.actional.com/soap2)

# The Smart Money's on OASIS BTP

WRITTEN BY



## Jim Webber

Jim Webber, PhD, a senior engineer with Hewlett-Packard, is based at HP's Arjuna Laboratory, HP's transactioning center. Jim architects HP's Web Service Transactioning solution (HP-WST), is involved with standards processes for transactional Web services, and is currently participating in the OASIS BTP effort.  
JIM\_WEBBER@HP.COM

By now we've all heard a fair bit about Web services, a lot of hype and few hints that there's something really innovative going on here. Trudge round any developer conference and you'll hear the chatter of eager developers wanting to roll together a host of disparate Web services into the most fantastic and powerful applications the enterprise has ever seen.

Composing enterprise applications out of Web services is a hot topic, but the practicalities of building real applications out of Web services are a world apart from the sales-speak and hyperbole that we've heard so far.

### Web Services: Some Home Truths

While Web services are beautifully simple and intuitive, with a really gentle learning curve, their simplicity belies a more troublesome implementation. Since the Web services model is so simple, much of the richness (and complexity) that we've been used to in previous enterprise architectures isn't there for us to draw upon – or at least not yet.

Enterprise developers are going to be given a hitherto unparalleled world of choice in terms of the components they'll use, and component suppliers will have to compete on criteria like cost, performance, and functionality. Better still, significant effort is going into improving the whole infrastructure on which the Web services architecture rests, like increasingly robust application servers and reliable transports like IBM's HTTPR. This all sounds like it should make a great platform – and it does, mostly.

With increasingly robust back-end and transport technologies, and plenty of component choice, it would seem that Web services really might be the silver bullet for enterprise computing. There is, however, a dark cloud hanging over this otherwise beautiful horizon: developers will now need to routinely support activities that span multiple Web services across multiple enterprises. This raises the rather prickly issue of how to maintain consistency among these components. While this isn't a new problem, it's a new one for Web services.

### A New Hope: OASIS BTP

About a year ago, some people from a range of vendors with similar concerns got together under the umbrella of the OASIS Business Transaction Protocol (BTP) initiative. Like me, these people spent their days worrying that application developers were going to write software that spanned multiple business systems, and wanted to ensure that these systems would work reliably.

The result of this effort is a draft standard for transactioning in loosely coupled environments like Web

services. OASIS BTP is an innovative, extended transaction model designed to allow work to progress even in the presence of failures – a significantly different take on the problem compared to the EJB or JTS transactioning we've been used to, where failure of any aspect is the undoing of the whole transaction.

While BTP is a lengthy and intricate work, it is predicated upon two fundamental constructs: the Atom and the Cohesion.

- **BTP Atoms** are similar to atomic transactions. The difference is that Web services participating in BTP transactions aren't usually owned by the initiator and thus atomicity via strict two-phase locking can't be guaranteed. However, Web services participating in an Atom are guaranteed the same outcome as all other participants.
- **Cohesions** allow business logic to dictate which combination of participants can fail or succeed, while permitting the transaction as a whole to make forward progress – this allows the transaction to proceed even in the event of failures.

BTP provides the necessary underpinning for conducting activities of real value over Web services, but at a cost. Web service developers will have to make their services BTP-aware using one of a number of BTP toolkits – such as HP Web Services Transactions. There's also a learning process associated with both BTP and the various BTP toolkits. The benefits far outweigh the costs, however, since Web services that support BTP implicitly benefit from being kept in step with other Web services involved in your business.

### How to Ruin an Enterprise Application...

If you really want to ruin a good Web services-based application, then simply ignore the matter of maintaining consistency among your partners', suppliers', and customers' Web services. Stand back and watch as your systems annoy anyone and everyone you do business with by sending them different, conflicting signals with no arbitration. Don't believe me? Just wait until you've "shipped" your first out-of-stock item that the credit card company couldn't authorize, with a courier that can't deliver. Now scale that to the Web – scary, isn't it?

### ...and How Not To

You guessed it. Where business transactions between multiple parties carry some intrinsic value, they must be supported by transactions. For Web services, BTP is the runaway leader in what's currently a one-horse race – it's a simple bet, isn't it? ©



# Web Services vs P2P

written by Darren Govoni

## Web Services Over P2P Networks

How Web services  
discovery can benefit  
from P2P decentralization



**AUTHOR BIO:**

Darren Govoni is a Distinguished Engineer at Cacheon ([www.cacheon.com](http://www.cacheon.com)), where he writes and mentors on subjects such as P2P computing, Web services, and component-based development and assembly. Darren's past research at Metadapt, a company he founded, is the basis for Cacheon products.  
[DGOVONI@CACHEON.COM](mailto:DGOVONI@CACHEON.COM).



Peer technologies seek to build vast ad hoc networks and communities around common interests, objectives, or content. Web services provide a common platform through which information and business processes can be exchanged, combined, and deployed across networks. There are interesting overlaps between the two as each seeks to become a common stack for publishing and discovery across networks. This article explores some of the similarities and differences between Web services and peer technologies and highlights key intersect points that enable interesting possibilities when used together.

## Web Services Architecture Recap

By now, most of us are familiar with the Web services architecture depicted in Figure 1. That is, the service requestor, service provider, and service broker form a process triad that provide publishing, binding, and interaction semantics.

published service descriptors match the implementation details of a particular service. For this reason, the service directory is a key component in the dynamic discovery of services, but the emergence of directories will bring a class of problem affecting the efficient discovery of published services.

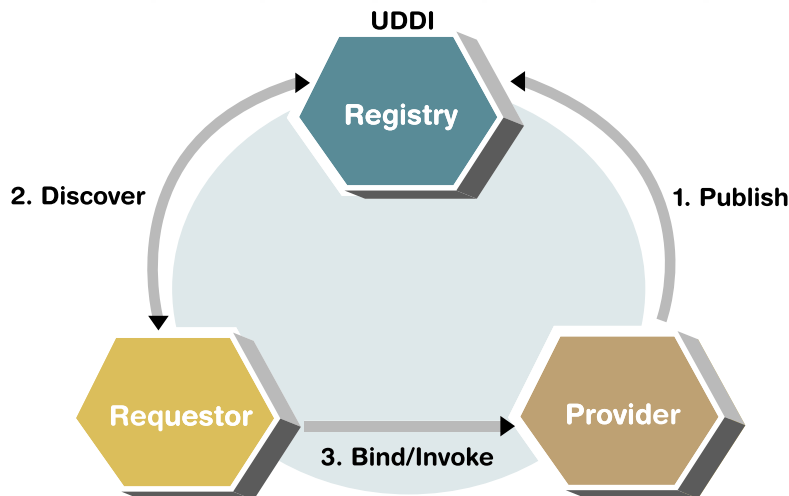


FIGURE 1 | Web services architecture

As the process indicates, a service *requestor* identifies one or more UDDI-enabled directories on a network that it can search for candidate services. It's the role of the directory to carry out queries and return service descriptors to the requesting application, which then binds directly to the service in a service-defined manner. All of this happens relatively automatically through the Web service stack.

The service *broker* exists to separate the details of the service provider from the service requestor and vice versa. This is necessary because the two endpoints can undergo changes in regard to their exact location, protocol, and other binding details. The act of publishing a service in a UDDI requires that it contain the necessary information needed by a client to contact the service directly. It will be the responsibility of the service provider to ensure that

## Registry Fragmentation

It's unclear exactly how registries will be organized across the Internet. Major companies are offering free registry services that you can build your Web service clients to for testing purposes (*note: visit [www.uddi.org](http://www.uddi.org) for a complete list of publicly accessible UDDIs*). The problem relates to the observation that no single directory will exist to provide discovery for all services. Rather, many UDDI directories will crop up over time as companies with common interests band together to ensure that they can benefit from Web services and seek to endorse reliable facilities and services to make that happen. No one company will provide this reliability.

Also, there are many opportunities for *directory-oriented* companies to attract service-providing constituents and assist in the formation of new markets around those services. Since many markets seek to benefit from Web services, various directories will probably offer different types, numbers, and qualities of services therein, all in an effort to compete for and establish new service-oriented markets around them.

## Registry Specialization

In addition to the natural fragmentation of registries across the Internet, some sectors will seek to build or couple with highly specialized registries where entries of particular interest are kept close together or frequently visited. For example, the financial services vertical might foster directories where only financial services are published and as such, applications and systems in those fields will gravitate towards them, strengthening their position in the process.

As specialized registries grow in popularity, they become more valuable to the constituents and the cycle continues, possibly weeding out inappropriate or irrelevant services along the way. This sort of refinement is much how we see things evolve around us, in nature and the complex systems of which we are already a part. In addition, corporate mergers/acquisitions seek to unify similar assets, markets, and revenue, perpetuating this trend.

A force resisting a single registry for all services will coexist with the forces unifying vertically aligned registries as well. The two forces simultaneously seek to bring *together* and to *separate*. This fractal phenomenon repeats itself on many scales. Vertical sectors will have a variety of registries that serve their community as a whole. Applications and services can be published and hosted throughout the network and used on demand. However, the same problems repeat themselves on this scale as well. Namely, how do service requestors (applications, systems, or people) know where to look? Even as registries unify or multiply, these forces cause disruptive changes regarding where applications can expect to find them.

Since UDDI services are provided on some fixed port address, it must be known in advance of the request (on the client) so the requesting applications can attach to the registry at *request time*. The precise knowledge about which registries to search will have to be managed and disseminated to client applications. That process carries not only the logistical burden of doing it, but also the legacy burden of maintaining it. It would be nice if these burdens could be diminished. One possible solution is to disseminate that information separately from the client application, instead placing the burden on users to acquire and input the addresses of directories they wish to search. However, leveraging the qualities of peer networks might offer another approach.



“As specialized registries grow in popularity, they become more valuable to the constituents”

## Qualities of Peer Networks

Peer networks like Gnutella and Freenet bear a unique quality that sets them apart from other peer networks – they are truly

distributed. Unlike Napster, which is a brokered peer network, networks like Gnutella have no central index or server. Rather, they operate collectively across all nodes in the network.

Searching a true peer network results in a request that seeks across the nodes of the network until most or all nodes are reached. Much like ripples in a pond, these searches fan out, seeking routes to unvisited nodes and maximizing their radius. However, the algorithms require the search to acquiesce after a certain number of “hops” have been reached. They do this to avoid lengthy search times and ping-pong effects where nodes are accidentally visited multiple times. Depending on the topology of the peer network, this has little effect on the precision of a search, but in general, peer networks operate best when *highly connected* – a term used to describe the nature of the network graph.

The benefit of these complex search algorithms across interconnected networks is that the *requestor* and the *provider* do not need a priori knowledge of one another. This is true even in Web services, but the knowledge of the provider has been displaced somewhat by the knowledge of a given repository.

Peers in peer networks behave like model routers in that they receive and forward message traffic from and to their neighbor peers. Much like network traffic

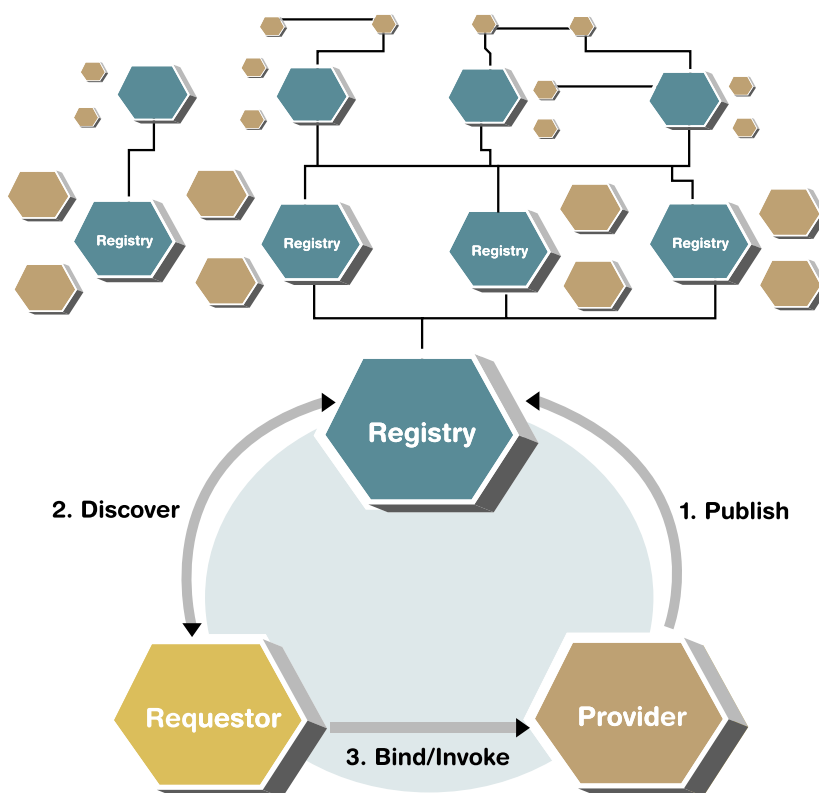


FIGURE 2 | Clustered federation of directories

# **Sams Publishing**

**[www.sampublishing.com](http://www.sampublishing.com)**

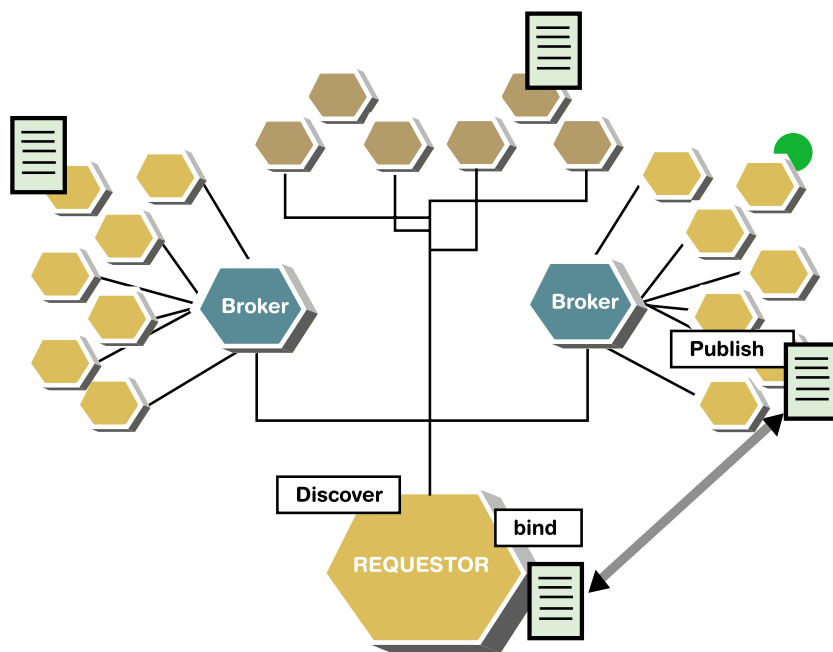


FIGURE 3 | Binding via federated discovery

routers, they operate independently on one level, yet collectively on another. This offers a strong degree of resilience as traffic can be routed and rerouted in numerous ways to ensure maximum exposure or endpoint reliability as well as discovery. The Gnutella protocol is based on a simple “find a friend” paradigm. It requires only that a given peer know of at least one other peer to send and receive messages across the entire network.

### Leveraging P2P and Web Services

Both Web services and P2P computing models provide a discovery process. Current peer networks focus more on the discovery of content in the form of common files published with metadata. The content served by a UDDI repository is metadata describing Web services. However, where peer networks serve their content in a highly distributed fashion, UDDI provides a central discovery mechanism for service provisioning. One intersection between P2P and Web services involves bringing the decentralization aspect of P2P to the central service discovery aspect of Web services.

In a decentralized peer network, content is described and indexed locally to a given peer. Since each peer acts as a tiny router, forwarding and responding to search queries as they pass through, it has an opportunity to address them in a peer-specific way. Thus, each peer qualifies the search criteria against its locally

maintained database and responds accordingly. In this model, no centrally managed index is required to span the network. Peers are interconnected and forward traffic in optimized ways that span all nodes.

### Peer Services: Peer-based Web Service Provisioning

It's not difficult to envision a class of peer networks that promote the decentralized searching mechanics we've discussed but provide Web service descriptors as their content just like a UDDI. In this way, any

client can specify the criteria for a service candidate and fire it across the network. The search will ripple across the peer nodes and responses will feed back to the originating client. Thus, no additional effort or knowledge is needed by a search client to increase its chances for a match, the peer network absorbs this responsibility. Nor does the service provider need to know in advance where best to publish its service to increase its exposure.

Using this highly decentralized approach, service providers can themselves become peers in a vast network of Web service providers, or *peer services*. Queries for services pass directly to service providers who can easily maintain the qualitative data associated with their service, including the number and types of services they wish to offer. Because the peer network is dynamic and not a cached or indexed searching scheme, information discovered at any given moment is likely to be current and availability higher than if discovered statically through a broker.

### Peer Directories: Clustering Directory Services Using Peer Technology

We can see that the process of publishing and discovery can be mapped from a static, central model to a dynamic, decentralized one with the benefit of freeing applications and services from static address logistics and maintenance. However, there are degrees of centralization that can mitigate between the qualities of centralization and decentralization. For example, rather than require all peers to publish their own service

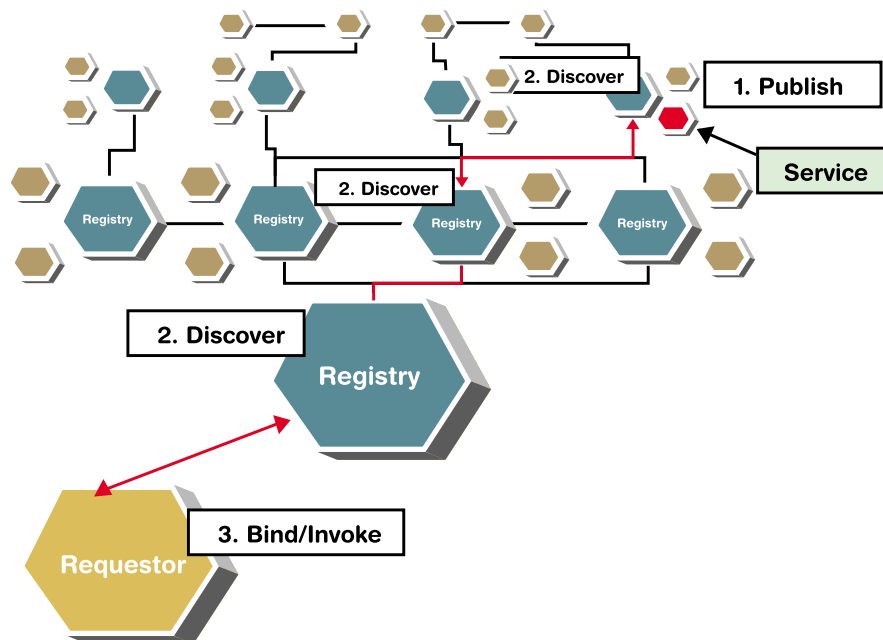


FIGURE 4 | Peer-based propagating discovery



“Peer networks like Gnutella and Freenet bear a unique quality that sets them apart from other peer networks – they are truly distributed”



descriptors (locally), you can employ a federation of UDDI-enabled, peer registries that operate in a decentralized way, yet do not directly provide the peer service implementation. In this way, UDDIs can join and leave a dynamic federation of directories. Such federations might indeed represent common interest groups and verticals as we mentioned earlier. Using peer technology, the general topology and addressing of registries in the federation can be concealed from the endpoint client performing a search. Figure 2 shows us what such a clustered federation of directories might resemble.

Figure 3 shows the binding process, which occurs as before – directly between the provider and requestor. Publishing is done locally to each service broker/directory and queries are propagated among them. Once service information is discovered, the registry is no longer needed and binding can occur immediately.

It might also be the case that service registries provide information pointing to additional registries. Peer technology can be

embedded within registries to cooperatively respond to queries. In contrast to a federation of registries, this approach allows a client to access a single registry that has the scope of many. The difference is centered on what the client knows and how it behaves. In Figure 3, the client knows less and provides mechanics for sending queries across the peer federation. Figure 4 shows the client accessing a single registry and finding a service in another registry that was discovered at query time.

### Service and Registry Syndicates

The ability to form ad hoc federations of services (peer services) or directories promotes the formation of service-oriented syndicates. Service syndication seeks to promote in-demand services by offering them throughout the federation, not just at one location. This also increases the reliability and availability of the service, since it can be acquired and bound to from a variety of providers. The incentive is such that use of the service benefits the syndicate members regardless of who provisions the service, since that responsibility is now distributed.

Probably more practical will be the syndication of directories (or registries). This too will benefit the reliability and availability of the discovery process. Regardless of whether a particular service is online or not, if it isn't visible in a particular broker it might not be found. To help this, UDDI federations will share or cache service descriptors or references to one another such that clients can either discover the remote registries directly or simply search them all at once using a more peer-based search algorithm.

### Practical Issues

There are certainly many issues that need to be resolved before peer technologies can exist peacefully within corporate boundaries and even more so for Web services. Security is still the number one concern, as companies want to ensure that the forthcoming automation doesn't leave them vulnerable to malicious attack or unauthorized use.

For peer networks, the lack of central control is also a big concern. The blessing is also the curse. Finding an acceptable balance between centralization and decentralization is key, yet any approach must justify itself against the benefits, the cost, and the need.

That peer networks are highly distributed doesn't mean that they can't be centrally controlled. Networks are highly distributed, yet companies have found ways to manage their private networks and

secure them from trespassers.

From an authentication perspective, businesses will have to resolve how they intend automatic or anonymous interactions to occur, even if (and especially if) there are usage fees involved. The idea is to let the wanted people through, and none of the unwanted, i.e., not to hinder the expansion of their e-commerce.

### Conclusion

In this article, we've explored the basic architectures of Web services and P2P systems. We also looked at ways in which Web services discovery can benefit from P2P decentralization. At first, it seems that searching UDDIs across networks is a key point of development for merging these two computing paradigms. In the future, directories may not even be required as service providers publish their own services (peer services) in the same way that you would expose content on their computer for sharing across FreeNet or Gnutella. It's quite certain that many issues will need to be resolved before such an approach would gain corporate appeal, but the technology pieces are ready and the new frontier of peer-based services lies ahead. ©

“the technology pieces are ready and the new frontier of peer-based services lies ahead”





## Apply a Little SOAP to Your Java

### Dynamically converting existing Java code to a Web service

It's beginning to look like you can't talk about Web development, enterprise application integration, XML, professional football, or cat juggling without someone mentioning Web services. If Web services have accomplished anything, succeeded in cornering the hype market.

In reality, they are starting to show up and they're proving they can actually get some work done. Perhaps not anything too exciting yet, but the possibilities they present, to imaginative propeller-heads, are downright exhilarating.

With enough creativity and the right Web services, I'll be able to craft a framework of software goodies to allow me to aggregate my financial portfolio, my fantasy football scores, the weather in Cameroon, the quote-of-the-day, and the time in Guam into a single, composite document I can bend, fold, and mutilate to display on my magic Orphan Annie decoder ring. If this doesn't prove useful, I don't know what will.

Web services, by themselves, are presently just globs of XML data floating around in HTTP ether waiting for someone or something to take them out for a good time. Since Java is being used for everything from ASP to ZSP, let's find out how we can turn some of it into something I like to call "Web services."



### Dirty Deeds Done With SOAP

Web services want to be, need to be, just have to be, associated with the terms "pervasive" and "ubiquitous." This means they must share their information and behavior using something everybody uses - XML.

XML is the foundation a Web service builds its house on. Everybody likes XML. XML does this and XML does that and XML...well, you know the rest. XML has infiltrated our systems and processes to make itself the number one choice for data transfer and integration for 9 out of 10 people who think they matter. Since XML is

used primarily for data transfer, how will we represent behavior for a Web service? We simply apply a little SOAP.

The Simple Object Access Protocol (SOAP) has found its way to the forefront of the Web services scene and stepped forward to handle the task of representing data and behavior simultaneously on different platforms. SOAP is an XML-based protocol used to describe Web service messages and user-defined datatypes, along with remote procedure calls (RPC) and responses.

SOAP does three things for us:

1. It defines a Web service as a message and defines the intended recipient of the message.
2. It presents a way to define and serialize arbitrary datatypes.
3. It defines operations a Web service can perform as remote procedure calls and the responses returned by these calls.

### The Envelope, Please

A SOAP message is encapsulated in an element called a "SOAP envelope." The envelope consists of an optional header element and a mandatory body element. The optional header can be used to represent SOAP extensions for things such as transaction management and authentication. The body element is used to contain the name and data for each operation of a message and one optional element known as a "SOAP Fault," which contains error or status information.

Listing 1 shows a SOAP envelope containing a body used to call the getTime method; Listing 2 shows a SOAP envelope containing a body used to return an error from the getTime method.

### Namespaces

A namespace is an arbitrary domain that provides a means for identically named elements to coexist. A file sys-



#### Author Bio

Jeff Hanson has more than 17 years of experience in the industry, including working as senior engineer for the Windows OpenDoc port and lead architect for the Route 66 framework at Novell. He is currently chief architect for Zareus, Inc.

TABLE 1: Common prefixes and corresponding namespaces

Prefix	Namespace	Definition
soapenc	http://schemas.xmlsoap.org/soap/encoding	SOAP 1.1 encoding
wSDL	http://schemas.xmlsoap.org/wsdl/soap	WSDL 1.1
xsd	http://www.w3.org/2001/XMLSchema	XML Schema

# **iWay Software**

**[www.iwaysoftware.com/guaranteed](http://www.iwaysoftware.com/guaranteed)**

tem provides a good example of a name space. In a Unix-like file system, I can have two files named `mycontacts.txt` residing on my hard drive. One file resides in the directory, `/home/myname/contacts`, and an identically named file is in the directory, `/home/myname/mydocs`. Since the fully qualified name for each file is different, there are no name collisions. In other words, to reference one file, I might enter `/home/myname/contacts/mycontacts.txt` and to reference the other file, I would enter `/home/myname/mydocs/mycontacts.txt`. As you can see, when applying the full paths to the files, we end up with names that are unique.

XML also uses namespaces to avoid name collisions. A namespace in XML is identified by a Uniform Resource Identifier (URI). URIs are strings that typically identify resources on the Web such as documents, images, services, e-mailboxes, and other resources. An XML URI points to a file or Web address that defines the namespace. An XML document can assign a namespace a variable name in order to designate which namespace subsequent elements belong to. Listing 3 demonstrates the definition of multiple namespaces in an XML document.

The example refers to a bowlers namespace and assigns it a variable name, `owl`. Also, shown is a second namespace, `lanes`, which is assigned to the variable, `lane`. The variable names are usually referred to as prefixes, since they're used as prefixes for element names.

Default namespaces can also be declared and used. Any element or attribute that isn't prefixed/qualified, belongs to the default namespace (see Listing 4).

In the example, the `<Name>` element has been used more than once. Since the teams namespace was declared as the default, the teams elements don't require a prefix.

When different document type definitions

(DTDs) are included in the same XML document, name collisions can occur. XML namespaces solve the name-collision problem between elements that have the same name in these different documents. Namespaces can be shared by multiple documents, and businesses will typically use namespaces to identify elements specific to their documents.

A namespace attribute defines a shorthand prefix for a namespace that's used throughout a document. For example, `xml`

request and response data. A typical SOAP request over HTTP proceeds as shown in Figure 1.

A SOAP method can be thought of simply as a SOAP-encoded HTTP POST request. SOAP requests should use the `text/xml` content-type. A SOAP request over HTTP should indicate the SOAP method to be called using the `SOAPAction` HTTP header. The `SOAPAction` header contains an application-specific method name.

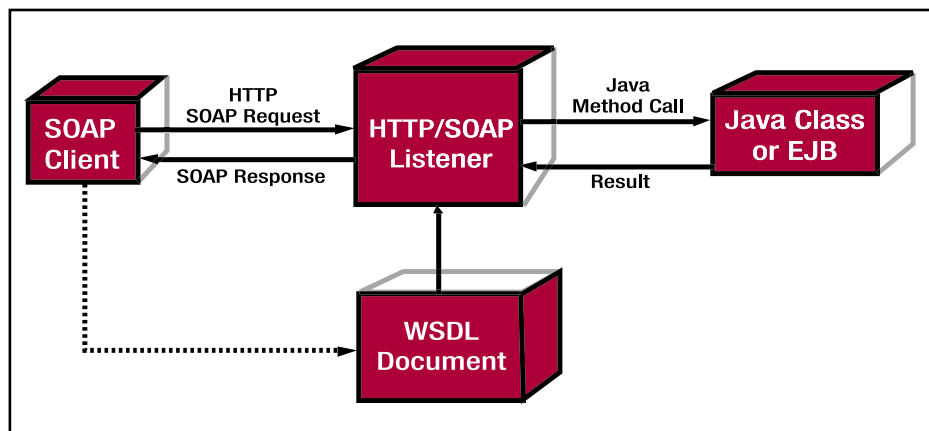


FIGURE 1 | A typical SOAP request over HTTP

`ns:xsd` defines `xsd` as a shorthand prefix for the namespace `http://www.w3.org/2001/XMLSchema`. With this shorthand prefix defined, we can qualify a reference to a datatype later in the document by using just the prefix, as in `xsd:int`. Table 1 shows some common prefixes and corresponding namespaces.

The `targetNamespace` attribute defines the namespace to which all names declared in a given element belong.

## SOAP and HTTP

Since HTTP is ubiquitous throughout the Web infrastructure, it's being presented as the protocol of choice for transporting SOAP

A SOAP request is an XML document containing the in and out parameters of the SOAP method. These values are child elements within the SOAP body and share the same namespace as the SOAP method name.

A SOAP response is similar to a SOAP request, in that the content for the response must be contained within the SOAP body and typically uses the same datatypes as the request.

## Handling SOAP Messages

Since HTTP is the primary protocol used to transmit SOAP messages, we want to find a mechanism in Java that will handle HTTP for us. The component we build needs to do three things:

1. Parse the SOAP message from XML and convert it into a Java method call.
2. Invoke the object responsible for handling the message.
3. Serialize a response or error (if either is needed) back into XML and deliver it to the submitter of the request.

Java HTTP servlets are designed to handle the HTTP protocol, so that's what we'll use.

**// XML has infiltrated our systems and processes to make itself the number one choice for data transfer and integration for nine out of ten people who think they matter"**

## Servlets to the Rescue

A servlet is a Java object designed to handle a request/response-based protocol. Servlets that extend `javax.servlet.http.HttpServlet` are used to respond to HTTP requests. We'll exploit the `HttpServlet` to implement a SOAP-over-HTTP handler that will field HTTP requests and respond with appropriate responses or errors. We'll use the SAX parser APIs provided by the Apache/Axis project to handle any XML parsing tasks. The code in Listing 5 will be the servlet implementation we'll dynamically produce. (Listings 5-7 can be found at [www.sys-con.com/webservices/sourcecode.cfm](http://www.sys-con.com/webservices/sourcecode.cfm).)

## Some SOAP with Your Java?

Java is being used in a vast number of enterprise/Web application architectures. In fact, some would argue (at the risk of starting a holy war) that Java is the language of choice for enterprise/Web application development. We'll avoid military action here and just assume that Java is worthy of our attention and deserves an attempt at Web services integration.

Java is a very descriptive language. At runtime, Java presents all kinds of information about its environment and the objects running in it. We can exploit this information to dynamically generate data and objects that are needed to create and publish a Web service.

## Reflections

Java provides a powerful API for obtaining information about classes and objects at runtime. This API, contained in the `java.lang.reflect` package, will be used to dynamically create the servlet code for our

// Since XML is used primarily for data transfer, how are we going to represent behavior for a Web service? We simply apply a little SOAP"

Web service. Listing 6 provides our reflection class.

As the code demonstrates, we handle EJBs and regular Java classes quite similarly. The main difference is the lookup and creation of an EJB, as opposed to a standard object instantiation for a regular Java class.

## Datatypes and Encodings

The latest XML Schema at <http://www.w3.org/TR/2001/PR-xmlschema-2-20010330> defines a set of built-in datatypes for the namespace, <http://www.w3.org/2000/10/XMLSchema>. Along with these built-in datatypes, user-defined datatypes can be specified. Table 2 shows a list of some built-in XML datatypes and the built-in Java datatypes we can use to represent them

XML Schemas can be used to specify user-defined datatypes. These datatypes can be represented in XML as part of a SOAP envelope. This gives us the ability to encode any datatype in a SOAP message that can be described in an XML schema.

## Publish That Thing

Once we have our Web service implementation complete, we must provide a way for clients/consumers to discover our Web service. The current trend for solving this problem is to publish a Web service in a globally accessible registry. The Universal Description, Discovery and Integration (UDDI) specifications define one such registry.

## UDDI

UDDI uses XML to define a distributed registry of businesses and their Web service descriptions. UDDI defines a meta-data construct, known as a `tModel`, to identify technical specifications for Web services. Of interest to us is the `overviewURL` child element of the `overviewDoc` element within a `tModel` construct. The `overviewURL` element is used to store a URL that points to a specification document describing our Web service. This specification document is called a Web Services Description Language (WSDL) document.

## WSDL

A WSDL document is used to describe a Web service's operations and data types in XML. A WSDL document specifies a namespace to which the Web service and its datatypes belong. If our Web service defines elements or documents that don't need to be considered globally unique, we can use the URI, <http://tempuri.org/>, as the base for a namespace within our WSDL document. This "temporary" URI is useful when testing a version of our Web service that isn't ready for production. It's also useful for versioning an entity without having to come up with new, unique URIs for each version

A summary of the WSDL file, known as an implementation file, is what's actually used to publish a Web service to a UDDI registry. Listing 7 is an example of a WSDL


TABLE 2: Built-in Data Types

Java Type	XML Type
<code>java.lang.String</code>	<code>string</code>
<code>java.lang.Boolean</code>	<code>boolean</code>
<code>java.lang.Double</code>	<code>double</code>
<code>java.lang.Float</code>	<code>float</code>
<code>java.lang.Integer</code>	<code>int</code>
<code>java.lang.Long</code>	<code>long</code>
<code>java.lang.Short</code>	<code>short</code>
<code>byte</code>	<code>byte</code>
<code>java.util.Date</code>	<code>dateTime</code>
<code>byte[]</code>	<code>base64Binary</code>
<code>byte[]</code>	<code>hexBinary</code>
<code>java.math.BigDecimal</code>	<code>decimal</code>

// In fact, some would argue (at the risk of starting a holy war) that Java is the language of choice for enterprise/Web application development"

file. In this example, you'll notice that a WSDL file is comprised of elements with names such as Message, PortType, Binding, and Service. The main thing to remember is that a WSDL file defines an abstract definition of our Web service and details about the concrete implementation of our service. We won't delve into the details of each element in this article, instead, we'll point out that the Apache/Axis project defines a valuable tool called Java2WSDL, which will dynamically generate a WSDL file for an existing Java class.

## Summary

SOAP, WSDL, UDDI, and XML form a robust and comprehensive framework for delivering Web services. Java provides an extensive API set for dynamic discovery of object and class information at runtime. Using these technologies with a toolkit such as the Apache/Axis project, gives us the power to produce Web services dynamically with little or no interaction with the original developer of a Java class or EJB. 



### Listing 1

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
  xmlns:xsd=http://www.w3.org/1999/XMLSchema
  xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xmlns:SOAP-ENC=http://schemas.xmlsoap.org/soap/encoding/
  SOAP-
  ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <myns:getTime
      xmlns:myns="http://www.myserver.com/webservices">
      <city xsi:type="xsd:string">Buffalo</city>
      <state xsi:type="xsd:string">New York</state>
    </myns:getTime>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Listing 2

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
  xmlns:xsd=http://www.w3.org/1999/XMLSchema
  xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xmlns:SOAP-ENC=http://schemas.xmlsoap.org/soap/encoding/
  SOAP-
  ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Client</faultcode>
      <faultstring>Time retrieval failed: missing city
        name.</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Listing 3

```
<Browsers
  xmlns:bowl=http://www.myserver.com/schema/browsers
```

```
xmlns:lane=http://www.myserver.com/schema/lanes>
  <bowl:BowlerInformation>
    <bowl:BowlerName>Big Daddy Jones</bowl:BowlerName >
    < lane:LaneInformation>
      < lane:Name>Rock Roll</lane:Lane>
      < lane:Street>123 Anywhere St.</lane:Street>
      < lane:City>Mytown</lane:City>
      < lane:State>MN</lane:State>
      < lane:Zip>54321</lane:Zip>
    </lane:LaneInformation>
  </bowl:BowlerInformation >
</Browsers >
```

### Listing 4

```
<BowlerInformation
  xmlns:bowl=http://www.myserver.com/schema/browsers
  xmlns=http://www.myserver.com/xmlns/teams>
  <BowlerData>
    <bowl:Name>Big Daddy Jones</bowl:Name>
    <bowl:Handicap>195</bowl:Handicap>
    <Teams>
      <Team>
        <Name>Hurricanes</Name>
        <Score>210</Score>
      </Team>
      <Team>
        <Name>Gutter Rats</Name>
        <Score>180</Score>
      </Team>
    </Teams>
  </BowlerData >
</BowlerInformation >
```

Download the code at

[sys-con.com/webservices](http://sys-con.com/webservices)



# Sitraka

[www.sitraka.com/jclass/ws](http://www.sitraka.com/jclass/ws)

written by Murthy L. Mantha

# Make products and services available more quickly than ever before



#### AUTHOR BIO:

Murthy L. Mantha is  
an independent

software consultant based in Austin, TX,  
focusing on Web services. He has more  
than 25 years of IT industry experience,  
ranging from start up companies to  
major corporations. His experience  
includes enterprise software architecture  
for e-commerce systems, Web portals,  
and other Internet solutions in the areas  
of automotive, banking and eLearning  
applications. Murthy holds 2 patents in  
Internet technologies.

MLMANTHA@YAHOO.COM

Web Services Primer

**T**his article identifies and describes the roles of major participants within the evolving complex ecosystem of Web services. A brief introduction to Web services is offered. Next, the limitations and/or issues with Web services in general as seen by these participants are described. Then, a methodology for adopting Web services will be proposed. Finally, a discussion of what it takes for an entity to participate in the Web service space is presented.

## Toward a pragmatic approach

## What Are Web Services?

Web services are software components that are developed by a service provider and made available in a public registry for prospective clients to discover and use over the Internet. Examples of Web services are stock quotes, sports scores, credit card verification, and so on.

At the core of Web services is the Internet infrastructure, XML technologies, HTTP, Simple Object Access Protocol (SOAP), and other protocols. Service providers, service brokers, and clients communicate by exchanging SOAP messages with appropriate XML payload. The various elements of Web services and the corresponding applicable standards are shown in Table 1.

The Web Services Flow Language (WSFL)

TABLE 1: ELEMENTS OF WEB SERVICES

Elements of Web Services	Applicable Standards
Service Description	WSDL (Web Service Description Language)
Service Publication	UDDI (Universal Description, Discovery and Integration)
Service Discovery	UDDI
Integration with Business Processes	WSFL (Web Service Flow Language)
Service Invocation	SOAP, XML
IT Infrastructure	Internet, HTTP
XML Payload	Industry specific

is an XML language for describing how individual Web services can be composed as part of a business workflow. UDDI is used for defining a structure for publishing and discovering information about business and the services they provide. The WSDL description makes it possible for a client to invoke a service after it's been discovered. Security, Management and Quality of Service aspects span the entire service stack.

In addition to the service providers, consumers, and service brokers, standards organizations, technology partners, and IT infrastructure vendors form the backbone of the Web services space.

The standards organizations – W3C, OASIS, OMG – and other industry leaders, such as IBM, Microsoft, and SUN, work on defining and publishing open standards for the emerging technologies. This long list of available and working drafts of specifications is a testimony to the monumental effort made possible only through the visionary and concerted efforts of various standards bodies, committees with active participation from technology vendors, industry experts, consumer groups, etc. The standards fall all over the spectrum of the Web services stack and industry-specific areas.

The technology partners typically provide the necessary tools and architectures for distributed service management and monitoring, distributed security models, XML engines, service frameworks and architectures, integration with business processes, modeling tools, development tools, and services enablers.

The IT infrastructure vendor provides the hardware and software infrastructure such as networks, communications, servers, systems integration, and everything that makes the Web services run (see Table 2). Enterprise issues such as scalability, availability, throughput, latency, bandwidth, clustering, hardware management, and network topology are factors in deploying Web services.

TABLE 2: Roles of participants in Web services

Participant	Roles and responsibilities
Service Provider	Develop, deploy, publish, and provide Web, service software components
Service Broker	Provide listing services for published Web services
Consumer	Find, Bind (connect to service), and Use Web services
Technology Partner	Develop necessary hardware/software technologies
IT Infrastructure Vendor	Provide necessary IT infrastructure
Standards Organizations	Define and communicate standards W3C, OMG etc.

It's not difficult to see that an entity can participate in multiple roles to take advantage of the multitude of opportunities provided by this technology.

## Why Web Services?

Web services is based on technologies and open standards coming together to provide a common good for doing business using the Internet. There's a lot of excitement and momentum in the industry because of the value perceived in making applications and services available in a quicker and better way. In the world of the Internet, it appears to be a natural progression for taking computing using the World Wide Web to the next level. The arguments for using Web services include:

- **Availability:** Uses existing Internet technologies to use and deploy services
- **Transparency:** Services can be located anywhere and used from any where.
- **Encapsulation:** Implementation details of the services are well hidden.
- **Platform-independent:** Open technologies that aren't tied to a particular vendor.
- **Standards based:** A variety of standards address almost every aspect of services
- **Interoperability:** Open and platform-independent standard-based implementations pave the way for interoperable services.
- **Support:** Well supported by industry leaders, vendors, and standards committees

## Current Limitations/Issues

Participants in this space are faced with a number of limitations and issues due to the evolving nature of the technologies and standards that are continually being addressed and resolved. Here's a look at some general issues:

# to adopting Web Services





## “The standards fall all over the spectrum of the Web services stack and industry-specific areas”

### Standards

Evolving specifications are in various stages of finalization and approvals. At the time of this writing the current standards are:

- **UDDI 2.0 specifications from UDDI.org:** Supported by IBM, Microsoft, HP, Sun
- **SOAP 1.2 working drafts just published:** SOAP 1.1 W3C approved
- **WSDL 1.1 submitted as W3C note**
- **WSFL 1.0 from IBM** competing with XLANG from Microsoft
- **EbXML 1.04 Technical Architecture** Specification approved by OASIS

Open-standard specifications are needed across the board in the Web services stack. Providing tools that support these ever-evolving and, at times, competing standards can be difficult.

- **Limited communication protocols:** SOAP 1.0 was limited to HTTP protocols only. SOAP 1.1 included HTTP (S), JMS, and SMTP. Including more protocols enhances the ability to expose more legacy applications as Web services because of increased connectivity. After all, one of the main themes of Web services is to make as many applications available as possible without having to do a major rewrite. In addition, some vendor implementations of SOAP 1.1 may not be interoperable, resulting in services that won't work properly.
- **SOAP object activation and life cycle management raise issues:** When should a service clean up its resources? How long should the objects be kept around in anticipation of the next service call? Object life cycle management can have serious performance implications.

### Quality of Service

- **Interoperability and interchangeability:** While the consumer may choose between a number of service providers, the practical issue is whether service A1 from provider A can be used interchangeably with service B2 from provider B with the same results and the same quality. How does a service aggregator work with two different service providers? How do you convey the semantics of a service? Dynamic discovery is not yet available.
- **Transactions:** The transactional nature of a service, such as database updates, hasn't been addressed. Can you roll back a service after the results are returned to the client? Is there support for distributed transactions?
- **Performance:** Response time can be a key factor in determining the usefulness of a service.

### Security

Sharing of information between the service provider, broker, and consumer in an open environment in a loosely coupled manner over a period of time can lead to additional security risks. Single-signon, establishing secure data services by brokers, and runtime protection security issues must be addressed. How do you define and communicate method-level security?

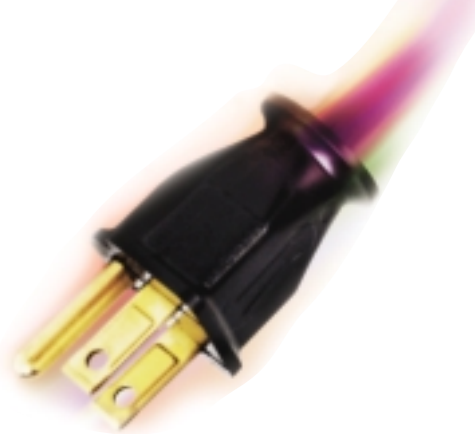
### Management

- **Service management** This becomes complex as services are composed of other services that are geographically distributed. How do we test such a complex system?
- **Payments and escrow services:** How much and for how long do you charge for a service? How do you maintain and track use? How do you specify escrow provisions?
- **Business Integration:** The impact on current business processes needs to be analyzed.

### Methodology For Adoption

Web services offer another channel for delivering quality products and services over the Internet. Perhaps the biggest question from a service provider's perspective is how to take an existing application or write a new one and turn into a Web service. One general approach for a Web services technology strategy contains three steps:

- **Understand the technology for what it is worth:** Conduct research using industry expert reports such as Gartner, Tech Metrix, and so on. Know what the technology is about, what it can do, and where it will be in both the short and long term.
- **Evaluate the technology for use in your space:** Develop a prototype and architect a



solution around it. Look at how it impacts your business processes. Evaluate and estimate project costs. Identify the value-add or issues in using it:

- Time to market
- Product development costs
- Risks
- XML protocols specific to the domain (e.g., banking, health care, etc.)

Look into existing Web services.

- **Adopt the technology:** Once the technology is understood and you've ensured that it's viable, the next step is to incorporate it into a product or service offering. Deploy it internally before making it public.

Now let's look at adapting Web services from a service provider's perspective. We'll focus on building robust applications suitable for services by incorporating feedback from the deployed Web services as well as information gathered from researching Web services external to the organization.



## “At the core of Web services is the Internet infrastructure, XML technologies, HTTP, Simple Object Access Protocol (SOAP), and other protocols”

# **Altova**

**[www.altova.com](http://www.altova.com)**



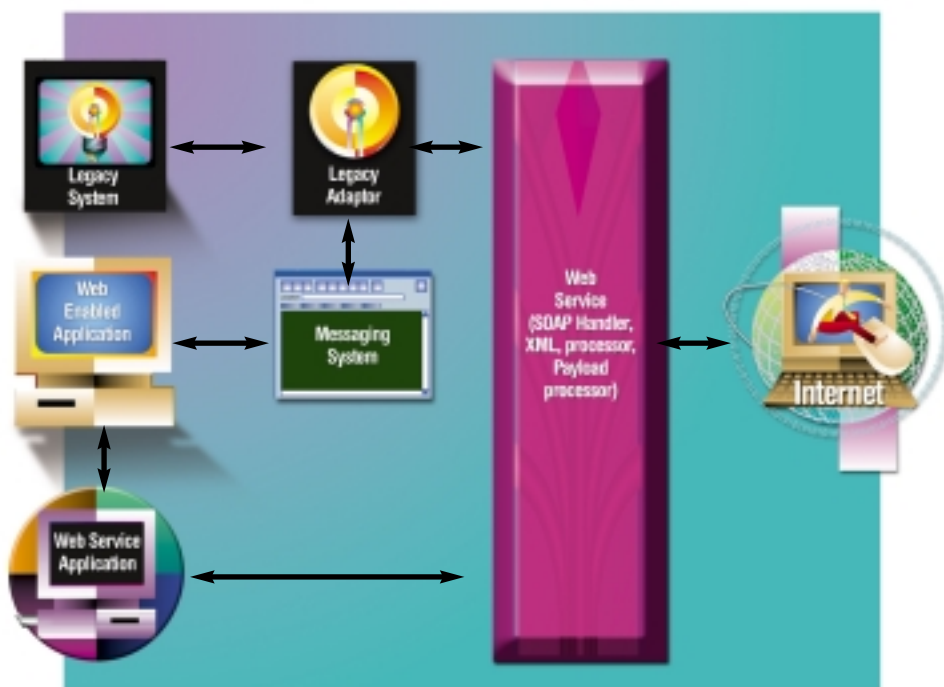


FIGURE 1 | A skeletal framework for enabling applications for Web services

### Step 1: Build Robust Applications

Whether they are Web services or not, it's imperative that you build robust applications for your core business competency. High-quality applications are good candidates for Web services. The choice of technology (J2EE, .NET, etc.) for developing the applications is really a business decision. One of the biggest advantages in Web services is that the service implementation logic and the language of implementation are all transparent to the user of the service. Wrappers are needed to expose these applications.

### Step 2: Identify Applications That Are Suitable for Web Services

A quick look at public registries reveals that a number of the available services are trivial, such as a calculator or temperature converter. The general guideline should be that the service offer some tangible value to the business. The following considerations can help you decide whether a particular application should be enabled under the Web services framework:

- **Service granularity:** Is the package too much, too little, or too soon?
- **Cost of enabling the service:** Web applications generally need less integration work compared to legacy applications.
- **Manageability:** Deployment, code maintenance complexity, and support structure
- **Security:** To whom and for how long can the service be made available? What are the

security policies and how do you define and enforce them for isolation, authorization, authentication, auditing, and monitoring? Are there any method level/component level security restrictions?

- **Transaction:** Read-only service versus real-time database updates? Do you need to roll back?
- **Availability:** What's the impact on existing systems? How will it affect the current system load?
- **Dependencies:** What are the third-party or other critical-system dependencies for this application?
- **Publishing criteria:** Choose between a public free registry, public fee-based registry, or internal registry.
- **Fees and licensing:** How will you charge and receive payments?
- **Current Internet infrastructure**

Incorporate feedback from already published services and research from external services into your plans.

### Step 3: Enable Applications as Web Services

This step involves developing wrappers and adapters for existing applications, followed by deployment, publishing, managing, and monitoring the services (see Figure 1).

### Development and deployment

Applications that need to be enabled fall into three general categories: legacy systems, Web applications, and new applications written for Web services. The most important aspect of enabling is architecting a solution that contains a Web services layer to handle the incoming and outgoing SOAP/XML messages. This layer may also handle XML parsing, translation, XML message writing, and so on. Processing the results returned from the remote procedure call (RPC) or packaging an RPC call can be handled here as well. Business functions, such as interpreting results or packaging the service call, can be deferred to a lower-level module that provides the integration.

A variety of packaged solutions and product offerings, such as add-ons to application servers, standalone Web service containers, toolkits, and so on are available from vendors. Choose one that fits your current technology platform.

Just like any other development project, developing for Web services requires specialized skills. A number of vendors provide tools and toolkits to hide some of the complexities of these protocols to facilitate development and deployment. The following are some of what you'll

“Sharing of information between the service provider, broker, and consumer in an open environment in a loosely coupled manner over a period of time can lead to additional security risks”

# **Sams Publishing**

**[www.sampublishing.com](http://www.sampublishing.com)**



# “ Web service software components are developed by a service provider and made available in a public registry for prospective clients to discover and use over the Internet”

need for consuming or developing Web services.

- **Development Environments:** IDEs, toolkits, frameworks
- **Web skills:** J2EE, JSP, .NET, ASP
- **Web services skills:** SOAP, XML, WSDL, UDDI, ebXML, domain-specific XML schemas
- **Web-enabling skills:** Messaging systems, EAI, distributed computing, systems management, integration with business processes and workflows

After you build and test the service layer and adapters, the services are deployed.

## Publishing

A deployed service is now available for publishing to a registry. Use a vendor provided tool to publish to a known registry.

## Managing and Monitoring

Plan to monitor usage of your Web services. Update the registry with any changes to the service or withdrawal of service. If your service uses other services, then the deployment and rollout plan should include a risk-management plan such as use of alternate registries. The plan might include removing the service from the registry either temporarily or permanently.

## Step 4. Seek Out Relevant and Useful Web Services

One nice way to learn about Web services is to look into an existing Web service. Be your own critic and try out your own published service or one that you might want to package. Check out the registries periodically. Architect solutions to consume XML data received from Web services and integrate with legacy or new applications. Provide feedback into product development.

## Step 5. Use Those Web Services Internally or to Develop Others

Do a proof-of-concept use. Build aggregate services as a value-add to core competency. Provide feedback into Step 2 for evaluating candidates for future Web services or feedback into Step 1 for developing robust applications.

This methodology can also be used, with minor modifications, during the evaluation phase. Service consumers and aggregators can benefit from this as well.

## What It Takes To Participate

It's conceivable that a single organization might have several roles in the ecosystem of the Web services world. Following is a look at what is needed to take part in each role. As you'll see, some of the requirements overlap into several areas.

### As a consumer of Web services:

- **Develop technical skills**
- **Be aware of quality of service issues** such as response time, lack of flexibility of the service, remote procedure versus local call, security, robustness, granularity of the service, etc.
- **Monitor service usage costs** and quality of brokers
- **Participate in standards development**
- **Continuously explore** for quality relevant services
- **Monitor usage of services**
- **Maintain service versions** and withdrawal of services
- **Monitor security policies**
- **Participate in standards development**
- **Build wrappers** to existing applications to enable Web services

### As a service broker:

- **Set up and maintain** free or fee-based UDDI servers and registries
- **Develop technical skills**
- **Self-impose quality control checks** on invalid entries in the repository
- **Implement security policies** on data access as needed
- **Provide or enable electronic payments** for registry use
- **Participate in standards development**

- **Seek creative avenues** to offset the costs of maintaining the registries
- **Make repositories easy to use**
- As a technology partner :**
- **Research and develop technology foundations** in the areas of distributed service management and monitoring, distributed security models, XML engines, service frameworks and architectures, integration with business processes, modeling tools, IDEs, and service enablers such as adapters, wrappers, connectors, client integration etc.
- **Work with IT infrastructure vendors** to make the technology viable
- **Participate in standards development**

### As an IT infrastructure vendor:

- **Develop hardware and software** infrastructure products and services for developing, deploying, and delivering Web services
- **Provide hardware and software infrastructure**, such as networks, communications, servers, systems integration, and everything that makes the Web services run.
- **Provide tools and solutions to enterprise issues:** Scalability, availability, throughput, latency, bandwidth, clustering, hardware management, and network topology


### As a standards organization:

- **Evolve XML-based standards** by working with other standards organizations
- **Promote participation** and use of standards

## Conclusion

Web services are based on proven Internet infrastructure, XML technology, HTTP, SOAP, UDDI, WSDL, and other industry-standard protocols. They provide service encapsulation and location transparency and are easy to publish and access. Web services makes a very promising and useful technology as it provides a new channel for making products and services available more quickly than ever before.

## References

- <http://dcb.sun.com/practices/webservices/>
- [www-4.ibm.com/software/solutions/web\\_services/](http://www-4.ibm.com/software/solutions/web_services/)
- [www.w3.org/](http://www.w3.org/)
- [www.oasis-open.org](http://www.oasis-open.org)
- [www.ebxml.org/](http://www.ebxml.org/) 





THE LARGEST INTERNATIONAL

# XML

## CONFERENCE & EXPO IN THE WORLD!

**WIN A  
\$35,000  
LUXURY CAR!**



ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A \$35,000 LUXURY CAR!

## XML-NEXT G OF ENTERPRISE DEPLOYMENT

### REGISTER ONLINE TODAY

FOR LOWEST CONFERENCE RATES  
EARLY SELL-OUT GUARANTEED!

VISIT [WWW.SYS-CON.COM](http://WWW.SYS-CON.COM)

### Focus on XML

XML is today's essential technology to develop and deploy Web services. Here's your chance to learn from expert practitioners how XML is making the next phase of the Web a reality.

Focus on standards, interoperability, content management, and today's new quest for more efficient and cost-effective Internet and intranet-enabled business process integration.

Focus on XML during information-packed days while you enjoy an XML/Web Services keynote panel, comprehensive conference sessions, and an unparalleled opportunity to exchange ideas with peers and industry leaders.



### A Sampling of XML-Focused Sessions

- XML STANDARDS - AN OVERVIEW
- OASIS STANDARDS UPDATE
- UBL - A UNIVERSAL BUSINESS LANGUAGE
- BRINGING XML TO PKI
- LINK MANAGEMENT WITH XLINK
- PRACTICAL XSLT AND XPATH
- ENTERPRISE CONTENT MANAGEMENT WITH XML
- XML IN THE ENTERPRISE AND INTER-ENTERPRISE WORLD



**NORBERT MIGULA**  
XML CHAIR  
BOARD OF DIRECTORS, OASIS  
INDUSTRY EDITOR  
WEB SERVICES JOURNAL

### Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 I-TECHNOLOGY PROFESSIONALS

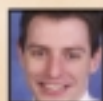
### Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- I-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...



**JEAN FRANCIS ABRAMATIS**  
SENIOR VP R&D, FORMER  
CHAIRMAN, NDC • ELOG



**TYLER JEWELL**  
PRINCIPAL TECH  
EVANGELIST • BEA



**DAVID LITWAK**  
CEO  
SICHERSTREAM



**ANNE THOMAS MANES**  
CTO  
SYSTEMET



**BARRY MORRIS**  
CEO  
IONA



**RICK ROSS**  
FOUNDER  
JAXA LOBBY



**PATRICIA SEYBOLD**  
FOUNDER & CEO  
SEYBOLD

### For Exhibit information

CONTACT: RICHARD ANDERSON  
125 CHESTNUT RIDGE RD.  
MONTVALE, NJ 07045  
201-832-3054  
RANDAR@SYS-CON.COM

**XML**EDGE  
conference & expo

**JUNE 24-27**

JACOB JAVITS  
CONVENTION CENTER  
NEW YORK, NY

**OCTOBER 1-3**

SAN JOSE  
CONVENTION CENTER  
SAN JOSE, CA

### SPONSORED BY:

IONA | END 2 ANYWHERE™

ADOS

bea

TogetherSoft

SilverStream

Actional

MERANT

Altaworks

PolarLake  
Enterprise Strength XML

ALTOVA

### MEDIA SPONSORS

Federal Computer Week

WebServices.org

XML TIMES.com

Java Skyline

CE Advisor

WebServicesMail

WIRE

XML-ORG

WROX

XML-ORG

SDTimes

JAVA

wireless

XML

WebLogic

WebServices

WebSphere

COLDFUSION

OWNED AND PRODUCED BY

SYS-CON MEDIA | SYS-CON EVENTS

**AUTHOR BIO:**

Tom Tuduc, principal of WebArchives, was a member of IBM Scientific Labs and Boeing's management staff. He has worked in Internet infrastructure technologies including Web services, SCM, eMarketPlaces, and business intelligence. [TOMTUDUC@WEBARCHIVES.COM](mailto:TOMTUDUC@WEBARCHIVES.COM)

# Driving Commerce Value Systems with the Web Services Paradigm

## Web Services on the Move

written by Tom Tuduc

**A** maturing generation of computing technologies is creating a service framework that enables enterprises to adapt to the business climate in real time, to improve their functions, reduce cost of development, and generate new value; this propels the entire commerce value system into a higher gear of economic reasoning.

These technologies center on Web services, which are standardized software components that can be programmatically invoked over the Internet. More significant in the long run, behind the production of Web services-based applications, are the open standards, architectures, confluent technologies, platforms, development methodology, and services providers.

This article first describes the concepts in the Web Services Paradigm (WSP). It then expands on how WSP drives commerce value systems by enhancing the structure of inter/intrabusiness processes and applications; going beyond operations to the center of business strategy, providing a business services framework to implement service tradeoffs; and providing an overview of the coming real-time commerce webs, also known as dynamic commerce webs.

### Under the Hood: What Is WSP?

WSP embodies four IT industry concepts (see Figure 1):

**1. Web Services Architecture (WSA):** Web services are software components that can be programmatically invoked by other software components using canonical XML formats over the Internet in a loosely coupled architecture with network protocols such as HTTP or SMTP. WSA has five layers, along with respective standards: network (HTTP/S), messaging (XML, SOAP), description (WSDL), infrastructure (services), and applications (BPM). Different Web services value providers and consumers will be interested in different layers. For example, while human users mostly interact with Web services within the application layer where services are delivered, systems and devices interact with Web services simultaneously.

A canonical Web service, regardless of what programming language implements it, uses the extensible Simple Object Access



(SOAP), and is registered and exposed using the Web Services Description Language (WSDL). Note that at the network level, using persistence technologies, Web services can interact with applications in real-time and overcome the stateless and reliability problems inherent in http/https.

**2. Service-oriented Architecture (SOA):** SOA provides the business services framework in which services are created and consumed. SOA harnesses the confluence of computing concepts, including networking, software engineering, software architectural styles, object-oriented technologies and business process management (BPM, which here includes process orchestration, execution, monitoring, and administration).

Deploying inside SOA are software development and business process engineering methodologies needed for the development, registration, classification, and deployment of Web services. These applications and process development methodologies play out mainly at the top two layers of WSA: the infrastructure and the application layers. These two layers share functionalities and service grids that use proposed standards such as Universal Description, Discovery, and Integration (UDDI), and ebXML. For transaction standards, SOAP-CTX is in the works.

**3. Integrated Systems Environments (ISEs):** ISEs provide platforms for development with functionalities such as the creation, assembly, linkage, description, registration, discovery and simulation of Web services, data mapping, messaging of loosely coupled events, and dynamic bindings of services at runtime. In the immediate future, ISEs will expose existing applications and components such as EJB, COM+, CORBA, CICS, SAP, People Soft, and Siebel.

**4. Web Services Value Chain:** This is made up of value providers and Web services consumers. The various classes of consumers include businesses, the general public, systems, devices and appliances, digital dashboards, mobile devices, smart cards, portals, industrial applications, moving vehicles, and content and service aggregators. The value providers include new and reinvented IT functions such as business solution provider and process architect, independent broker/evaluator, and infrastructure provider/architect.

### Moving Forward

Registered Web services evolve from simple to business class to include contracts, contractual personalization, itineraries, service level agreements, nonrepudiation transactions, and digital signatures. Furthermore, to leverage the object-oriented paradigm, Web services will exhibit composite functionalities, "kind-of"

inheritance features, and polymorphism. For example, business applications components can be grouped as related Web services with concept/subconcept relationships. To provide naming simplicity, the same Web services name can behave differently depending on context (polymorphism).

### The Benefits

WSP provides openness, canonization and standardization of processes, applications, and services. Modularity can be achieved in both functionality and representation, resulting in cross-platform usability. On top of this, the ubiquity, simplicity, and accessibility of the Internet provide the network-effect value to consumers and businesses of all sizes.

## Architecting Dynamic Commerce Value Systems

Internet e-commerce has been puttering along with the "World Wide Wait" stuck in first gear. Now, WSP kicks it into fifth. For all industries involving information, starting with the IT industry itself, WSP transforms both the structure and nature of linkages of inter/intrabusiness processes by fluidly architecting dynamic business models and real-time business applications with real-time BPM capable of quickly adapting to the unpredictable business climate.

More specifically, WSP enables enterprises to orchestrate business processes seamlessly inside and across company boundaries by dynamically combining elements from interactive models such as portal, dashboard, cockpit, and business web models, including Agora, aggregation, value chain, alliance, and distributive network. This kind of orchestration creates new composite and fluid business models leading to dynamic commerce webs spanning all suppliers, distribution channels, and customers across all industries. For day-to-day operations, this leads to the real-time implementation of business strategy across all phases of market adoption of products and services, whether it's the creation of disruptive innovations, shortening time-to-market, enabling operational excellence, or facilitating continuous enhancements. The result: real-time business applications (see Figure 2).

## A More Powerful Engine: WSP and the Structure of Business Processes

WSP improves enterprises' operations by changing the structure of business processes and applications. While the structure of intrabusiness processes refers to linked activities within an enterprise, including CRM, SCM,

marketing, product development, and ERP, the interbusiness structure refers to linked activities across enterprises. WSP improves the structure of business processes and reduces cost of development by adopting WSA and SOA (see sections 2 and 3 of the figure).

### How It Works

The process orchestration in WSA changes the architecting of business processes as it brings business architects and software engineers closer together to implement business applications. BPM with Web services provides fluid orchestration, including nested process composition from high to low, or coarse- to fine-grain levels of process abstraction. For example, Biztalk Orchestration Designer shows business process flowcharts side by side with implementation components and connects the two worlds by various wizards.

Underlying WSA and SOA are the following behaviors that are crucial to the architecting of industries' dynamic value systems and value chains:

1. The decoupling of business logics as they are modeled and implemented as coarse- and fine-grain Web services and as declarative and explicit business process templates and schemas in canonical XML formats
2. The exposing of applications and data, both new and legacy, as Web services, to registries and repositories
3. The translation between syntactics, i.e., XML dialects and EDI
4. The frictionless orchestration, composition, assembly, and integration of Web services by linking them together statically or dynamically at runtime instead of writing custom "glue code"

WSP also complements the improvements to BPM by enhancing business-process modeling, analysis, simulation, and process im-



FIGURE 1 Real-time business applications: architectural view



FIGURE 2 Real-time business applications: process view

provement since update and maintenance of process flows and components can be done at different levels of granularity, across platforms, and with ease. The Web services components, regardless of what language they are written in and how often they are modified, are exposed as Web services in registries and repositories ready to be personalized and linked together by real-time business process flows.

Efficient supply-chain solutions lie not with frictionless technologies but with federated planning among enterprises sharing objectives and insights. WSP takes this to a different playing field by creating new synergies between frictionless technologies and federated planning by enabling new compositions of business and value and supply chain models, no matter how unconventional or unprecedented the business and technology requirements are, and the aligning of business objectives among partners to define and manage the value chain using explicit rules and policies that are themselves exposable Web services.

Enterprises can optimize their value chains to specific objectives by assembling and linking components and frameworks of Web services. The decoupled business process model and WSP provide increasing layers of abstraction, allowing increasing layers of control touch points, which in turn allows executives and specialists to communicate in real time with precision. The linking makes possible what-if optimizations of process flows, real-time changing of business rules, and accommodation of cost-inhibitive flow exceptions.

Ring true in supply chain technology, decoupling the business process is the key to balancing the divergent requirements of functional efficiencies on one hand, and agility in sourcing resources, as well as customer/supplier-oriented functions, on the other. WSP provides the keys to the systematic decoupling and the ease of orchestrating to different tunes and constraints.

Ease-of-service integration using formal business process flow languages such as ebXML, Microsoft's Biztalk/XLANG, IBM's WSFL, or Vitria's VCML, will blur the boundaries of supply, customer, and channel relationship management and tighten the feedback loops in value systems. In addition, real-time product developments across organizations and in collaborative settings involving direct and indirect materials can be realized by linking together Web services.

### Test Drives: WSP and Business Processes

**Web services at Dell:** Dell uses the Internet and Web services for touch points across value chain activities to give direct interactions. For example, customers can buy, make changes, view inventory, and track every step of the process online and in real time. In addition to the outsourcing principle, Dell heads toward a real-time joint-managed supply chain and manufacturing to reduce cost. To reduce inventory from weeks to hours at various nodes in the value chain, Dell uses Web services on its extranet to publish manufacturing data in real time. For e-procurement, Dell uses Web services and XML standardizing data and processes. The result is impressive: Dell saves a business customer \$4 million per year.

**Creativity at work:** Web services enable DigitalWork to aggregate over 30 different Web services, such as Dun & Bradstreet's credit reporting, billing logistics, and payment processing, and provide them for brick-and-mortar Web sites such as Costco, Home-Depot, and Mail Boxes Etc., which then offer these composite services to small business and consumers.

### Pushing Productivity

Operationally, Web services push the productivity frontier outward by enhancing the efficiency of information retrieval and processing. Full-fledged Web services enable business users to create process flow and business applications and communicate in tighter feedback loops with software designers. Before long, business users can create applications with "off-the-repositories" Web services.

Ease-of-service integration creates the mass customization of Web services. The costly best-of-breed integration with glue code will now be replaced by linking Web services together. The marriage of BPM and the component paradigm finally blossoms as business processes are encoded in XML, transmitted over the network using XML formats (i.e., SOAP), combined with other processes, and executed by state machines specified by an XML Schema in a dialect such as XAML.

The mass customization of information

technology is fueled by a culture whose time is ripe: reuse, which is simpler and from distributed sources. Every business process can be reused, including the executing/transaction service engine itself.

Organizations will define how and what Web services can be aggregated or delegated among participating members to create a new kind of dynamic value chain management, including real-time synchronization of supply and demand, collaborative planning, and joint-managed inventory, bringing unprecedented efficiency to the supply chains and benefiting all value systems.

For example, tools such as Microsoft's Biztalk or Vitria's BusinessWare provide functionalities for orchestrating, managing, and monitoring business processes. Bowstreet's Web Factory can be used with IBM's WebSphere or BEA's WebLogic to dynamically assemble J2EE components and Web services from back-end systems to external Web services using visual point-and-click to create customized end-to-end applications.

### The Result

As the structure of a business process is more fluid, enterprises can dynamically link and repurpose decoupled business logics to implement dynamic business strategy, which includes value propositions, product designs, partnerships, channels, sales, pricing, and marketing strategies. Specifically, WSP enhances:

1. Collaborative development environments with open and easily integrated components
2. Opportunistic one-time and ad-hoc services
3. Implementation of "point" solutions from an increasing availability of registered components
4. Integrated solution packages
5. Verticalized frameworks
6. Service-usage statistics enabling real-time process improvement
7. Mass availability and customization of analytics services

### Drawing Up the Roadmap: WSP and Business Strategy

Strategy defines how all the elements of business work together. WSP is at the center of business strategy by deciding what business processes are linked and how they are linked. For example, business processes can be linked at compile time or at runtime, or be substituted by other processes (first-order systems) or by a composite framework of processes (second-order systems).

WSP gives enterprises more positioning strategies as customization of services is more efficient. WSP catalyzes ease of product cre-



THE LARGEST INTERNATIONAL

# WEB SERVICES CONFERENCE & EXPO IN THE WORLD!

**WIN A  
\$35,000  
LUXURY CAR!**



ATTENDEES WILL BE LIMITED TO TAKE A GOLF SHOT TO WIN AND RIDE OFF IN A \$35,000 LUXURY CAR!

## WEB SERVICES SKILLS, STRATEGY, AND VISION

**REGISTER ONLINE TODAY**

**FOR LOWEST CONFERENCE RATES  
EARLY SELL-OUT GUARANTEED!**

**VISIT [WWW.SYS-CON.COM](http://WWW.SYS-CON.COM)**

### Focus on Web Services

Web Services, the next generation technology that will enable the Internet to work for you and your business, and finally provide that ROI you have been after, will be put under a microscope at Web Services Edge East 2002.

Information-packed sessions, exhibits, and tutorials will examine Web Services from every angle and will provide cutting-edge solutions and a glimpse at current and future implementations. Hear from the innovators and thought leaders in Web Services. Enjoy a highly interactive CEO Keynote panel that will debate and discuss the realities and promise of Web Services.



### A Sampling of Web Services-Focused Sessions

- PRACTICAL EXPERIENCES WITH WEB SERVICES AND J2EE
- STATE OF THE WEB SERVICES INDUSTRY
- THE ODD COUPLE: MAKING .NET AND J2EE WORK TOGETHER
- EXPLORING THE .NET MY SERVICES INITIATIVE
- STANDARDS WATCH
- GUARDING THE CASTLE: SECURITY & WEB SERVICES



SEAN PADDY  
CONFERENCE TECH CHAIR  
WEB SERVICES TRACK CHAIR  
EDITOR-IN-CHIEF  
WEB SERVICES JOURNAL

### Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 J-TECHNOLOGY PROFESSIONALS

### Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- J-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

**Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...**



JEAN-FRANÇOIS GAGNON  
SENIOR VP R&D, FORMER  
CHAIRMAN, W3C • ILOG



DAVE CHAPPELL  
VP CHIEF TECHNOLOGY OFFICER  
SONIC SOFTWARE



GREGG KIESSLING  
CEO  
SITRAKA



ANNE THOMAS MANES  
CTO  
SYSTEMET



BARRY MORRIS  
CEO  
IONA



ERIC RUDDER  
SENIOR VP DEVELOPER  
AND PLATFORM EVANGELISM  
IN MICROSOFT



PATRICIA SEYBOLD  
FOUNDER & CEO  
SEYBOLD

### For Exhibit Information

CONTACT: MICHAEL PESACK  
125 CHESTNUT RIDGE RD.  
MONTVILLE, NJ 08045  
201 882-7067  
MICHAEL@SYS-CON.COM

web services **EDGE**  
conference & expo

**JUNE 24-27**

JACOB JAVITS  
CONVENTION CENTER  
NEW YORK, NY

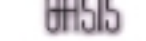
**OCTOBER 1-3**

SAN JOSE  
CONVENTION CENTER  
SAN JOSE, CA

### SPONSORED BY:



### MEDIA SPONSORS



### OWNED AND PRODUCED BY





FIGURE 3 Real-time business applications: value-systems view

ation, and in turn increases business creativity. Tradeoffs, which arise from business processes, product configurations, core competencies, and human resources, need to be reevaluated as inefficiencies from customization decrease.

#### Using the Roadmap: Creating Industry-Specific Real-time Applications

Nearly all industries' activities involve the creation, processing, and communication of data and processes. Using WSP and industry-specific XML, software solution providers can translate service creativity into executable processes that dynamically assemble available services. For example, a future service can integrate using process flow languages, internal core services with services of external organizations, continual or one-time services, and other personalization. A customer can create in real-time a tailored service using fine-grain components and coarse-grain frameworks in

verticals such as financial, legal, or health care. These services include functionalities such as preliminary assessments over the Web to find out what other services and class of services are further needed; searching and locating providers in the preferred geographical areas; scheduling and coordinating of appointments; and handling and routing of necessary documents, information, and forms.

#### Strategy

Strategy has to reflect how WSP enables value systems to be architected, as well as optimized, in real-time, involving customers, product designers, process architects, value chain analysts, and suppliers. For example, a customer can specify desktop workstation specifications at one end of the value chain and trigger the automatic linking of fine-grain computer-aided manufacturing Web services to provide just-in-time engineered products and services. Meanwhile, value-chain analysts and process architects, which can be Web services, specify the optimized process flow and parameters for sourcing software agents and auction bots. This is a real-time system approach calling for the strategic selection of analytics and heuristic rules.

Standardization at all levels of the Web service value chain enables not just mass customization, but also carries the duplication of enterprises' strategies and activities to the next stage of IT evolution. A document sent by one Web service to another can encode all the data, processes, and contexts enabling a re-creation of a whole business-process state machine ready to plug into participating supply chains. The result isn't just the enabling of a paperless factory, or a paperless office, but also the enabling of the duplication of the enterprise's functionality and the entire

value chain with deployment on specified hardware/platforms included.

WSP is the engine enabling the real-time behaviors in dynamic value systems to replace the traditional sequential and linear ones so that all mutually dependent participants can create community solutions and federated planning. WSP can cut through layers of value systems, making them behave like frictionless information ecosystems, while combining dynamic business models to provide integrated solutions, ad hoc point solutions, and vertical frameworks.

For example, the real-time health care value system and related information value systems seen in Figure 3 consist of millions of participants from the following categories: the health care industry, software solution providers, Web services solution providers, and Web services platform providers. This value system can include community solutions and federated planning to stabilize and balance the real-time effect of millions of value chain participants across value systems, activating exponentially larger numbers of Web services.

Furthermore, given the complete understanding of the relationships between all participants in the value chain and the chain's constraints, the loosely coupled architecture makes it easier to cut through the layers of interconnected decisions from thousands of interacting nodes to dynamically show hot points. Strategically, real-time optimization doesn't need to respond wildly to every variation in the value chain to accommodate unusual and unpredictable demands. When done appropriately, real-time optimization complements models based on trends and federated strategy.

"Technology changes; economic laws do not," states Carl Shapiro. It's certain that both macro- and microeconomic principles work

**RECEIVE \$150**  
DISCOUNT OFF FULL CONFERENCE  
WEB SERVICES EDGE REGISTRATION

**web services** **EDGE**  
world tour 2002

**Learn How to Develop  
SOAP Web Services NOW!**  
at a One-Day Seminar...Coming to a City Near You!



more efficiently in increasing frictionless dynamic commerce webs. As always, limited resources in value systems dictate enterprises' policies on what services and information can be shared versus made proprietary in a collaborative and federated environment. But only now is federated planning in real-time across value chains and value systems possible. Techniques in operation research, including game theory and control theory, will commingle with those in computer science and software engineering to produce frameworks and infrastructures for Web services, Web services traffic hubs, and BPM involving Web services in order to create more efficient marketplaces and business values.

Strategy will reflect consumers' and businesses' increasing interest in accessing and renting, rather than owning, services because they are continuously improving in real-time. WSP will enhance CRM, catalyzing direct sales and configuration of complex products and services by suppliers and manufacturers to businesses and consumers. In addition, new intermediaries in the Web services value systems will emerge to provide "relationship" services to publishers and requesters, including relationship-based insurance services.

The increasingly frictionless market gives rise not only to new business strategies, but new marketing strategies as well. This opens up a whole new world of customer experiences. For the consumers, renting services creates communities of similar service renters. More relationships will be formed instead of fewer, and new intermediaries will emerge, inventing new value systems.

Information-based value systems increasingly leverage the economics of networks while decreasingly depending on the economics of software component development.

Businesses and consumers inevitably favor Web services linkages and infrastructures with the most traffic, just as they favor communication networks with broad reach. Metcalf's Law will apply to the development and to the utility phases of Web services as more and more enterprises become "componentized" and register their services. Business processes and Web services will become first-class objects enabled to call and execute one another. Integration becomes frictionless, and frictionless marketplaces add fuel to dynamic commerce webs.

### Blue Sky, Open Road

WSP, providing increasingly open and common standards, enhances the egalitarian ecosystems of services. Integration schemes, becoming so elegantly simple yet powerfully utilitarian (and nearly free), give rise to increasingly profound changes in the structure and nature of value systems. As the line between business strategy and information technology continues to blur, WSP will provide the instruments to address the encoding and processing of information and knowledge driving the commerce web.

There will be metaservices providing "goal and constraint" programming services for the real-time configuration and assembly of Web services using business rules. An example of such metaservices would include a second-order configuration capability, possibly involving Prolog, which applies sophisticated business rules to real-time configuring of processes and services both automatically and semiautomatically.

Beyond real-time optimization, the extensible WSP will provide services for reasoning about optimization. Here Web services interact with other Web services acting as economic advisors to determine preferred value chains

and value systems in dynamic commerce webs. This reasoning about optimization will automate how the nature and structure of value systems work in real time, catalyzing even more frictionless dynamic commerce webs. Orders of magnitude more utilitarian value systems and value chains will bring forth increasingly frictionless marketplaces the likes of which Adam Smith himself could only dream of. We're cruising now!

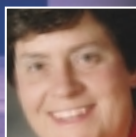
### References

- Glossary: <http://webarches.iteholdings.com/glossary.htm>
- Hagel III, John, John Brown. "Your Next IT Strategy." Harvard Business Review. October 2001.
- Oliver, Keith, Anne Chung, Nick Samanich. "Beyond Utopia: The Realist's Guide to Internet-Enabled Supply Chain Management." Strategy+Business. Second Quarter, 2001. [www.strategy-business.com/press/article/?art=17966&pg=0](http://www.strategy-business.com/press/article/?art=17966&pg=0)
- Porter, Michael. "Strategy and the Internet." Harvard Business Review. March 2001.
- Reddy, Ram. "Chasing Windmills. The paradox of efficiency and agility in supply chain management technology." Intelligent Enterprise. October 24, 2001. [www.intelligententerprise.com/011024/416infosc1\\_1.shtml](http://www.intelligententerprise.com/011024/416infosc1_1.shtml)
- Shapiro, Carl, Hal Varian. *Information Rules*. Harvard Business School Press, 1999.
- Tanner, Michael. "Framework for Strategy." Strategy Brief. Issue 2, September 6, 2001. [www.chasmgroup.com/strategybrief/index.html](http://www.chasmgroup.com/strategybrief/index.html)
- Tapscott, Don. *Business Models in the New Economy*. [www.agilebrain.com/tapscott.html](http://www.agilebrain.com/tapscott.html)
- WebArches. *Web Service Paradigm* <http://webarches.iteholdings.com/WSPTT.htm> ©

# Jump-start your Web Services knowledge. Get ready for Web Services Edge East and West!

AIMED AT THE JAVA DEVELOPER COMMUNITY AND DESIGNED TO EQUIP ATTENDEES WITH ALL THE TOOLS AND INFORMATION TO BEGIN IMMEDIATELY CREATING, DEPLOYING, AND USING WEB SERVICES.

EXPERT PRACTITIONERS TAKING AN APPLIED APPROACH WILL PRESENT TOPICS INCLUDING BASE TECHNOLOGIES SUCH AS SOAP, WSDL, UDDI, AND XML, AND MORE ADVANCED ISSUES SUCH AS SECURITY, EXPOSING LEGACY SYSTEMS, AND REMOTE REFERENCES.

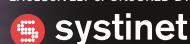


### PRESENTERS...

**Anne Thomas Manes, Systinet CTO**, is a widely recognized industry expert who has published extensively on Web Services and service-based computing. She is a participant on standards development efforts at JCP, W3C, and UDDI, and was recently listed among the Power 100 IT Leaders by Enterprise Systems, which praised her "uncanny ability to apply technology to create new solutions."



**Zdenek Svoboda is a Lead Architect** for Systinet's WASP Web Services platform and has worked for various companies designing and developing Java and XML-based products.  
EXCLUSIVELY SPONSORED BY



Register at [www.sys-con.com](http://www.sys-con.com) or Call 201 802-3069

**BOSTON, MA** (Boston Marriott Newton) . **SOLD OUT!** **JANUARY 29**  
**WASHINGTON, DC** (Tysons Corner Marriott) **SOLD OUT!** **FEBRUARY 26**  
**NEW YORK, NY** (Doubletree Guest Suites)..... **NEW DATE!**.....**APRIL 19**  
**SAN FRANCISCO, CA** (Marriott San Francisco) .....**MAY 13**

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE \$495 REGISTRATION FEE.



# JWS: Web Services in Java

## Making integration easier

**W**hy is it so hard to build applications that integrate enterprise systems? One problem is that the platforms that integrate diverse systems are more difficult to learn than they need to be. Another problem is that even after integrated applications are built and working, the systems are brittle, hard to change, and expensive to maintain.

It should be easier.

This article is about Java Web Services (JWS), a standard under development that simplifies these problems. The JWS standard makes it possible to put together more robust, less brittle, integrated systems using Web services standards, including SOAP and WSDL, yet without requiring you to learn any complicated interfaces. As a result, JWS has broad appeal to sophisticated enterprise Java developers as well as programmers who don't even know Java.

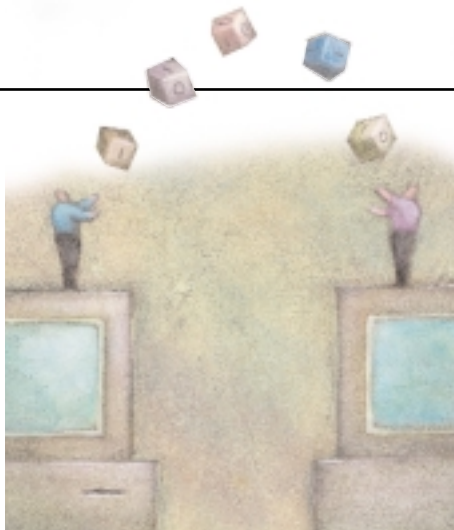
What is JWS? This article introduces the standard, provides several examples demonstrating its capabilities, and discusses how to use JWS to build robust solutions.

### A Simple Example

First, an example:

```
class HelloWorld {
    /** @jws:operation */
    public String hello(String name) {
        return "Hello, " + name;
    }
}
```

This simple program is a complete Web service in JWS. No kidding!



To use it, just save it as a file called HelloWorld.jws in a Web application directory on a JWS-enabled J2EE Web server, just as you would save a JSP page. The JWS server performs the necessary code generation, compilation, and deployment steps to take the JWS file and provide a fully functional Web service. Your Web service is immediately made available at <http://myserver/myapp/HelloWorld.jws> (where myserver and myapp are the names of your server and application). You can then exchange standard SOAP messages with the Web service, and you can use a Web browser to test your Web service with an automatically generated testing UI. Additionally, the JWS server generates a downloadable WSDL contract describing the precise wire format supported by your service, some Java code to be used by clients, and so on.

Modify the JWS file a bit, and the Web service is automatically updated, with new WSDL contracts, the ability to accept new SOAP messages, etc. Building a Web service with JWS is as quick and simple as working with JSP.

What's going on in the "Hello World" example?

A JWS file is an ordinary, compilable Java file with some special Javadoc annotations in the comment blocks. The @jws:operation annotation is not an ordinary comment just for documentation. It tells the JWS container that the hello method is supposed to be an exposed operation of the Web service that should be made accessible via SOAP and described through WSDL. Removing the @jws:operation annotation would remove the method from the external public contract of the Web service, making it a private internal method. Adding @jws:operation to other methods would add those methods to the external contract.

There are about two dozen other @jws: annotations you can use to control aspects of your Web service without manually calling any complicated APIs. These annotations are designed to be easy to edit with visual modeling tools. For example, BEA WebLogic Workshop includes a Web services design tool that represents a Web service graphically. Modifying the graphical diagram in the tool modifies the values of the Javadoc annotations in the underlying JWS file. Also, since the annotations are just Javadoc comments, they are easy to work with in any standard source code editor.

Let's discuss some of these other annotations.

### The Three Principles of Building Robust Web Services

In the first example, you can see how easy it is to expose Java logic as a Web service. So why isn't that all there is to JWS? Why would you need any other annotations beyond @jws: operation?

The answer is that building a robust Web service involves more than just exchanging SOAP messages and WSDL files. Web services technology is designed to allow durable, maintainable integration between systems. But simplistic use of new Web services technology results in fragile integration just as with older integration technologies. To exploit Web services fully, they need to be engineered with the following three principles in mind:



#### Author Bio

David Bau is a program manager at BEA Systems. He is one of the principal designers of JWS and WebLogic Workshop. David previously worked for Crossgain Corporation and Microsoft, and has helped to develop several successful software platforms, including Internet Explorer and ASP.NET.  
DAVID.BAU@BEA.COM



THE LARGEST INDEPENDENT

# JAVA DEVELOPER CONFERENCE IN THE WORLD!

**WIN A  
\$35,000  
LUXURY CAR!**



ATTENDEES WILL BE INVITED TO TAKE A GOLF SHOT TO WIN AND RIDE OFF IN A \$35,000 LUXURY CAR!

**JAVA IN JUNE  
ESPECIALLY IN NEW YORK**

**REGISTER ONLINE TODAY**

**FOR LOWEST CONFERENCE RATES  
EARLY SELL-OUT GUARANTEED!**

**VISIT [WWW.SYS-CON.COM](http://WWW.SYS-CON.COM)**

## A Sampling of Java-Focused Sessions

- JAVA 1.4: WHAT'S NEW?
- BUILDING TRULY PORTABLE J2EE APPLICATIONS
- JAVA TOOLS FOR EXTREME PROGRAMMING
- BUILDING ASYNCHRONOUS APPLICATIONS USING JAVA MESSAGING
- JET VS. J2EE
- J2EE: SETTING UP THE DEVELOPMENT ENVIRONMENT
- BUILDING WEB SERVICES WITH J2EE
- DETECTING, DIAGNOSING, AND OVERCOMING THE FIVE MOST COMMON J2EE APPLICATION PERFORMANCE OBSTACLES



ALAN WILLIAMSON  
JAVA CHAIR - EDITOR-IN-CHIEF  
JAVATODAY JOURNAL

## Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 I-TECHNOLOGY PROFESSIONALS

## Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- I-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

**Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...**



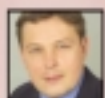
TYLER JEWELL  
PRINCIPAL TECH  
EVANGELIST • BEA



GREGG KIESSLING  
CEO  
SITRAKA



DAVID LITWACK  
CEO  
SILVERSTREAM



BARRY MORRIS  
CEO  
IONA



GREGG O'CONNOR  
PRESIDENT  
SONIC SOFTWARE



RICK ROSS  
FOUNDER  
JAVA LOBBY



JAMES DUNCAN  
DAVIDSON  
FATHER OF ANT

## For Exhibit Information

CONTACT: MICHAEL PESACE  
135 CHESTNUT RIDGE RD.  
MONTVILLE, NJ 07045  
201-932-2017  
MICHAEL@SYS-CON.COM

**JDJEDGE**  
conference & expo

**JUNE 24-27**

JACOB JAVITS  
CONVENTION CENTER  
NEW YORK, NY

**OCTOBER 1-3**

SAN JOSE  
CONVENTION CENTER  
SAN JOSE, CA

## SPONSORED BY:

IONA | END 2 ANYWHERE™

ADOS

bea

TogetherSoft

SilverStream

Actional

MERANT

Altavox

PolarLake  
Enterprise Strength XML

ALTOVA

## MEDIA SPONSORS

Federal Computer Week

WebServices.org

XML TIMES.com

Java Skyline

CE Advisor

WebServicesMail

WIRE

XML.ORG  
Certified Java  
Management Practitioner

WROX

BASIS

SDTimes

JAVA TODAY

wireless

XML JOURNAL

WebLogic

WebServices

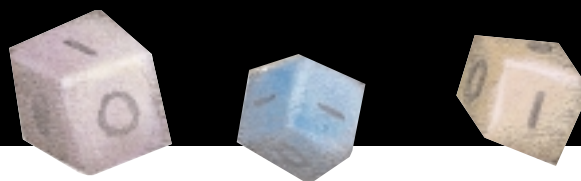
WebSphere

GOLD FUSION

## OWNED AND PRODUCED BY

SYS-CON  
MEDIA

SYS-CON  
EVENTS



1. **Loose coupling:** Interfaces must be separated from implementation.
2. **Asynchronous operation:** Distant systems must not block one another.
3. **Rapid integration:** Relevant data and applications must be easy to connect.

## Loose Coupling

Anybody who has maintained a complex integrated system knows what it's like to be trapped by *tight coupling*: once the information flow between parts of the system becomes too intricate, it becomes nearly impossible – or at least, very expensive and risky – to change any single part of the system. One of the goals of Web services technology is to break the gridlock of tight coupling by providing tools to simplify, clarify, and rationalize the information flow between systems.

Robust, maintainable loose coupling is grounded in two bits of wisdom:

1. Public contracts and private implementations evolve on different schedules.
2. Chunky messages are easier to maintain than fine-grained call sequences.

With automatic WSDL and stub-generation tools (such as the capabilities that are built into basic JWS usage in the first example above), it is actually very easy to fall into the tight-coupling trap of programming public contracts in the same way as private implementations. Since automatic WSDL and stub-generation tools work by directly generating a message shape from a function signature (or vice versa), when you change your implementation code, it changes your public contract at the same time. These tools also make it easy to use numerous, fine-grained messages, but they don't make it as easy to deal with more maintainable coarse-grained messages. So using WSDL generators can be extremely brittle.

When you're concerned about maintainability, you need your tools to help you work with robust, chunky, public message con-

tracts that can continue to work even when your underlying implementation changes.

### Example of Loose Coupling: XML Maps in JWS

Here's an example of loose coupling, using a feature of JWS called XML maps. Suppose you have a public message format that's designed for submitting purchase orders to an ordering system. Since the message format is public, it will have been designed to be coarse-grained, permanent, and evolvable. A sample message might look like the purchase order XML in Listing 1.

Implementation data structures, on the other hand, are tied to the particular details of the program and how it is implemented. For example, our implementation may track customers using only the customer number, not the name, and it may track ordered items using the catalog number, not the description. We might not even have any single data structure representing a purchase order; instead, we may just pass around arrays of line items.

Listing 2 shows a simple JWS example that adapts between our public purchase order contract and our private purchase order implementation.

The Javadoc comment annotating the `submitOrder` method in Listing 2 includes the following code:

```
* @jws:parameter-xml xml-map::
* <purchaseOrder>
*   <customerInformation>
*
<customerNumber>{customerNo}</customer
Number>
```

(In the full listing the rest of the XML is available.) The XML message inside the annotation is an XML map. It is a picture of the relevant parts of the public input message, and it contains curly braces that correspond to places in the XML message where data should be bound to a Java

parameter or field. As you can see, when you use an XML map in `@jws:parameter-xml`, you can see and control the public message format that is consumed by our private implementation.

By allowing you to make the message format explicit, XML maps provide a key advantage over the use of traditional WSDL generation or proxy generation techniques. With ordinary WSDL generation, you don't control the public message format explicitly; instead, when you need to change your implementation signatures, the message format is automatically changed by the system. If your WSDL generator happens to choose a message format in a way that breaks compatibility, then it will have broken your integration.

Additionally, XML maps make it easy to evolve either the implementation or the public contract without breaking the system. For example, if we decide to enhance the Java implementation by validating prices and descriptions of catalog items, we could add additional fields to our private `LineItems` data structure in Java and add additional lines to the input XML map to extract information from the `<price>` and `<description>` tags that are present in the public message. Or if we want to evolve the public contract by adding additional tags in the `<customerInformation>` section, we could do that without perturbing the existing implementations that don't need to use the extra information. XML maps allow the public contract to evolve on a different schedule from the private implementation.

XML maps have other features that we haven't shown here: they can be used for output as well as input, and they can deal with more complicated data structures or even invoke procedural code to do nontrivial data transformations when needed.

XML maps in JWS make it very easy to implement loosely coupled systems. They are a simple tool that maintains and maps between a separate public Web services contract and private implementation.

## Asynchronous Operation

When integrating high-volume systems, your goal is to make the most of available bandwidth and maximize overall throughput without sacrificing latency and reliability. To achieve this, it is extremely important that the integrated systems cooperate asynchronously. That is, distant systems should never block on each other waiting for a response.

// Simplistic use of new Web services technology results in fragile integration just as with older integration technologies"



This principle is important because when you integrate databases, Web servers, and enterprise application servers, some systems are always much faster or much slower to respond than others. There are latency differences because some systems are transactional and others aren't. Other latency differences arise when some systems work in a batch and others don't, or when some systems need to interact with people and others don't.

With different systems operating on different time scales, the challenge is to be able to manage conversations between systems without allowing any one system to block another. If a client blocks waiting for a response, the latency and reliability of the client will be tied to the latency and reliability of the server: if the server is slow, the client will be slow; and if the server is down, the client will be down. In other words, blocking clients by forcing them to wait for servers makes your whole system as slow and as unreliable as your weakest link. Blocking also wastes valuable threads, connections, and other cached resources by locking them up while idle: blocking in the face of latency makes it impossible to scale up overall throughput.

There are a few basic techniques that can be used to manage conversations and avoid blocking:

- **Queuing:** Instead of processing a message right away, queue it to process later.
- **Polling:** The client sends correlated messages to check status later.
- **Callbacks:** The server sends correlated messages back to the client to report status.

Since queues can continue to operate even when senders or listeners slow down or stop, queuing is useful for smoothing out spikes in load, and for improving the overall reliability of a system that may have unreliable components. JWS provides several queuing mechanisms, including buffering for HTTP messages and binding Web services directly to a JMS queue.

JWS Web services can be bound directly to JMS message queues for use in transporting either SOAP messages or unenveloped XML messages. That is,

JWS can treat a J2EE Java Message Service queue as an additional protocol for Web services transport – and one with all the desirable properties of a reliable message queue. This capability enables JWS developers to easily leverage the J2EE mechanism for asynchronous messaging.

Servers that avoid blocking the client need a way to get information to the client later. This is done either by sending callbacks to the original client, or by letting the client poll for information. In general, callbacks have more desirable properties than polling, since a client doesn't waste time looking for a result that isn't ready yet. However, callbacks require that clients act as servers, and if this isn't possible you must use polling.

#### *Example of Asynchrony: Queuing and Callbacks in JWS*

Let's take a look at an example that includes an asynchronous callback that gets some credit information for a specific purchase.

Here's what we want to do when a credit check arrives:

1. Quickly queue the message and release the client.
2. Without blocking the client, determine whether the credit charge is valid.
3. Send a callback indicating whether the charge is valid or not.

This implementation shouldn't tie up clients even if there is a heavy load and the server needs a lot of time to validate messages.

This is a simple two-message conversation with one inbound call and one callback. The conversation could easily be extended to include subsequent messages to be exchanged over a period of time, such as "reserve charge," "commit charge," and "expire reservation."

Listing 3 has the JWS code example to implement the asynchronous conversation. We've already seen how to define incoming operations, but how are callbacks defined? The JWS file defines an inner interface called Callback whose methods are the callback methods. In the main JWS class, a member

SAVE 30% off the annual newsstand rate

## JAVA DEVELOPER'S JOURNAL

Offer subject to change without notice

ANNUAL NEWSSTAND RATE	
<del>\$71.88</del>	
YOU PAY	
<b>\$49.99</b>	
YOU SAVE	
<b>30%</b>	Off the Newsstand Rate

### DON'T MISS AN ISSUE!

Receive 12 issues of **Java Developer's Journal** for only \$49.99! That's a savings of \$21.89 off the cover price. Sign up online at [www.sys-con.com](http://www.sys-con.com) or call 1 800 513-7111 and subscribe today!

### In April **JDJ**:

#### *Pervasive Computing: The Next Generation of Consumer Applications*

Now that broadband connections are becoming commonplace and wireless technology is becoming a reality, developers should begin positioning their applications to leverage the new capabilities.

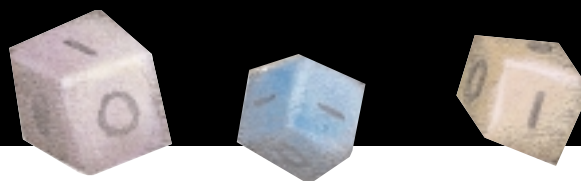
#### *J2EE Application, Module and Component Packaging*

The different types of J2EE modules and how they fit into the J2EE architecture.

#### *JDiff – What Really Changed?*

JDiff is an open source Java tool, based on Javadoc and developed by the author, that produces HTML documentation describing the precise API changes between two versions of a product.





variable called `callback`, which is an instance of the inner `Callback` interface, is also declared.

The following definition declares the callback operation for the service (the code below is valid but simplified; if you look at the full listing, you'll see that it adds some additional annotations for extra functionality):

```
interface Callback {
    /** @jws:conversation
    phase=finish */
    void sendResult(boolean
allowed);
}
```

In order to actually send callbacks, we first define the following field in the main JWS class:

```
Callback callback;
```

We then call code such as the following in one of the Web service methods:

```
callback.sendResult(true);
```

The above call sends a callback message back to the client associated with the proper conversation ID.

Notice that the callback object is automatically hooked up to the correct client on your behalf: one of the features of JWS is to automatically manage callback plumbing. Before any of your code runs, a callback object supporting your `Callback` interface is created, initialized, and hooked up so that calling it will send messages back to the proper client instance. What would ordinarily be a lot of manual setup is all done automatically.

In the example, we've queued incoming messages using JMS as a transport for SOAP messages. This code has a couple relevant annotations above the declaration of the `processCharge` method:

```
* @jws:conversation phase=start
* @jws:protocol jms-soap=true
http-soap=false
```

The `@jws:conversation phase=start` annotation indicates that sending the message starts a conversation that consists of a sequence of messages sent over time. When you send a message to a method marked as a start message, the client and server establish a new conversation ID that it used to correlate all the messages in the sequence.

The `@jws:protocol jms-soap=true` annotation binds the Web service to a JMS queue. In addition to being able to provide Web services on HTTP, JWS includes the capability to provide Web services on a JMS queue. (It can also be configured to handle raw XML messages that aren't in SOAP envelopes.) When a client message is sent to the method that is listening on JMS, it doesn't wait for the server to do any work. The message will be processed later using a message-driven bean that consumes messages at a rate determined by how quickly the server can do the credit-validation work.

In this example, we've seen how JWS supports JMS as a transport for binding Web services messages. JWS also provides support for a JMS control that enables a JWS file to communicate with existing messaging systems (e.g., an integration broker) using either the point-to-point or publish/subscribe paradigms. The JMS control is just one of a collection of controls that can be used to integrate a JWS service with other enterprise resources.

## Rapid Integration

One of the goals of JWS is to make it easy to build Web services that connect to other resources, such as databases, other Web services, EJBs, or enterprise application logic. To facilitate this, JWS supports a clever mechanism called JWS Controls.

JWS Controls provide a standard programming framework that allows developers to communicate with various heterogeneous resources. They eliminate the need to learn a set of sophisticated APIs to access and integrate with a range of different types of enterprise applications and services. Each JWS Control has default behavior associated with it, and using the JWS annotation syntax you can easily customize control properties and override default behavior. Since JWS Controls hide the complex class hierarchies and APIs normally used to access enterprise resources, they allow any developer who understands basic programming to fully leverage these resources.

The example in Listing 3 uses a database control to implement its logic. The control field declaration in the example looks like this:

```
/**
 * @jws:control
 */
CustomerCreditDatabase creditDB;
```

In a JWS file, any field that is annotated as a `@jws:control` is automatically managed. So the details of how the `creditDB` control object is created, hooked up, and shut down are left to the system. For example, the control automatically manages the database connection and transaction context for you. You don't have to worry about preparing the statement, closing the database connection, dealing with result sets, or other details of JDBC; the database control and the JWS container will make sure the database dance is done correctly. As a JWS programmer, all you need to do is call the control within your Web services methods, as follows

```
double currentLimit;
currentLimit =
creditDB.getLimit(customerNo);
```

Use of other controls is just as simple: a JWS programmer can access most enterprise resources using simple, procedural Java.

Much of the magic of controls is in how they are customized. Developers and vendors can define their own controls. For example, the BEA WebLogic Workshop JWS control package includes the following base controls:

// Blocking clients by forcing them to wait for servers makes your whole system as slow and unreliable as your weakest link"

- **ServiceControl:** A highly customizable Web service proxy
- **DatabaseControl:** Provides JDBC functionality
- **EJBControl:** Provides access to session and entity beans
- **TimerControl:** Can wake services at specified intervals
- **JMSControl:** For listening, sending, and pub/sub on JMS queues and topics
- **ApplicationViewControl:** Provides access to the J2EE connector architecture

Customized controls, such as Customer CreditDatabase, that derive from base controls are defined using a .ctrl file that defines the desired control interface in Java along with some Javadoc annotations. The annotations make customization easy, and they allow the details of the implementation of the customized control to be left up to the base control.

Listing 4 (Listings 4-6 can be found on the Web site at [www.sys-con.com/weblogic/sourcec.cfm](http://www.sys-con.com/weblogic/sourcec.cfm)) shows the Customer Credit Database control, which is based on the Database Control. Since the Database Control has been customized using annotations such as @jws:sql, all the ordinary JDBC plumbing code for communicating with a database is handled automatically. While the customized control is concise and easy to maintain, its automatic implementation is also meticulously correct.

Another interesting example, shown in Listing 5, is a ChargeCheckjControl that can be used to access the service from Listing 3 remotely. It is based on the Service Control. The JWS Web Service Control is a highly customizable Web service proxy. The customization allows you to explicitly manage aspects of the proxy, such as the XML messages, shapes and buffering.

The JWS Control model is designed to be extensible in the future, so new types of controls can be added over time. As you can see, controls can simplify integration problems and reduce the cost of development and maintenance of a Web service.

## Putting it Together

Listing 6 shows one final example: here, we've modified OrderingSystem.jws to operate asynchronously using a conversational sequence of messages. We've also chained the use of the asynchronous Web service ChargeCheck.jws through the ChargeCheck Control object called chargeCheck.

In this final example you can see how the JWS callback model works between two Web services (one Web service generating the callback and the other receiving the callback): conversational callbacks from chargeCheck are automatically routed to a callback handler method in OrderingSystem.jws based on a simple naming convention. Again, the JWS container is responsible for doing all the callback routing and plumbing for you; you just supply a method that follows the proper naming convention. The callback handler to listen for sendResult events from chargeCheck is written as follows:

```
void
chargeCheck_sendResult(boolean
ok)
{
    if (ok)
        callback.sendAck();
    else
        callback.sendError("Not enough
credit.");
}
```

When the callback handler gets a message indicating whether or not the credit has been approved, the code sends a callback to the original client. The technique of using asynchronous callbacks allows long chains of computations to be put together without blocking the original client.

## Summary

JWS makes it easy to solve knotty integration problems using a simple service-oriented architecture. In a couple hundred lines of code, we have defined two Web services and two custom JWS Controls. The Web services exchange SOAP messages over HTTP as well as JMS, and they expose WSDL contracts. They man-

SAVE 30% off the annual newsstand rate

**wireless**  
BUSINESS & TECHNOLOGY

Offer subject to change without notice

ANNUAL NEWSSTAND RATE	
<del>\$71.88</del>	
YOU PAY	
<b>\$49.99</b>	
YOU SAVE	
<b>30%</b>	Off the Newsstand Rate

## DON'T MISS AN ISSUE!

Receive 12 issues of **Wireless Business & Technology** for only \$49.99! That's a savings of 30% off the cover price. Sign up online at [www.sys-con.com](http://www.sys-con.com) or call 1 800 513-7111 and subscribe today!

## In April **WBT**:

### More Than an Overnight Success

Federal Express developed the first wireless package-tracking system in the industry over 20 years ago.

### It's a Small World

And getting smaller at Disney World, where wireless LANs are employed throughout the giant park to aid employees and enhance visitors' experiences.

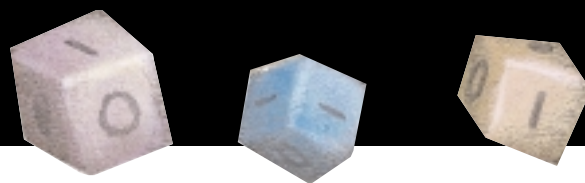
### Will SPRINT'S Strategy Succeed?

The number 4 wireless carrier seems to be rolling the dice with heavy bets on Java.

### Wireless Java is Coming

With the arrival of wireless Java, users are finally able to add exciting new applications to their devices.





age asynchrony with sequences of conversational messages, they interact with a database, they use queues for handling load, and they use custom XML serialization to improve robustness of the wire contract. And the code is simple.

- By annotating ordinary Java code with special JWS Javadoc comments, we can use advanced Web services and J2EE technology without using complicated APIs.

- JWS supports automatically generated XML message shapes, but it also facilitates loose coupling by allowing the XML messages to be specified explicitly.
- JWS facilitates asynchronous integration by managing conversational sequences of messages and callbacks, and by providing message buffering.
- The JWS Control model allows us to easily connect to external resources such as

databases, enterprise applications, EJBs, and Web services.

Since JWS is ordinary Java code running in a J2EE container, you always have access to the full power of standard J2EE functionality whenever it is needed. But JWS doesn't require use of complicated XML, J2EE, or Web services APIs. The power and robustness of JWS is fully accessible through Java code that is simple, short, and easy to write and maintain. ©

## Listing 1: An example purchase order

```
(SOAP envelope not shown)

<purchaseOrder>
  <customerInformation>
    <customerName>Acme Manufacturing
      Inc</customerName>

  <customerNumber>332</customerNumber>
  </customerInformation>
  <basket>
    <lineitem>
      <description>High Temperature
        Hex Bolt #66</description>
      <price>0.47</price>

    <catalogNumber>17466</catalogNumber>
    <quantity>220</quantity>
    </lineitem>
    <lineitem>
      <description>High Temperature
        Hex Nut #66</description>
      <price>0.47</price>

    <catalogNumber>18266</catalogNumber>
    <quantity>200</quantity>
    </lineitem>
  </basket>
</purchaseOrder>
```

## Listing 2: OrderingSystem.jws Web service

```
class OrderingSystem
{
  /**
   * @jws:operation
   * @jws:parameter-xml xml-map::
   * <purchaseOrder>
   *   <customerInformation>
   *
   * <customerNumber>{customerNo}</customer
   *   Number>
   *   </customerInformation>
   *   <basket>
```

```
*   <lineitem
xm:multiple="item in items">
  *
  <catalogNumber>{item.catNumber}</
  catalogNumber>
  *
  <quantity>{item.quantity}</quantity>
  *   </lineitem>
  *   </basket>
  * </purchaseOrder>
  * ::
  */
  public void submitOrder(int
    customerNo, LineItems[] items)
  {
    // logic omitted for now
  }

  static public class LineItems
  {
    public int catNumber;
    public int quantity;
  }
}
```

## Listing 3: ChargeCheck.jws Web service

```
class ChargeCheck
{
  Callback callback;

  /**
   * @jws:control
   */
  CustomerCreditDatabase creditDB;

  /**
   * @jws:operation
   * @jws:conversation phase=start
   * @jws:protocol jms-soap=true
   * @jws:parameter-xml xml-map::
   *   <request-charge>
   *
   * <customer>{customerNo}</customer>
   *
```

```
<amount>{amount}</amount>
  *   </request-charge>
  * ::
  */
  public void processCharge
    (int customerNo, double amount)
  {
    double currentLimit;
    currentLimit = creditDB.
      getLimit(customerNo);
    if (amount < currentLimit)
    {
      creditDB.setLimit(customerNo,
        currentLimit -
          amount);
      callback.sendResult(true);
    }
    else
    {
      callback.sendResult(false);
    }
  }

  interface Callback
  {
    /**
     * @jws:conversation
     *   phase=finish
     * @jws:protocol
     *   jms-soap=true
     * @jws:parameter-xml
     *   xml-map::
     *     <charge
     *   allow="{allowed}" />
     * ::
     */
     void sendResult
       (boolean allowed);
  }
}
```

Download the code at  
[sys-con.com/webservices](http://sys-con.com/webservices)



# Dynamic Buyer

[www.ibm.com/smallbusiness/dynamicbuyer](http://www.ibm.com/smallbusiness/dynamicbuyer)

# Business Process Management of Web Services



## AUTHOR BIO:

Christen Lee is a product marketing manager at Sun Microsystems. She is responsible for marketing iPlanet Integration Server and iPlanet Message Queue, as well as tracking emerging integration technologies, with a special focus on business process management.

christen.lee@sun.com

**C**reating a Web service is not a technology story that's limited to applications talking XML to one another and using registries. Rather, the real endgame is enabling companies to deliver useful services to key constituencies – employees, customers, and partners. To cite Metcalf's Law, a single Web service interfacing entity on the network is of little value. However, as Web services grow in number, they grow in value exponentially because they present the potential for new combinations.

Integration plays a pivotal role in the delivery of Web services. Ranging from encapsulation of data objects as Web services to the transformation of Web services from one system to another, integration solutions can add tremendous technical capabilities to an entire Web services infrastructure. In fact, a key integration concept, business process management (BPM), represents one of the most critical elements of composing Web services into complete applications. This article highlights how organizations can elegantly achieve this goal with BPM tools.

BPM concepts, including modularity, routing, and automation, are all critical elements of Web services. After all, Web services are service objects that can be inserted into larger solutions in a modular

fashion; interactions with Web services depend on messages being routed to and from them; Web services have description files which allow for automatic discovery. BPM tools should be used to incorporate various Web services to produce composite applications that actually serve a business function. In other words, the



value of establishing a Web services-ready infrastructure will be realized by the coordination of multiple Web services – in the intelligent BPM-driven integration of the separate components.

BPM tools have to extend beyond the traditional programmer environment to include line-of-business managers and analysts who are often responsible for delivering these services. With traditional IDEs, the developer works with segments of code such as functions or libraries, but BPM tools operate at a higher level of abstraction than code. With BPM tools that coordinate Web services, the user works with entire applications or services. BPM tools allow for the orchestration business activity.

## Evolution of BPM

Unfortunately, the numerous terms used in the BPM space have given rise to con-

fusion. The legacy of "workflow," "process automation," and even "BPR" (business process reengineering) all contribute to the confusion over what's meant by BPM. The development of Web services has given rise to even more applications of BPM concepts, and led to greater misunderstanding. The following stages outline the evolution of BPM from pre-Internet days to its current applicability with Web services.

### Stage 1: Workflow before the Internet

Workflow applications before the takeoff of the Internet were employee productivity tools delivered via desktop applications, often integrated with document routing and document version control. Automation at this level bypassed interoffice mail and manual re-entry of data.

### Stage 2: EAI and Process Automation

When IT departments and software engineers divorced the workflow components of these applications from the document management portion and then applied it to systems management, the enterprise application integration software advanced to a new level of sophistication and agility. The concept of capturing business rules to route work from one employee to another was transferred to machines in order to manage the communication between two applications, with goals of even higher throughput rates and greater data integrity.

### Stage 3: Application Servers and Component Coordination

With the rise of Internet browsers and the Java programming language, employee productivity tools were no longer limited to desktop applications, and applications created by integrating disparate legacy systems can be given a Web front. Java components could be used to present Web forms to employees or to invoke existing applications. Process automation then took on the

role of component coordination. "Work" flowed from one JSP, servlet, or EJB to another. Web interaction through JSPs allowed for manual inputs into the process; automated steps are captured by servlets or EJBs. BPM tools in this group combine human workflow and machine process automation and managed the entire process.

#### Stage 4: Web Services and Business Process Management

Even with more mature tools, integrating applications within one company is no small task. With Web services, BPM tools need to apply concepts such as code reuse and graphic modeling to multiparty application development. While in its early inception stage, Web services can and will be used in intraenterprise applications as an integration standard; the greater value proposition of Web services lies within a business-to-business context. In fact, BPM applied to Web services can be thought of as a standard means to B2B integration or even B2BPM. Following are some of the ways that BPM tools can enable Web services:

- **B2B Integration and B2BPM:** For starters, a successful Web services infrastructure will often include elements such as a business-to-business friendly messaging system and a lingua franca for the messages to be passed. The new Web services-related standards such as ebXML and SOAP address these needs. The ebXML specification includes an entire section on Message Handling Service (MHS) that describes the transport, routing, and packaging (TRP) of messages to be passed between partners. The SOAP specification describes how SOAP-based XML messages can serve as the common message format. Messaging protocols and formats are the areas in which the standards are first maturing. Currently, software vendors are most active in this area. Building on top of the messaging system, BPM tools can be used to establish exchange patterns, conduct the flow of intercompany communications, and automate processes at a business transaction level.
- **Service Description and Collaboration Modeling:** In addition to taking part in the orchestration of Web services at runtime, BPM tools can have a role in creating the Web service descriptions needed for registries and repositories. Web service descriptions will not only contain basic business function identifiers and contract definitions, but by adhering to specifications such as BPSS (Business Process Specification Schema), these descriptions can represent inter-enterprise processes

within the XML-formatted description file. Such descriptions hold process-related information such as what to do with a virtual purchase order once it has been received, what and when information needs to be sent back to the business partner who posted the Web service, and in what order each step should be carried out. BPM tools offer a way to model this business level collaboration through a graphic user interface and then generate the necessary description file.

- **Service Registry and Dynamic Discovery:** Furthermore, BPM tools can be extended to dynamic discovery of services as well. As Web services adoption grows, more and more public services become available, and registries are well populated, process tools will adopt features that enable dynamic discovery of Web services. This is the final frontier of Web

- Knowing which registries to research and in which order
- Keeping track of the time restraints on the discovery process
- Initiating a negotiation when description of a service that fits the requirements is found
- Negotiating the terms on which the communication between the two entities will be conducted

Carefully crafted processes will capture the business logic needed to conduct such discovery and negotiation processes without manual intervention, thus instilling a level of artificial intelligence into Web service-incorporating applications. Extending the analogy of the process manager as the central nervous system, advanced BPM tools in the Web services world have the potential to turn once-conscious activities into autonomic functions.

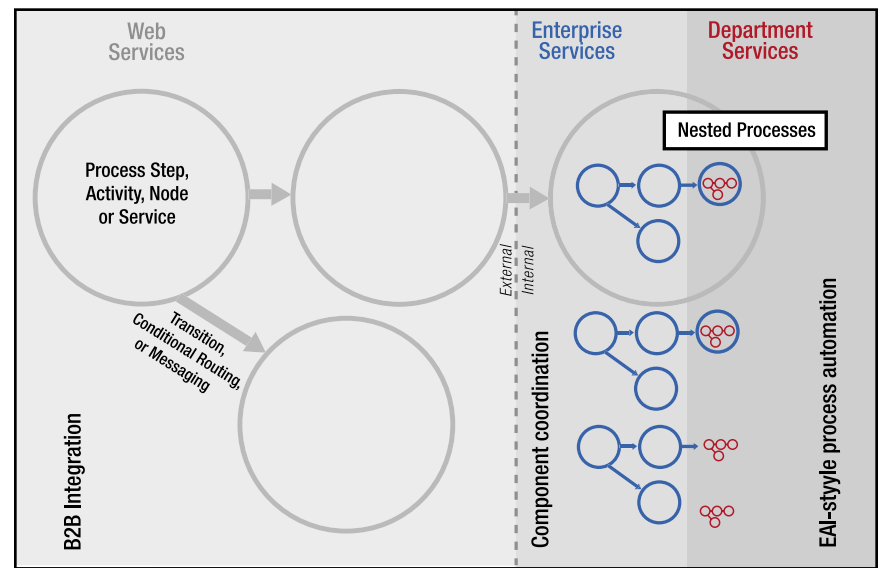


FIGURE 1 | Nested processes depicting the varying scale of process integration projects

services: the ability to go to an outside source for a service before knowing the source. In order for dynamic discovery to become a reality, the following tasks would need to be automated:

- Detecting when the process needs to go to an external source; i.e., in an expense reporting application, an up-to-date currency converter is required

- **Service Orchestration: Monitoring and Enforcement:** The potential exists for BPM tools to expand their relevance over the entire life cycle of a Web service. Once discovered and engaged, how will Web services be monitored? How will the rules of engagement be enforced? Such process management challenges are nothing new to BPM tools of the past.

greater potential for  
return on investment



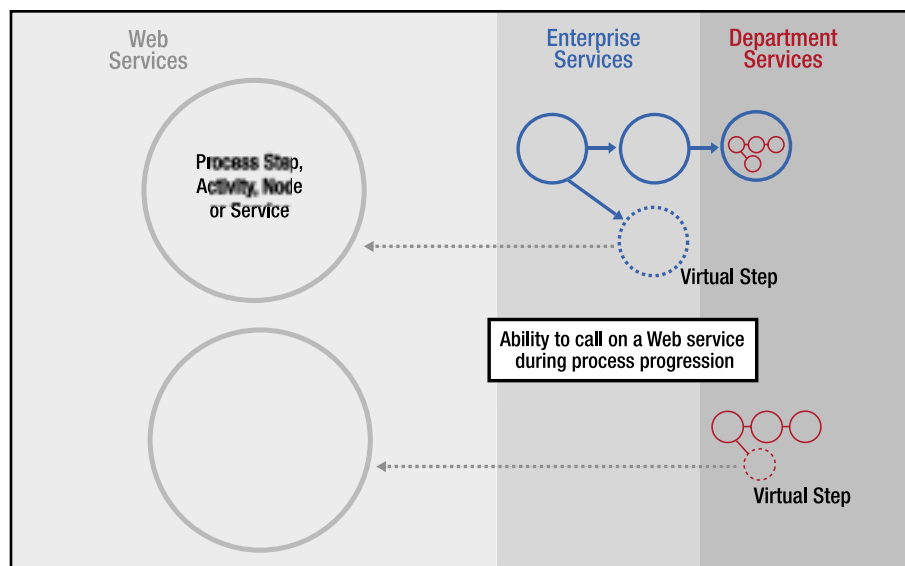


FIGURE 2 | Existing processes with the incorporation of a Web service as one of the steps



Traditionally, they've included administrative consoles with various views of the activity and interactivity they create. These monitors are no less important with the inclusion of Web services. In essence, the process monitors themselves will be monitored to ensure that the terms established during the negotiation phase are being adhered to. For example, a Web service procures certain widgets for a company from another for a certain price. The price may have been promotional and has now expired. Once an internal procurement application tries to call on this Web service again, the request is rejected; it's in violation of the contract. BPM tools with enforcement

capabilities could detect this and stop any further requests for this service or even initiate a search for a replacement service.

As outlined above, the utility and ubiquity of BPM concepts are clear. Different analyst groups may propose slightly varying taxonomies, and just about every software data-sheet contains similar terminology. In order to discern the differences and find the right tool for the job you must understand scale. Different BPM tools may have been optimized for process integration at different levels of granularity.

Take, for example, a very large, long-established company with IT shops in various departments. During the early '90s they streamlined their business by taking out manual steps where they could and integrating large packaged applications and mainframes with some of the custom work they had done. Let's say that the department in this case was the Shipping department and the streamlined process involved integrating the various package-tracking systems. It's very possible that an EAI-style BPM tool was used here.

In the late '90s, the same company changed their focus from direct mail to online catalogs. In order to advance from simple online brochures to an e-commerce-enabled Web site, the company

needed to further integrate its processes; the online catalog had to be integrated with the billing department and the inventory system as well as the shipping process mentioned above. Keeping with the object-oriented programming model, each newly developed part was developed as a Java component. A single servlet running on a J2EE application server was written to trigger the shipping process. The component-coordinating BPM tool treated this application as one node in the larger order fulfillment process.

Today, the company has the option of starting a wholesale business, selling directly to a reseller partner. From the point of view of the partner, the entire order fulfillment process could be encapsulated as a Web service called "pro-cure widget (see Figure 1)." The partner may have solicited this service from several different vendors in hopes of a finding the lowest price. The process involved in this bidding will also benefit from BPM tools to give it shape.

The example above describes how a process may be nested in another process, which is in turn nested in another. The final process happens to be one that spans company boundaries and does so in a very specific Web service-compliant way.

A complete Web services-enabled BPM tool can present an internal application as a Web service and manage all the processes involved in creating, describing, and registering Web services. Furthermore, it should be able to find, negotiate, elicit, and monitor Web services. This is a rather tall order. Expect the first entrants into the market to resemble traditional BPM tools with an added feature: the ability to call on Web services into its process when necessary (see Figure 2).

## Conclusion

Just as Web services are a natural progression of networked computing, message-oriented-middleware, and object-oriented programming, BPM tools continue to evolve, making the gradual transition to a Web-services style of B2B integration. This change will track the development and adoption of Web-services-related standards. Initially, the upfront investments required in establishing a homogenous way for businesses to communicate electronically will seem great, but the potential for return on that investment is much greater. BPM tools have the potential to bury much of the complexity, offer an easier user interface for soliciting Web services, and provide greater impetus for adopting Web services in the enterprise architecture. ©

**BPM tools continue to evolve, making the gradual transition to a Web-services style of B2B integration.**



# **Simplex Knowledge Company**

**[www.skcn.com](http://www.skcn.com)**

# Asynchronous Web Services

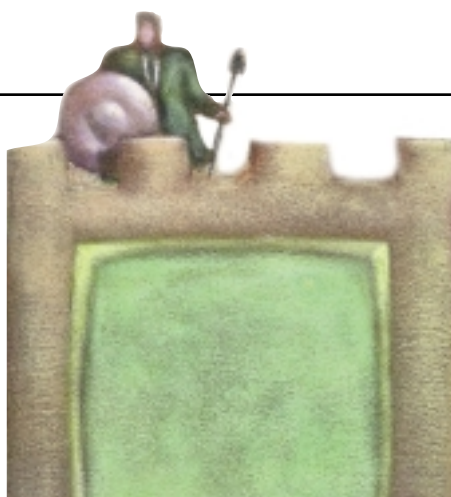
A trend that makes even the timid happy

In a recent "Strategic Planning" research note, Gartner issued a prediction that "by 2004, more than 25 percent of all standard Web services traffic will be asynchronous...." and "by 2006, more than 40 percent of the standard Web services traffic will be asynchronous."

One of the cornerstones of Web services interoperability is the SOAP (Simple Object Access Protocol). SOAP began simply as a way of performing a synchronous RPC (Remote Procedure Call) across the Internet over an HTTP connection.

However, the current "owners" of SOAP – the XML Protocol WG at the W3C – have been trying to break free of that limiting portrayal. The SOAP spec itself says:

- "SOAP messages are fundamentally one-way transmissions from a sender to a receiver, but as illustrated above, SOAP messages are often combined to implement patterns such as request/response.
- SOAP implementations can be optimized to exploit the unique characteristics of particular network systems. For example,



the HTTP binding... provides for SOAP response messages to be delivered as HTTP responses, using the same connection as the inbound request."

It sounds to me like verbiage that's going out of its way to say that RPC over HTTP is purely coincidental. SOAP is designed to be layered on top of, or "bound" to, any protocol. So far the only protocol binding defined in the specification is HTTP.

In the current SOAP 1.2 working draft, SOAP for RPC has been moved into the "Part 2, Adjuncts" portion of the spec, the place where optional features are kept. Another visible indication of the desire for SOAP to become asynchronous is evident in the XML Protocol Abstract Model, which clearly talks about asynchronous interactions across multiple intermediaries.

WSDL (Web Services Description Language) describes whether an invocation style is "document" or "rpc". It defines four operation modes—one-way and request/response modes are used to describe an inbound asynchronous receive by a service or an inbound/outbound

synchronous RPC. Notification and solicit/response are used to describe the reciprocal of those operations, when the operation is initiated by the server to the client. All operation modes are equal citizens in the world of Web services interactions.

Synchronous RPC provides the luxury of immediately knowing whether an operation was successful. However, performing a synchronous operation across multiple processes is an all-or-nothing proposition. If one of the processes isn't available, the application initiating the request must somehow make note of the failure, try again later, or take some other more drastic course of action – like inform a human that they are simply out of luck.

In contrast, asynchronous messaging allows each communication operation between two processes to be a self-contained, stand-alone unit of work. Each intermediary in a multi-process communication can have its own distinct conversation with its sender and its sendee. The process initiating the original request need only be concerned with initiating the "request", knowing that it will eventually receive a "response" asynchronously. Asynchronous processing also allows critical failures to be centrally reported and dispatched via an administrative alert system.

## Asynchronous Reliability

Asynchronous processing also brings with it the requirement of ensuring reliability. In Web services, there are multiple ways of achieving that.

One approach to achieving asynchronous reliability between Web services is to provide a reliable protocol at the SOAP level. In this approach, message acknowledgements and delivery receipts are encoded in predefined SOAP envelope header constructs. An example of this is specified in the ebXML Message Service. However, be careful about jumping on that bandwagon just yet. Microsoft, one of the major players leading the charge in defining Web services, isn't participating in ebXML. This same problem is likely to be addressed eventually by SOAP itself.



### Author Bio

David Chappell is vice president and chief technology evangelist for Sonic Software. He is responsible for articulating Sonic's vision for secure, reliable Web services infrastructure. Dave is co-author of *The Java Message Service*, *Professional ebXML Foundations*, and *Java Web Services*.

CHAPPELL@SONICSOFTWARE.COM

## // All operation modes are equal citizens in the world of Web services interactions"

Another alternative is to extend reliability to the HTTP layer. HTTPR has been proposed by IBM but to date has not been submitted to any standards bodies.

The Java Message Service (JMS) provides message queuing and guaranteed delivery semantics, which ensure that "unavailable" applications will get their data queued and delivered at a later time. It accomplishes this via a rigid set of rules governing message persistence and message acknowledgements. JMS has been used for several years to transport XML data between applications asynchronously and reliably.

In the end, the clear solution is a Web services *deployment* that's based largely on JMS, yet allows the ability to "extend and blend" SOAP-over-HTTP with SOAP-over-JMS.

Gartner further claims that more than 70% of application integration deployments today are based on asynchronous processing rather than synchronous request/reply. I'm a firm believer that the lion's share of Web services projects in the next few years will still involve some form of "application integration." The analysts might give this phenomenon a new name, but the problems are still best addressed by asynchronous processing. New services will be built that act as participants in a larger composite application. Many of these will be coarse-grained interfaces to legacy applications.

The Gartner predictions look promising – somewhere between 25% and 70% of Web services interactions will be asynchronous in the not-too-distant future. I actually believe the percentages will be higher. However, even if they're right on the money, I'll be extremely happy with that. ☺

# SUBSCRIBE AND SAVE

## XML JOURNAL

Offer subject to change without notice

ANNUAL NEWSSTAND RATE

~~\$83.88~~

YOU PAY

\$77.99

YOU SAVE

\$5.89 Off the Newsstand Rate

### DON'T MISS AN ISSUE!

Receive 12 issues of **XML-Journal** for only \$77.99! That's a savings of \$5.89 off the annual newsstand rate.

Sign up online at [www.sys-con.com](http://www.sys-con.com) or call 1 800 513-7111 and subscribe today!

### In April XML-J:

#### Deploying Web Services on WebSphere

A real-life example, using the tools and services provided

#### ADO.NET & ASP.NET...

...How to use them to build XML applications

#### .NET Web Services: The 'Three I' Monster

Web services...middleware for the masses...XML miracle tools...instant integration: just add SOAPXML – *Beyond*

#### Middle-Tier Data Management

The middle tiers in the XDBMS architecture can be confusing

#### A Pragmatic Convergence of ebXML & Web Services

Is this a second chance for the industry?

#### Transport

Persistence for Web services





written by Laury Verner

# Why Orchestrating Business

## Processes is Key to

## Web Services

**W**hatever happened to the e-commerce movement? Is it an idea whose time has come...and gone? Was the idea of Web-based business just a passing fad despite the effusive predictions of the industry pundits?

New technologies and standards are emerging that will create the foundation for a second – and much larger – e-commerce revolution. The essence of this work is a framework for developing and deploying applications across the Internet – Web Services.

The goal of Web services is to enable business solutions to be developed rapidly by assembling prebuilt components. These components can be geographically dispersed, built independently, and provided by unrelated companies. Most importantly, solutions based on Web services will offer the user a rich and seamless application experience, quite different from the primitive link chasing we're used to. The ultimate promise of this technology is to change the way companies collaborate and conduct business. How will this promise be realized? A key success factor lies in an unexpected direction – the orchestration of business processes.

Success depends on  
effective integration  
into business processes



**AUTHOR BIO:**  
Laury Verner, PhD, is the chief technology officer of ProActivity, Inc. ([www.proactivityinc.com](http://www.proactivityinc.com)) He is an expert on business processes analysis and design (BPA).



## A Simple Analogy

To clarify the technical concepts, we'll use an analogy. Let's imagine a symphony concert, with numerous performers playing their parts. The conductor directs the musicians according to the score, which was written by a composer. In our analogy the instruments of the orchestra will correspond to Web services. The goal is to produce music – real business value – from this assemblage of instruments.

## Components

Traditional Internet applications are monolithic. As such, they exhibit a high degree of redundancy and offer little interoperability. Introducing software components changes everything. Each component behaves like a mini-application with its own interface. Through its interface, a component offers services that can be invoked by other components (see Figure 1).

Web services build applications from heterogeneous components that communicate across the Internet. The components may be from different vendors and may be located in different cities or countries. These components are the instruments in the "Internet Symphony."

The beauty of this approach is that components can be reused across applications. For example, we might use a common login and security components to gain access to two different applications.

## What Are Web Services?

"Web services" are self-contained, modular components that can be described, published, located, and invoked over a network. The basic setup for Web services is a requester, a provider, and a broker. A requester finds service providers using a broker. Once the requester has found the desired provider, it binds to it and the provider carries out the requested service. Consider a mechanism for obtaining stock quotes over the Web.

The service requester sends a message to the service provider requesting a quote. The message contains the stock's ticker symbol. The service provider responds with the current stock quote (see Figure 2).

## Three Standards for Web Services

To ensure interoperability of Web services, certain basic rules and regulations are clearly needed. Three important standards that have been defined toward this end are SOAP, WSDL, and UDDI. These standards are driven by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force, as common mechanisms for building Web services.

### SOAP: Simple Object Access Protocol

SOAP defines the messages that components use to communicate with one another. SOAP, a lightweight protocol based on XML, specifies the rules for message exchange between

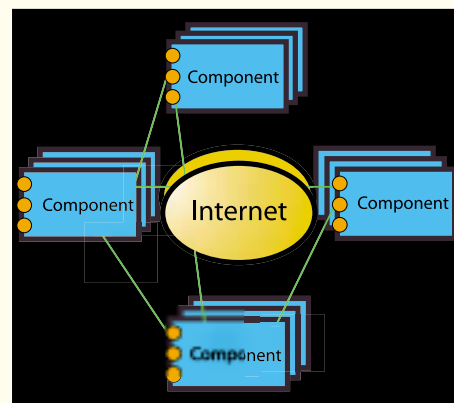


FIGURE 1 Building applications from components

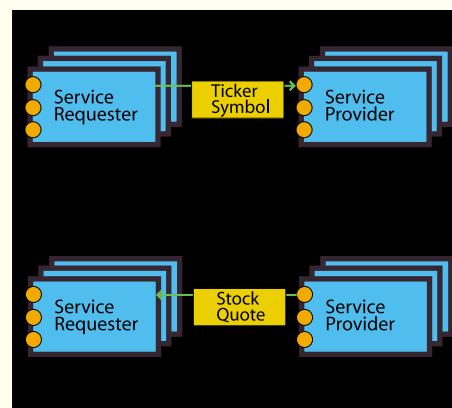


FIGURE 2 Web services

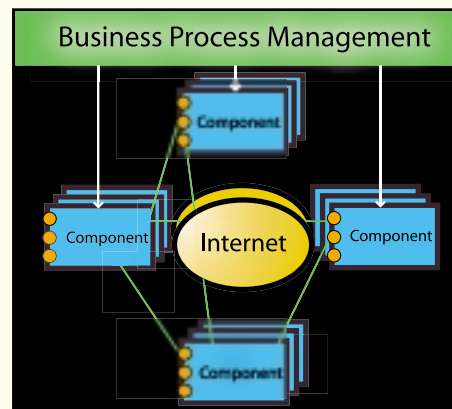


FIGURE 3 Business Process Management system

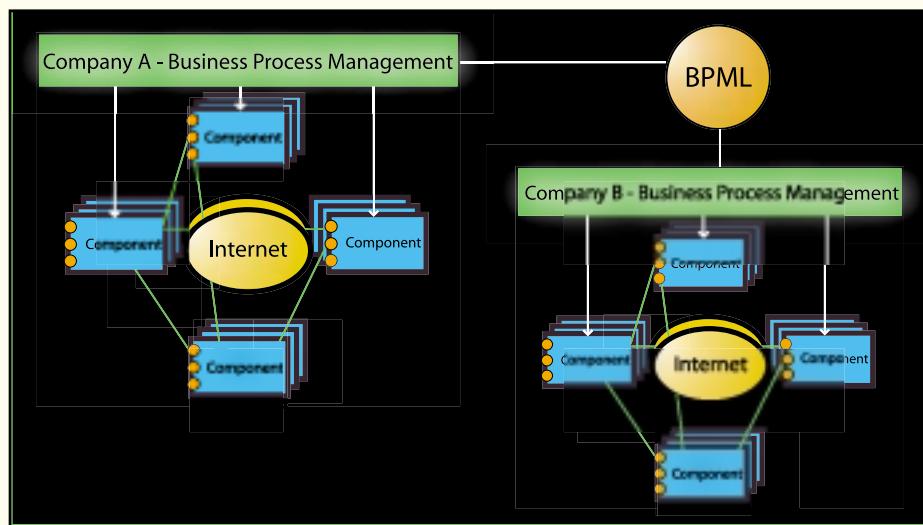


FIGURE 4 | Cross-enterprise processes using BPML

requester and provider components. Using SOAP, one component can send a message to another component to invoke a specific service. Components can communicate because they use a consistent approach to XML-based messaging.

#### WSDL: Web Services Description Language

WSDL is a standard language for describing the services offered by a service provider. WSDL considers a service to be composed of a collection of ports or "end-points." Each port has a network address and messages can be sent to it using a network protocol. These message formats are defined using SOAP.

Different instruments have different capabilities. You can pluck a violin and you can play a tuba with a mute. WSDL specifies these capabilities.

#### UDDI: Universal Description, Discovery, and Integration

Software components will be developed by many companies and will be accessible

across the Internet. The question then arises: How do I find the component I need? UDDI provides a global business registry that enables companies to discover one another, advertise their components, and explain how their components are designed to interact. UDDI can be thought of as a listing of musicians, describing the instrument each plays, and their contact information.

#### Directing the Orchestra

To make the Internet Symphony harmonious, we need a conductor. This role is played by the Business Process Management (BPM) system (see Figure 3). BPM directs the components, sequences component invocations, controls synchronization, manages transactions, and takes action when a component is busy or not working properly.

Many companies are developing or offering next-generation BPM products. Among these are BEA, Fuego, IBM, Intalio, Elagent, Lombardi, MetaServer, Nobilis, Q-Link, Savvion, and TIBCO.

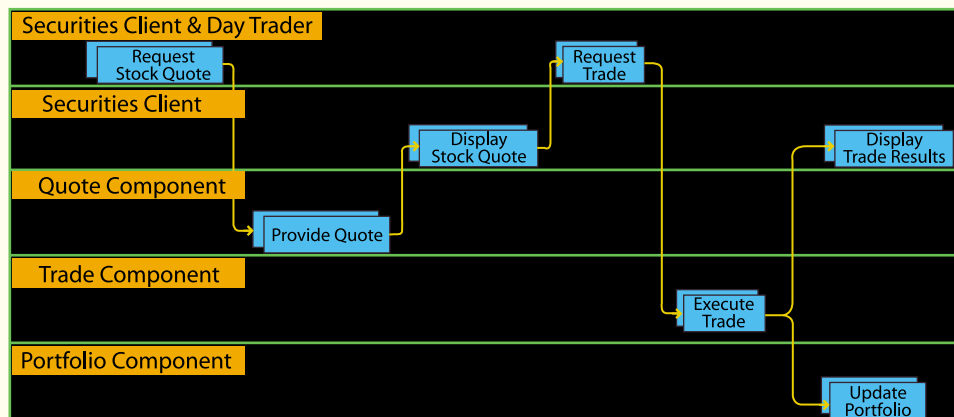


FIGURE 5 | Process map

The business process is the fundamental structure underlying BPM. It's a well-defined, repeatable sequence of services that can be performed by people, systems, components, or even mechanical devices. Key requirements for process execution include the ability to:

- **Handle exceptions and failures:** This is critical since the steps in a business transaction are not invariably successful. There must be an explicit means of unwinding transactions or providing compensating transactions.
- **Redeploy processes on the fly:** Since the environment is inherently unpredictable and requirements are constantly changing, processes must be dynamically changeable.



“Web services” are self-contained, modular components that can be described, published, located, and invoked over a network”

- **Dynamically reconfigure process participants:** In particular, the BPM system must provide a disciplined mechanism for re-assembling components.
- **Understand and preserve state:** This will allow users to query the status of an order or an insurance claim and to monitor and improve processes in flight.

Now, how can a company and its supplier communicate and exchange process information if both have BPM systems? The Business Process Modeling Language (BPML), sponsored by the Business Process Management Initiative, was developed for this purpose. It provides a set of concepts and rules for defining business processes and is designed to foster interoperability of systems that exchange

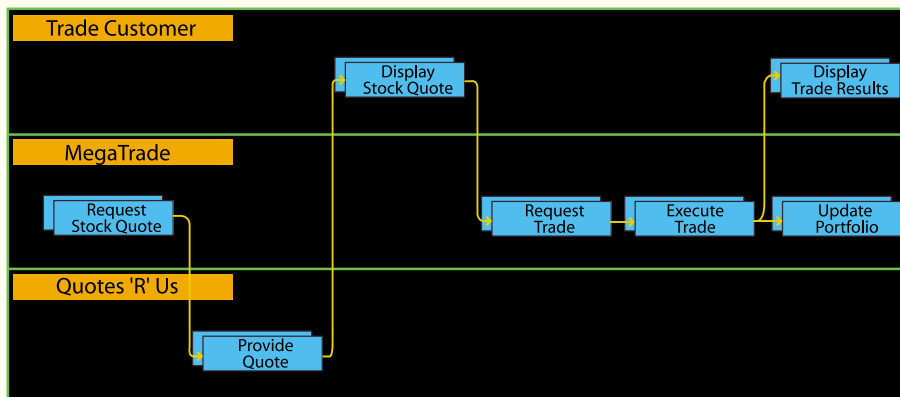


FIGURE 6 | Adjusted Process map

process information. Just as XML is language for modeling business data, BPML is a language for modeling business processes.

BPML views business process in terms of the flow of control, data, and events. Supporting these flows are business rules, security roles, and transaction management. All these elements are described by a XML schema. The idea is to enable trading partners to collaborate in their processes, even if they are run on different system infrastructures. BPML is designed to address all four requirements for process execution mentioned above (see Figure 4).

## The Score

The goal of the Web services movement is to enable the creation of new business value, within and across enterprises. To create value, business processes have to be incrementally improved, fundamentally redesigned, or freshly created. Each enterprise must begin by understanding their existing "as-is" business operations, analyzing them and diagnosing their strengths and weaknesses. Then they can redesign their processes and exploit the capabilities of Web services. The general term for this effort is "Business Process Analysis" (BPA).

In terms of our analogy, the business process is the score. It's the blueprint for the business, just as the musical score is the set of instructions for the conductor. The score specifies the notes each instrument should play, in what sequence these notes should be played, and which notes should sound at the same time. In many companies the instruments are playing – but without a conductor or a score.

At this point, it's natural to ask how the score is written. How do we effectively understand and redesign the business processes of the enterprise?

## Composing the Score

In the business world, business process design is the responsibility of business analysts, supported by IT professionals. They are the composers designing the future of the enterprise. Sophisticated tools are needed to help the business analysts to work quickly and successfully. Unfortunately, what most analysts use today are whiteboards, yellow stickies, and drawing tools, all of which are labor-intensive, prone to error, and hard to change.

There are now BPA tools on the market that are much more powerful and that

# WebServices

.NET J2EE XML JOURNAL

**The world's leading independent Web Services information resource**

**Get Up to Speed with the Fourth Wave in Software Development**

- Real-World Web Services: Is It Really XML's Killer App?
- Demystifying ebXML: A Plain-English Introduction
- Authentication, Authorization, and Auditing: Securing Web Services
- Wireless: Enable Your WAP Projects for Web Services
- The Web Services Marketplace: An Overview of Tools, Engines, and Servers
- Building Wireless Applications with Web Services
- How to Develop and Market Your Web Services
- Integrating XML in a Web Services Environment
- Real-World UDDI
- WSDL: Definitions and Network Endpoints
- Implementing SOAP-Compliant Apps
- Deploying EJBs in a Web Services Environment
- Swing-Compliant Web Services
- and much, much more!

**www.wsj2.com**

\*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Only \$69.99 for 1 year (12 issues)  
Newsstand price \$83.88 for 1 year

**Special Introductory Offer**  
**SAVE \$13.89\***



TABLE 1: Analytic report assembled by the BPA tool

Component	Activity
Login	Capture user ID and password
Security	Validate login and set access
Portfolio	Display user's portfolio Display recent transactions
Trade	Perform buy transaction Perform sell transaction
Quote	Provide stock quote
Analysis	Provide analyst report Display stock trend chart

enable detailed, dynamic analysis and design. Key requirements for next-generation BPA tools include the ability to:

- **Discover existing "as-is" processes:** Capture roles, activities, data flow, and IT infrastructure. Discovery should be focused, collaborative, and rapid.
- **Capture custom metadata:** User-defined goals, attributes, and metrics that are critical for the business.
- **Incorporate Web services** into the fabric of the design
- **Depict process information visually:** Present different views to different stakeholders.
- **Analyze process information:** Allows the user to design ad hoc reports that consolidate and summarize information across multiple processes.
- **Support iterative analysis and design:** Changes to process information should be reflected immediately and dynamically in all process views.

Since business processes are performed by components as well as systems and people, the BPA tool should deal with components as full-fledged participants that can perform and support process activities.

### Visualizing Processes: Viewing the Score

BPA allows the analyst to present processes in a visual format, with system components as process participants. Figure 5 is a process map organized into horizontal bands, called "swimlanes," each representing the activities performed by one of the Web components. In this process the participating components are Security, Quote, and Trade. The activities performed correspond to the services the components offer. Thus, the process map is a network of Web services.

The process map shown in Figure 5

presents one of many possible views. Other views highlight different aspects of the business process. For example, suppose that the buy/sell transaction involves a customer and two cooperating trading partners. The first partner is a Web-based stock trading service called MegaTrade. The second partner is Quotes'R'Us, a service that provides current stock quotes. You issue a stock ticker symbol to Quotes'R'Us and it returns the current price of that stock. Suppose we now need to change perspective to view the process through the lens of the two companies involved. Certain activities will be performed by the trading customer and others will be carried out by MegaTrade or Quotes'R'Us. We would therefore want to refactor the process map to look like the swim lane organization in Figure 6

With this structure, the business analyst sees exactly which functions each company performs. They see the touch-points and handoffs between the two companies. This is what is meant by "presenting different views different stakeholders."

Needless to say, this example is unrealistically simple. Actual business processes tend to be much more subtle and complex, often involving hundreds of activities, roles, components, routing logic, and business objects. When Web services are involved, it's necessary to be extremely rigorous and precise. The simple, high-level approach of the past will no longer suffice.

This complexity is raised to an even higher level when processes traverse multiple divisions and extend to customers, suppliers, and partners. The BPA tool must handle such complexity gracefully, providing effective mechanisms to filter out unneeded detail, and scaling to the extended enterprise.

The process map in the business world is analogous to the full orchestral score used by the conductor. The score is also organized into horizontal "swim lanes." One swim lane represents the violins, another the violas, and so forth. The conductor sees at a glance what each instrument is playing and understands how they combine to create the music.

### Process Analytics

The BPA tool should serve as a design laboratory, allowing business analysts to perform what-if experiments and generate various analytics. To achieve this goal, we must think of processes as *data* rather than pictures. All information about the activities, the participants, and so forth, must be stored as data in a database from which we can extract a wide variety of analytical information.

For example, we may want to under-

stand the activities performed by the system components. These components may provide services in multiple processes, but we need a consolidated view. The BPA tool will extract this information from the various processes, consolidate it, and publish it as a report (see Table 1).

The value of analytics is greatly enhanced when information about business objects, activities, resources, and custom metadata can be organized and summarized in any way the analyst desires, using a simple drag and drop user interface. Like the process maps, analytical reports should be dynamic, so that if the underlying process information changes, the reports will be instantly refreshed with the new information.



By integrating BPA with BPM these processes will be immediately executable, linking application components quickly and seamlessly"

### Integrating BPA and BPM

A composer can now buy software for creating a score. Not only that – there are products that will perform the music directly from the score, producing the actual tones of the musical instruments. In this way, the composer can understand in advance how the symphony will sound. (Note: One example is finale, by Coda Systems. In our analogy, the Internet Symphony provides audience feedback, in real time, directly to the composer to correct and improve the quality of the music. In business, unlike music, the score is iterative: it's always being adjusted and improved.



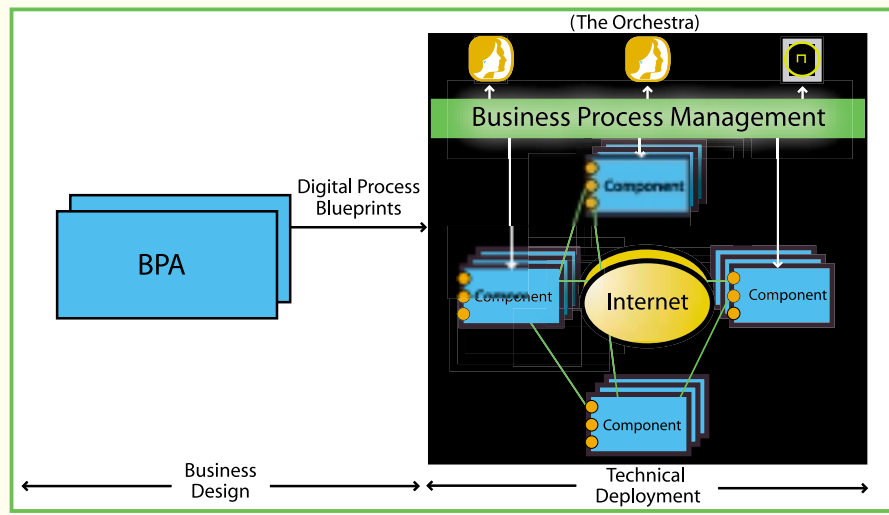


FIGURE 7 | Exporting digital process blueprints to the BPM environment

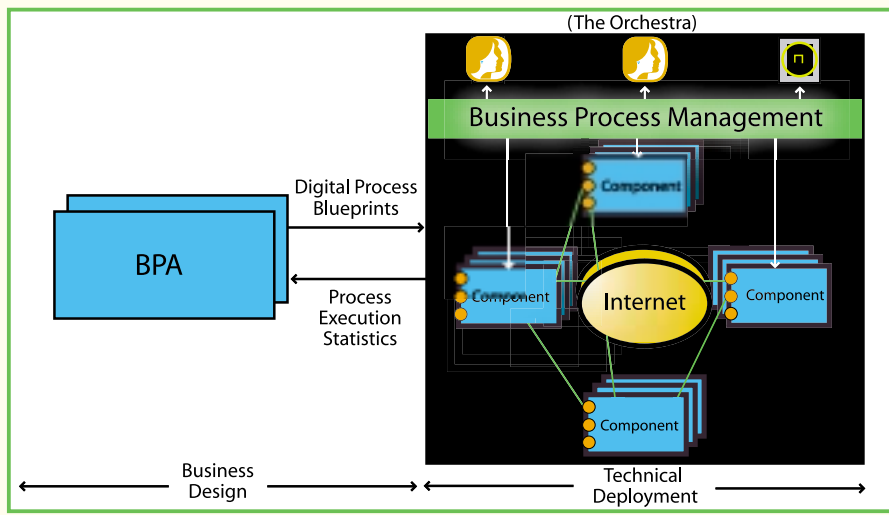


FIGURE 8 | Extracting information from the BPM tool

An analogous capability is needed in the business world. The BPA tool is used to capture the blueprint for the new business processes. The BPM tool must carry out the instructions of the blueprints. Clearly, what is needed is to export the process blueprints in digital form to the BPM environment (see Figure 7).

Each business process will be described as an XML document that will be exported into the BPM tool, accelerating the path from concept to operation. After the process is exported, there will be additional technical details that will be added in the BPM tool. But the business process architecture will have been defined in the BPA tool.

Over the long run, I envision the inverse procedure: extracting information from the BPM tool and integrating it into the BPA tool. This will enable business activity monitoring, to examine real-time process execution information or historical trends (see Figure 8).

With this feedback approach in place, companies will be able to compare the results of the process execution with the original

design. This will provide a means to correct the process design as well as to eliminate bottlenecks in the process execution. It will give managers the hard facts to determine if the business is achieving its operational goals.

Today, most companies have elaborate systems to support financial management, but there are no comparable systems to define, measure, analyze, improve, and control the operational side of the business. By closing the loop, companies will be able, for the first time, to optimize their business processes in near real-time. In terms of our analogy, the Internet Symphony is providing audience feedback, in real time, directly to the composer to correct and improve quality of the music. In business, unlike music, the score is iterative: it's always being adjusted and improved.

## Conclusion

Web services promise to transform application development and usher in a new era of e-commerce. My goal in this article has been to illuminate the major building blocks and

certain key success factors of this program. The analogy with musical performance, while imperfect, shows how these technologies fit together.

All technologies must be evaluated in terms of how they contribute to business goals. Web services, in particular, must be understood in this context. A key enabling concept for successful implementation of Web services is the business process. Without a rigorous, fine-grained approach to orchestrating business processes, Web services will remain an interesting topic for researchers but will have minimal practical impact on business. Orchestrating business processes requires attention to Business Process Analysis and Business Process Management. Processes designed using BPA are executed using BPM. Linking them will be an XML-based standard, such as BPML.

Next-generation approaches to BPA and BPM are emerging to meet these needs. BPA will give business analysts and IT professionals the laboratory they need to design and orchestrate business processes. By integrating BPA with BPM these processes will be immediately executable, linking application components quickly and seamlessly. This will allow enterprises and extended enterprises to realize the full promise of Web services.

## References

- Gisolfi, Dan. *Web services architect, Part 1: An introduction to dynamic e-business*. IBM developerWorks
- Gisolfi, Dan. *Web services architect, Part 2: Models for dynamic e-business*. IBM developerWorks
- Snell, James. *Web services insider, Part 1: Reflections on SOAP*. IBM developerWorks
- Microsoft. "Building User-Centric Experiences. An Introduction to .Net My Services." White Paper.
- *Simple Object Access Protocol (SOAP)*. W3C. ([www.w3.org](http://www.w3.org))
- *Web Services Description Language (WSDL)*. ([www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl))
- Shohoud, Yassir. *Introduction to WSDL*. ([www.devxpert.com](http://www.devxpert.com))
- *UDDI Executive White Paper*. ([www.uddi.org](http://www.uddi.org))
- *UDDI Technical White Paper*. ([www.uddi.org](http://www.uddi.org))
- *BPML* ([www.bpml.org](http://www.bpml.org))
- *BPML* ([www.bpml.org](http://www.bpml.org))
- *Finale* [www.codamusic.com](http://www.codamusic.com)
- Verner, Laury. "Developing and Implementing Enterprise Applications." White Paper, ProActivity. [www.proacti.vityinc.com](http://www.proacti.vityinc.com)
- GartnerGroup: Will Web Services Standards Ever Happen? ©



Reviewed by Joseph A. Mitchko

**About the Author:**

Joe Mitchko is a senior consultant with Phase II Consulting, a leading New Jersey firm specializing in Internet and database-related architecture.  
JMITCHKO@RCN.COM

## Orbix E2A Web Services Integration Platform, XMLBus Edition 5.0

*CORBA, Web services, and beyond*

**B**ack in the early 1980s, a popular fast-food restaurant ran a series of humorous commercials on television based on an elderly woman visiting a competing fast food chain. After receiving her order, she would pry open an oversized hamburger bun and utter a shrill complaint for all to hear: "Where's the beef?" It seemed that for a while everyone was saying it, including yours truly. Now, roll the clock forward a decade or two (sigh), and I'm having my first look at IONA's Orbix E2A XMLBus Web Services Integration Platform, XMLBus Edition 5.0. After a while I couldn't stop muttering to myself a slightly modified version of that catchy phrase – "OK, where's the XML?"

First off, please don't think there's something wrong with the product, because there isn't. I tend to be a hands-on kind of person, and I like to be able to futz with the XML code when I create a Web service. It's similar to wanting to continue to program in assembler code when a 3GL language can do it all for you in a fraction of the time.



In a similar fashion, having all of the XML programming done for you is part of an increasing trend by Web service integration platforms – you're literally just a few point-and-clicks away from deploying your own Web service. Here's what IONA has to offer:

### Overview

The XMLBus Edition 5.0 Web Services Integration Platform consists of a suite of Web service development and administration tools, including:

- Web Service Builder
- Web Services Manager
- Web Service Test Client
- UDDI Browser
- Soap Message Test Client

The suite can be installed standalone, or deployed to leading application servers, including WebLogic, WebSphere, and IONA's Orbix E2A Application Server platform. I chose to install the product on the Orbix E2A Application Server.

In addition, the platform includes a toolset for integrating various B2B Web services in a process engine framework:

- XDI Suite.
- XDI Monitor.

This feature in particular caught my interest in that it takes the whole notion of Web services to the next logical step in the evolutionary process. I'll have more on this later.

### Features

IONA XMLBus is compatible with current Web service specification levels, including:

#### SOAP 1.1 and 1.2

- Various encoding styles are supported, including document-literal, document-encoded, RPC-literal, and RPC-encoded.
- SOAP with attachments, for transporting large XML documents or binary data.

**IONA TECHNOLOGIES, PLC**

200 West Street  
4th Floor  
Waltham, MA 02451  
Phone: 1-800-672-4948  
Web: [www.iona.com](http://www.iona.com)

**PRICING:**

Licensing Information  
Developer License: \$495  
Runtime License: \$2500 / CPU  
Download a 30-day evaluation copy from:  
[www.iona.com/downloads](http://www.iona.com/downloads)

**TEST ENVIRONMENT:**

OS: Windows-XP  
Hardware: Dell Inspiron 8000



### WSDL 1.1

- Supports primitive, common, and complex data types.

In addition, XMLBus Edition 5.0 includes:

- Synchronous transport of SOAP protocol over HTTP and HTTPS Internet protocols
- Support for MS .NET toolkit and MS SOAP
- Ability to expose CORBA objects as Web services
- Client-code generation for J2ME (WAP-enabled wireless equipment) and J2SE proxy clients
- Process flow engine for long-running Web service transactions
- UDDI Browser
- A rich Java API set

### Web Service Component Model

IONA innovatively introduces the concept of a component model in their product by designing their Web services to operate within a runtime container in the application server. Similar to an EJB component, a Web service can be activated and deactivated within its container by using an administrative interface. The container provides services such as performing SOAP-

message validation against the WSDL of the containing service.

In addition, Web services are compiled into a deployable unit called an XAR file, which contains WSDL definitions, interface code, etc. One note: it would be great if Web service vendors could begin the process of developing a single standard (similar to the EJB specification) regarding a component model and deployment mechanisms.

## Web Service Builder

The Web Service Builder is a Windows-based application that includes a series of navigation and builder panes that allow you to create Web services from existing CORBA-, EJB-, and Java-based applications. A wizard guides you through the process of creating a Web service for each legacy technology. The output of the process is a deployable XAR file containing the Web service. All of the SOAP and WSDL code is generated for you.

The wizards employ an introspection mechanism that exposes the various APIs in the source enterprise applications. In the case of Java applications, the builder will list the public methods for you. All you need to do is enable the interface, choose the appropriate SOAP binding mechanism (RPC vs. document, data types), enter external libraries, and the builder takes care of the rest. In the case of EJB components, the builder will communicate with the application server (using the appropriate network protocol) and obtain the remote interface information without having to directly copy the deployment unit to the development system. In the same manner, the CORBA wizard will guide you through the process of converting a service IDL to a Web service interface (see Figure 1).

To generate Java client code for your service, the Web Service Builder will take the WSDL associated with your service and generate client stub code. One particularly nice and forward-thinking feature is that you can generate J2ME code designed for palm-sized devices.

## Web Services Manager

The Web Services Manager is a browser-based utility for monitoring Web services. You can enable or disable a service with the click of a button while running in a live production environment. With its drill-down capability, you can view the various services and ports of a Web service application, and view the underlying WSDL XML. The Web Services Manager doesn't include any instrumentation or performance-related data at the present time. You may need to wait a release or two to see it.

## Process Flow Engine

I found the Process Flow Engine to be the most interesting part of XMLBus. In a manner similar to TIBCO and webMethods, you can use the XDI Suite to assemble one or more Web service components together into a single aggregate Web service (see Figure 2). To develop the process flow, you can paste in flow-of-control components (switch, parallel operations, etc.), as well as transport components (MAIL, FTP, HTTP). What is interesting is that this is achieved without a message bus, and appears to rely on HTTP transport mechanisms to move the flow along.

## Debugging the Service

Here's where the XML was hiding out. XMLBus comes with two debugging facilities, a SOAP Message Spy and WSDL Dynamic Test Client. The SOAP Message Spy allows you to monitor the HTTP protocol request and response streams from the Web application server. In addition, you can manually enter a SOAP request and send it to the server.

The WSDL Dynamic Test Client is a browser-based utility that dynamically reads the WSDL for the service and provides a list of services and ports to test. After choosing a port, you can enter one or more parameters and directly invoke the service. A third window displays the SOAP request and response XML messages.

## Additional Tools

Another interesting feature is the UDDI Browser, which allows you to shop around for Web services listed on UDDI registered repositories (see Figure 3). Although the number of public UDDI repositories are limited at the present time, the ability to browse and integrate a Web service through a single development toolset is the wave of the future. IONA has the right idea, but should integrate the browser directly into the XDI toolset. Point-and-click service integration, anyone?

## Installation

The installation process is fairly straightforward, but not entirely foolproof. If you're installing on a Microsoft Windows-based platform, stay away from installing to a directory structure that has embedded spaces in it (i.e., "C:\Program Files"), and you should do fine. Depending on your development environment, you may need to tweak some of the CLASSPATH environment variables in the various script-based command files in the toolset. In most cases, the documentation will guide you through the necessary modifications, and shouldn't be a problem for the Java-versed. Also, make sure you know what

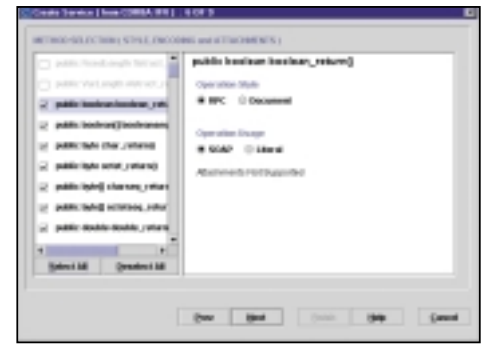


FIGURE 1 | Selecting Web services operations

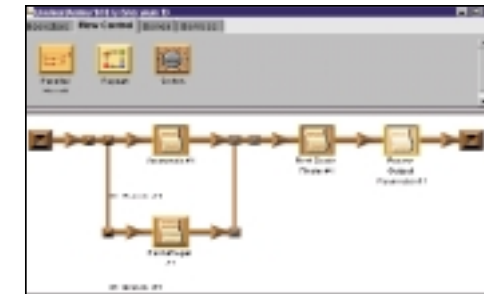


FIGURE 2 | A Process Flow



FIGURE 3 | UDDI Service Listings

IONA software prerequisites you need prior to installing. For instance, creating Web services from CORBA components will require you to install and configure IONA's CORBA Services before you can see any of the configuration features in the builder.

## Concluding Thoughts

It took a while for me to fully appreciate the capabilities of this product. Its true strengths aren't immediately apparent when initially working with the builder and admin tools. It almost seemed too easy to create a Web service. But what goes on under the surface makes XMLBus Edition 5.0 a leading contender in the family of Web service integration platforms. Given that the platform includes a process flow engine and can interface with a solid base of legacy CORBA applications, it makes it even more appealing at the enterprise level. And that's nothing to beef about! ☺



# Using Web Services with J2EE

## Moving into a technology-independent world of distributed computing

**B**y now, every Java developer and architect has heard the words "Web service." Loosely, a Web service can be described as any business enterprise asset enabled for access over the Web.



There's a need to make applications accessible independently of the technologies used in the enterprise. The goal is to gain interoperability among distributed systems spanning diverse hardware and software platforms.

SOAP and WSDL are XML-based standards that have made it possible to achieve this goal. SOAP facilitates development of heterogeneous distributed applications by providing a standardized format for transmitting information. WSDL enables Web services to describe themselves to prospective clients. Ubiquitous transport protocols such as HTTP, SOAP, and WSDL free developers from technology and vendor-imposed shackles that have been longstanding consequences of traditional service-based architectures using RMI, DCOM, and CORBA.

The interoperability afforded by Web services forms the cornerstone of any argument that suggests making them a part of B2B and EAI endeavors in enterprise collaboration. In the Java world, enterprise applications are primarily built on the J2EE platform. This article addresses Web service development and deployment in a J2EE environment, discussing various issues that arise in the process.

### Web Services vs J2EE

Web services are commonly perceived as a way to interface with back-end systems such as ERP, CRM, order-processing systems, and so on. Even so, they have great untapped potential for opening access to virtually any enterprise application for customers and partners alike.

J2EE provides a hardware-independent platform for building these distributed enterprise applications. It provides many benefits, such as a component-based development model; scalability; object life cycle management; transactional access to existing systems using JDBC, JTA, and JMS; and a unified security model. However, the J2EE specification by itself doesn't perpetuate the more open software-independent advantages of Web services. Nevertheless, it doesn't preclude the inclusion of SOAP or WSDL and the deployment of Web services in a J2EE-driven enterprise (there is currently a proposed final v0.8 of the draft JAX-RPC specification defining a standard Java API for developing and accessing Web services. Indeed, the J2EE platform is an excellent vehicle to provide the necessary plumbing for the development and deployment of Web services. Premier application server vendors such as BEA and

IBM have already delivered J2EE-based Web service solutions with their recent versions of WebLogic and WebSphere products respectively. As such, services deployed using these products inherit all the benefits of J2EE, while at the same time providing their clients with increased technology independence and Internet accessibility.

### A Web Services Opportunity

Consider the scenario in which broker ZTrade.com provides the best commissions for online trading and portal EveningSatellite.com lets registered users track portfolios online. ZTrade is a bare-bones operation. Its Web site doesn't provide any sophisticated stock screening or asset-management tools. As an added benefit to its clients, ZTrade decides to forge an alliance with EveningSatellite so it can create and maintain portfolios at the portal's site. ZTrade users can simply log on to EveningSatellite and perform various research tasks on their portfolios. ZTrade's basic requirement: the ability to update client portfolios at EveningSatellite in real time as trades are executed.

EveningSatellite needs to make its portfolio-tracking application accessible to enterprise applications of partners such as ZTrade. Together, both parties can develop an efficient proprietary solution. However, in the future, when EveningSatellite wants to offer similar services to other partners, the wheel will need to be reinvented, given the diverse technologies used by prospective partners. Deploying a portfolio-management Web service is the ideal solution for EveningSatellite.

A simplified view of the J2EE-driven EveningSatellite Web site is shown in Figure 1, with the requirement of a Web service included. The interface from the Web tier to the EJB tier is primarily through two stateless session beans specified by UserManager (see Listing 1; all of the source code referenced in this article may be found at [www.sys-con.com/web-services/sourcec.cfm](http://www.sys-con.com/web-services/sourcec.cfm)) and PortfolioManager (see Listing 2). As the names suggest, UserManagerSession



#### Author Bio

Saurabh Dixit is a senior technology partner at Axy Solutions, a professional services consulting firm based in Dallas, specializing in IS architecture, Java enterprise development and Web services.  
SDIXIT@AXYSOLUTIONS.COM

SHOP ONLINE AT **JDJSTORE.COM** FOR BEST PRICES OR CALL YOUR ORDER IN AT **1-888-303-JAVA**

**BUY THOUSANDS  
OF PRODUCTS AT  
GUARANTEED  
LOWEST PRICES!**

**GUARANTEED BEST PRICES  
FOR ALL YOUR  
WEB SERVICES  
SOFTWARE NEEDS**



### MICROSOFT

**\$2,238.99 Visual Studio .NET Enterprise Architect**

Visual Studio .NET provides developers with the most productive tool for building next-generation applications for Microsoft Windows® and the Web. Visual Studio .NET Enterprise Architect (VSEA) builds on the power of Visual Studio .NET Enterprise Developer by including additional capabilities for designing, specifying, and communicating application architecture and functionality. It enables software architects and senior developers to provide architectural guidance and share best practices across the development team.



### SYS-CON MEDIA

**\$79.00 Web Services Resource CD**

The Most Complete Library of Exclusive WSJ and XML-J Articles on one CD! This CD is edited by well-known WSJ Editor-in-Chief Sean Rhody and organized into more than 40 chapters containing more than 400 exclusive WSJ & XML-J articles.  
\$100 Coupon Inside for Web Services Edge Conference & Expo!



### MICROSOFT

**\$1,629.99 Visual Studio .NET Enterprise Developer**

With Visual Studio .NET Enterprise Developer, developers can securely version and share their source code, share best practices, target scalable .NET Enterprise Servers, choose from a wide range of third-party tools and technologies, and easily tune the performance of their Web applications and Web Services through the extensive performance-testing tools in Visual Studio .NET.



### WHIZLABS

**\$39.95 WebSphere@Whiz Certification Simulator**

3 Mock Tests (159 Questions). It comes with a test engine and a question bank of 159 questions on the latest pattern of the IBM WebSphere Certification Exam. It also consists of a diagnostic test which will help you know your strengths and weaknesses so you can plan your preparation accordingly.



### ALTOWEB

**\$3,540.00 Application Platform Release 2.5**

The AltoWeb Application Platform lets you build, deploy, and manage J2EE applications and Web services up to 10x faster without requiring extensive J2EE or Web services expertise. How? By replacing lengthy, custom and complex J2EE, XML and Web services coding with rapid component assembly and reuse.



### SILVERSTREAM

**\$495.00 Extend Application Server Developer Edition (5 User)**

SilverStream eXtend is the first comprehensive, real-world development environment for creating Web Services and J2EE applications. The seamless integration of our proven eBusiness engines and designers gives you the benefits of XML-based, enterprise-wide integration and the power to create, assemble and deploy service-oriented applications.



W W W . J D J S T O R E . C O M

OFFERS SUBJECT TO CHANGE WITHOUT NOTICE

**SAVE UP TO \$100 ON MULTIPLE SUBSCRIPTIONS**

**Special  
online offers**

Pick **4** or **5** and Subscribe  
for one **special low price**



**RECEIVE YOUR  
DIGITAL EDITION  
ACCESS CODE  
INSTANTLY  
WITH YOUR PAID  
SUBSCRIPTION**

Wireless Business & Technology • Java Developer's Journal

Web Services Journal • WebLogic Developer's Journal

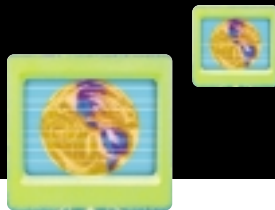
XML-Journal • WebSphere Developer's Journal

ColdFusion Developer's Journal • PowerBuilder Developer's Journal

**SYS-CON  
MEDIA**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

**WWW.SYS-CON.COM/SUBOFFER.CFM**



handles client user registration, authentication, and related tasks, while PortfolioManagerSession provides portfolio creation and maintenance functionality. The former makes use of the latter when required, for example, to purge the portfolio of a user that is unregistered. Obvious methods provided by these interfaces, such as getPortfolioDetails() in PortfolioManager and changePassword() in UserManager, have been excluded for clarity.

## Enterprise JavaBeans as Web Services

Let's assume WebLogic 6.1 is being used as the application server and Web services platform. Any stateless session bean in Web Logic

EveningSatellite back-end systems of interest to the partners. ZTrade can verify a user exists when a client provides a userId, or create a user at EveningSatellite and provide the password to the client. Methods such as changePassword() in UserManager are absent from the service interface for good reason.

For the sake of simplicity, PartnerService method signatures only use primitive data types. Due to differences in the level of support for user-defined types across various implementations today, it may be best to rely on simple data structures and primitive data types. In B2B and EAI scenarios, XML is the format of choice for communicating information. This XML can be passed as a string, rather than say, XML-node data structures, to

JAR that can be downloaded).

Let's examine the implementation of the *Create User* use case for our service (see Listing 4). It looks up the home interface for UserManager, creates an instance of the session bean, and simply invokes the appropriate method on it. An application exception User ManagerException, thrown from UserManager, is treated as a system exception (for illustrative purposes only) and rethrown as an EJBException. WebLogic translates the resulting RemoteException thrown by the EJB container into a SOAP fault in the outgoing SOAP message.

## Web Service Deployment

It should be no surprise that Web services are deployed using a servlet in a Web application provided by a given implementation. WebLogic uses the WebLogic.soap.server.servlet.Stateless Bean Adapter servlet. As the name suggests, it adapts incoming SOAP message invocations to methods on stateless session beans. WebLogic ships with an Ant (<http://jakarta.apache.org/ant>) task "ws-gen" that facilitates deployment of a Web service as an application archive (EAR) containing the servlet in a Web archive (WAR) and the stateless session bean for the service in an EJB archive (JAR). The items of most interest in the Ant build script (see Listing 5) are the name of the service bean, ejb/Partner Service, the context partners that form part of the WSDL URL, and the EJB archive name, portfolio.service.jar, containing the service bean. For detailed information on developing Web services with WebLogic, refer to <http://e-docs.bea.com/wls/docs61/webServices/index.html>.

## Web Service Invocation

CreateUser (see Listing 6) shows how a service operation is invoked using the WebLogic client API. As mentioned earlier, clients can download a JAR from a Web page available for the Web service on deployment. The WSDL URL used follows this pattern:

```
http://[host]:[port]/[context]/[JNDI-Name]/[JNDI-Name].wsdl
```

Note that the JNDI name for the service bean is specified in the WebLogic EJB deployment descriptor weblogic-ejb-jar.xml (see Listing 7) for portfolioservice.jar.

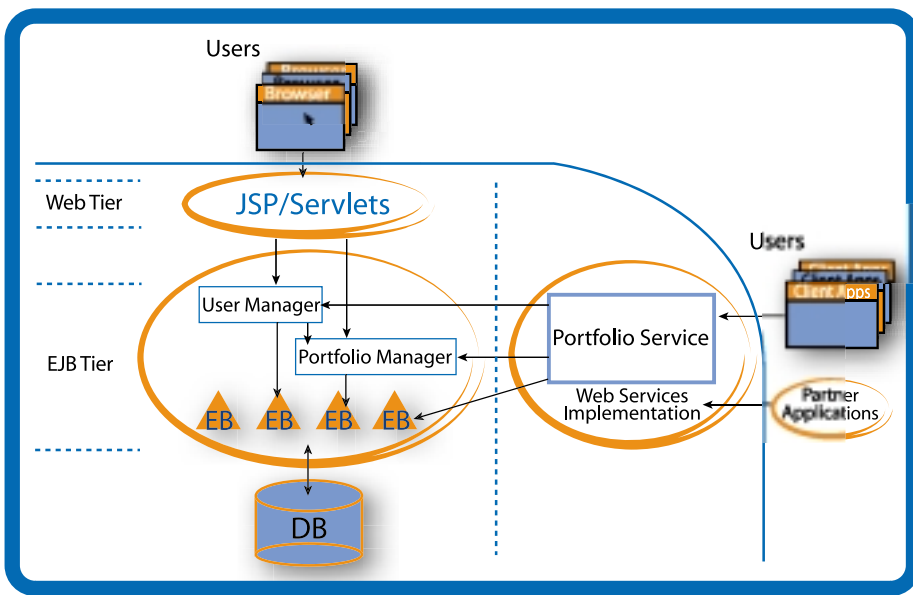


FIGURE 1 | A simplified view of the J2EE-driven Evening Satellite Web site

can be easily exposed as an RPC-style service (support for message-style services is also available but is outside the scope of this article). The interface for the EJB doubles as the interface for the service. PartnerService (see Listing 3) and PartnerServiceSession (see Listing 4) provide the interface and implementation for our service.

It would have been just as easy to expose both UserManagerSession and PortfolioManagerSession as two individual services. Such session beans are viewed as facades into an enterprise application. The interfaces from such facades do not necessarily translate to interfaces required of Web services. PartnerService provides a much-needed abstraction that exposes only those aspects of the

ensure all implementations can handle it.

The practice of making method signatures independent of artifacts used in the underlying application can go a long way toward decoupling systems and increasing technology options. The SecurityInfo JavaBean supplied to PortfolioManager.updateSecurity() was perceived as an internal implementation detail and thus not exposed in the Web service interface.

Implementations typically provide facilities for generating the WSDL for a service and generating classes or data structures for types described in a given WSDL (WebLogic takes a slightly different tack for providing the latter functionality; it packages its client-side API with any other service-dependent classes in a



## Exception Handling

When using the WebLogic client API, any exception generated by the service manifests itself as a `weblogic.soap.Soap Fault` exception at the client. If a user, "tom", exists in the system, then `CreateUser` dumps the following, including the SOAP fault to stdout.

```
User tom NOT created!
Fault Code: Client
Fault String: java.rmi.RemoteException
Fault Detail: Application fault:
java.rmi.RemoteException: EJB
Exception: ; nested exception
is:javax.ejb.EJBException
- with nested exception: [com.axysso-
lutions.ejb.UserManagerException: User
tom already exists!]
(deeper stack trace omitted)
```

WebLogic uses the class name of the exception thrown by the EJB container, in this case `java.rmi.RemoteException`, as the fault string and throws in the stack trace as the fault detail for good measure. Ordinarily, it should be possible for a service to specify information for these fault attributes. This is explored in more detail later.

## To EJB or Not to EJB

Most J2EE-based implementations use servlets to map incoming SOAP messages to invocations on a Web service. In this context, a "Web service" is the physical manifestation of executing code. In WebLogic, it happens to be a stateless session bean activated in its EJB container. But this isn't always the case in other implementations. For instance, WebSphere and GLUE (from The Mind Electric) also allow a simple `JavaBean` or class to act as the Web service as well.

Discovering which operations a stateless session bean exposes is a matter of inspecting its remote interface. When a Java class is used to represent the service, its public methods can

be introspected under the assumption that all of them are part of the service interface. Either way, these methods, with their signatures, can be specified as operations in the corresponding WSDL. One advantage of using a Java class is that only one instance must be created when the service is stateless.

The WebLogic implementation leverages its EJB container to deliver Web services as stateless session beans. This is a very reasonable approach that yields standard J2EE deployment and configuration semantics with its deployment descriptor and JNDI naming context. It also imparts an important benefit to Web services – declarative transaction management. Transaction behavior of an EJB can be controlled using its deployment descriptor, which sets transaction attributes for individual methods. Thus, the transaction context for a service session bean method can be propagated to one and all beans used within the method. This is especially useful if an application session bean can be used as-is for the service.

Other factors can influence a given Web service implementation. It's reasonable to expect that business partners will want to communicate information using XML. It's equally likely that some legacy client application generates data feeds consisting of name-value pairs. Given that an application is already deployed for traditional Web-based access, developing a session bean for the sole purpose of parsing input to feed application beans may be overkill. You wouldn't normally want to deploy parsing logic in the EJB Tier, which should ideally focus simply on the business aspects.

In the above situation, a Java class deployed for access through a Web application may be a better alternative. The service could then be pushed out to the Web tier. This could lead to more efficient use of resources – especially when multiple stateless services are needed – as it translates to reduced object creation. All

invocations on the service can be mapped to a single class instance rather than a session bean selected from a pool. Besides, a Java class is a more manageable undertaking than a session bean with its remote and home interfaces, and implementation class. Using a Java class can lead to a reduced number of deployment issues.

Another situation worth exploring is when the functionality to be exposed doesn't use any session beans. Consider a set of use cases, each requiring access to only one entity bean. The application may then dispense with the additional level of indirection available through a session bean. Given that a session bean is not readily available, you now have to decide whether to go with a stateless session bean or a plain Java class. Sometimes, it's more natural to view a Web service as a Web application component that's a client of beans in the EJB tier, rather than part of it.

Using a Java class for the service is not without its drawbacks. We lose declarative transaction control for the service methods, which may or may not be an issue. It depends on how many beans are explicitly accessed by a service method. If the point of access into the application for a given method is through one bean only, then its descriptor can be used for managing the transaction. If multiple beans need to be accessed, it's the Web service's responsibility to explicitly demarcate transactions (see Listing 8). Other J2EE benefits unavailable when using a Java class include explicit role-based access control, deployment, and configuration facilities.

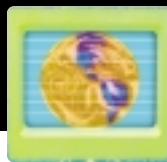
A Java class may be a better option for a read-only service that generally doesn't need transactional control. A `StockQuote` service doesn't have to be transactional, unlike a `StockTrader` service, for which a stateless session bean may be more appropriate.

## Matters of State

As we saw earlier, it's possible to deploy stateless session beans as Web services. One question that begs asking is: Why use a stateless rather than a stateful session bean?

A Web service provides an interface with a unified set of operations exposing some functionality. Unlike HTTP, SOAP doesn't provide any artifacts that help in managing session state across multiple invocations. SOAP headers can conceivably be used to implement a stateful model for a service. One may implement a custom solution by passing a session-ID in the SOAP headers. Unless such headers

**// The practice of making method signatures independent of artifacts used in the underlying application can go a long way toward decoupling systems and increasing technology options"**



are standardized in the SOAP specification, solutions using them won't be vendor-neutral.

State management is a feature that maps naturally to technologies used in the Web tier. It can be implemented quite efficiently with HTTP session management support provided by the servlet API, with the service itself remaining stateless. Though most implementations use HTTP to send and receive SOAP messages, other protocols, such as SMTP, may also be employed. If the service is to behave independently of the transport protocol used to access it, then stateful solutions are best implemented within the Web service infrastructure. This may involve using an application server's proprietary features in the absence of standards for state management in Web services.

Regardless of which protocol is used, it's possible to implement Web services to expose the standard J2EE state-management modes of "application", "session", and "request". WebLogic doesn't provide any functionality for implementing stateful Web services. Examples of platforms that do so (to varying degrees) are WASP, from Systinet, and GLUE.

When the application mode is used, all invocations are mapped to a shared implementation instance for the service. In this case, the implementation can simply map to a singleton instance of a Java class or to a stateless session bean. In the latter case, any application state would have to be persisted in in-memory caches or a database. When the request mode is used, an invocation readily maps to a stateless session bean or to a Java class instance. A new (or pooled) instance is used to handle each invocation. When the session mode is used, an invocation maps to either an HTTP session that passes required state information to a stateless session bean or Java class instance, or to a stateful session bean. The latter alternative is obviously the logical choice when a protocol other than HTTP is used.

Developers can reasonably expect support for state management modes to be a feature provided by a given implementation and typically shouldn't have to implement it.

## Customizing Web Service Errors

At the time of writing, the WebLogic implementation lacks support for enabling a Web service to include application-centric errors in the SOAP fault, and more importantly, the fault detail. The implementation simply uses the exception stack trace for the fault detail. When specific error information needs

to be communicated to clients, other implementations such as Apache SOAP or GLUE can be plugged into WebLogic (or any other J2EE-compliant server of choice).

Let's examine how to implement an Asset Service Java class for our service using GLUE 1.x. The AssetService methods are essentially similar to those of Partner ServiceSession, but with one notable difference. They all throw an `electric.net.soap.SOAPException`. GLUE takes any exception thrown from a service and makes it available at the client as an instance of SOAP Exception. To explicitly communicate a fault to a client, the service can throw an instance of SOAPException that GLUE simply echoes at the client.

The `AssetService.createUser()` implementation (see Listing 9) assumes that when an application exception is thrown from the `UserManagerSession`, it is because the specified user already exists. It then suggests alternative `userIds` to the client using the fault detail. Thus, the `CreateUser` version for GLUE (see Listing 10) dumps the following SOAP fault to stdout if user tom already exists in the system.

```
User tom NOT created!
Fault Code: Client
Fault String: User tom already exists!
Fault Detail: <detail>
  <userId>tom101</userId>
  <userId>tom102</userId>
  <userId>tom103</userId>
</detail>
```

GLUE can deploy Web services using its own Web server or in any J2EE-compliant application server. The source code accompanying this article was deployed using GLUE in WebLogic. GLUE can wrap incoming invocations to either a Java class or a stateless session bean. A Web application containing the `electric.servlet.http.ServletServer` servlet has to be configured with the appropriate properties. For detailed information on implementing Web services with GLUE, see [www.themindelectric.com/products/glue/does/guide/index.html](http://www.themindelectric.com/products/glue/does/guide/index.html).

## Securing Web Services Confidentiality

SSL over HTTP can be used for encrypting invocations on Web services. The standard JSSE (Java Secure Socket Extension) libraries can be readily used. In GLUE, for example, the `standard.javax.net.ssl.trustStore` and `javax.net.ssl.trustStorePass` word properties must be set

to the location and password of the certificate file. To bind to a service deployed securely, the WSDL URL specified by the client should begin "https:" instead of "http:". Appropriate with steps must be taken to deploy the underlying service applications securely at the back end.

## Authentication

HTTP-based authentication is a popular method for restricting access to resources on a Web server. The SOAP client sends the username and password as a base 64-encoded string in the appropriate HTTP header. Certificate-based authentication may also be used if required. However, HTTP authentication is the least common denominator supported on all server platforms and is easily implemented over SSL. Custom authentication is possible using SOAP headers that can be verified by the Web service implementation.

## Authorization

Authorization can be implemented in the same way as for any J2EE application. Once a client has been authenticated using HTTP-based authentication, authorization can be performed in the standard J2EE realm/role-based manner.

## Conclusion

Web services are moving the Internet toward a service-oriented architecture in which client applications and Web services can use other Web services on demand. With the vast number of J2EE applications already deployed, it makes sense to expose their interfaces as Web services. This is especially important, not to mention convenient, since it can all be done while retaining the benefits of J2EE.

Various J2EE application server vendors are already shipping Web services-enabled products. SOAP and WSDL support can vary greatly from one product to another. As we've seen, however, the J2EE architecture makes it possible to plug in custom and off-the-shelf components to yield optimally configured solutions. Web services not only coexist with applications executing within J2EE containers; they open up a whole new client space for these applications.

## Acknowledgement

I would like to thank Rhett Guthrie at Axy Solutions for his guidance. ☺

Now in More than 5,000 bookstores worldwide

**FORFAST**  
DELIVERY

subscribe **Now!**

Go  
Online  
and  
Subscribe  
Today!

The World's Leading  
Independent WebLogic  
Developer Resource  
FOR WLS DEVELOPERS BY WLS DEVELOPERS  
**WebLogic**  
DEVELOPER'S JOURNAL  
An introduction to...

Helping  
you enable  
inter-company  
collaboration  
on a global scale

- Product Reviews
- Case Studies
- Tips, Tricks and more!

**WebLogic**

**Journal.com**

SYS-CON Media, the world's leading publisher of *i*-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of WebLogic.

\*Only \$149 for 1 year (12 issues) regular price \$180.

**SYS-CON**  
MEDIA

**EnginData Presents**

# 2002 Developer Market Survey Reports



Our comprehensive reports offer insight and strategy to guide your most critical business decisions in today's fastest growing technologies...

- ✓ Establish your product and marketing strategy
- ✓ Understand your customer's needs
- ✓ Evaluate Technology & Trends

Preview and order reports at [www.engindata.com](http://www.engindata.com)

**engindata.com**

**engindata** ✓  
RESEARCH



**WebServices**  
JOURNAL

**XML** JOURNAL

THE FIRST & ONLY  
**WEB SERVICES**  
RESOURCE CD

# WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

INCLUDES EXCLUSIVE .NET ARTICLES

MORE THAN

**400**  
EXCLUSIVE

WEB SERVICES  
& XML  
ARTICLES



EDITED BY  
SEAN RHODY

**\$119**  
CD  
**VALUE**

FROM  
WEB SERVICES  
JOURNAL

Web services **EDGE** \$100  
conference & expo coupon inside

EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

# THE MOST COMPLETE LIBRARY OF EXCLUSIVE WSJ & XML-J ARTICLES ON ONE CD!

## "The Secrets of the Web Services Masters"

CD is edited by well-known WSJ Editor-in-Chief  
Sean Rhody and organized into more than 40 chapters  
containing more than 400 exclusive WSJ & XML-J articles.

**Easy-to-navigate HTML format!**

### Bonus:

Full .PDF versions of every WSJ & XML-J published  
since the first issue

XML in Transit	XML Industry	XML &	UML
XML B2B	Insider	Databases	Integration
Java & XML	<e-BizML>	Electronic Data	WSDL
The XML Files	XML & Business	Interchange	Beginning
XML & WML	XML Demystified	Ubiquitous	Web Services
Voice XML	XML &	Computing	Web Services
SYS-CON Radio	E-Commerce	Information	Tips & Techniques
XML & XSLT	XML Middleware	Management	Frameworks
XML & XSL	XML Modeling	Objects & XML	Management
XML & XHTML	CORBA & XML	XML Pros & Cons	Security
2B or Not 2B	Data Transition	Java Servlets	UDDI
XML Script	XML @ Work	XML Filter	.NET

3  
YEARS  
25  
ISSUES  
400  
ARTICLES  
ONE CD



Special Limited Time Price

Now  
Shipping

\$79

+ S&H

**ONLINE**  
ORDER AT  
**JDJSTORE.COM**  
**SAVE**  
**\$40**

[WWW.JDJSTORE.COM](http://WWW.JDJSTORE.COM)

OFFER EXPIRES JUNE 30, 2002



## Bind Systems Turns Web Services Design Upside Down

(Dublin, Ireland) – Bind Systems Ltd, an innovator in the area of open Web services–based technology, announces BindStudio, the product that turns Web services design upside down.

With leading Web services platforms rapidly maturing support for foundational Web services technologies Bind Systems now offers a powerful solution for business to maximize return on these infrastructure investments.

The BindStudio environment lets business requirements drive Web services technology solutions by providing a seamless bridge between business process modeling and Web services application development and runtime environments.

This top-down approach to Web services EAI and B2B enables the development of solutions that effectively implement business decisions. BindStudio generates WSDL and ebXML descriptions directly from unambiguous models of real business requirements that can then be deployed and executed on third-party Web services platforms.

BindStudio v1.1 is available now for download and evaluation.

[www.bindsys.com](http://www.bindsys.com).



## Kamiak Announces Omniopera 1.0, A WSDL and XML Schema Editor

(Sheridan, WY) – Kamiak Corp., which publishes tools for developing Web services, has announced the release

of Omniopera. Omniopera is the first tool designed specifically to allow easy creation of Web service interfaces. It complements any Web service development tool that allows its users to develop a Web service starting from WSDL, such as Borland's Delphi 6.0, Microsoft's Visual Studio .NET, Apache's Axis, IBM's Web Services Toolkit, and Systinet Corporation's WASP.

Omniopera WS 1.0 is a WSDL and XML schema editor that allows you to define Web service interfaces using an intuitive graphical interface. It generates WSDL that is compatible with most Web service development platforms.

[www.omniopera.com](http://www.omniopera.com)

## F5 Networks Delivers Complete Support for Microsoft .NET

(Seattle) – F5 Networks, Inc. a provider of Internet traffic and content management (iTCM) products, has announced that it's the first iTCM vendor to offer complete interface support for the Microsoft .NET platform and XML Web services. All



applications and services developed using Microsoft Visual Studio .NET and the .NET Framework can communicate with and influence the behavior of

the underlying network through F5's iTCM products and iControl API.

F5 ensures that applications work in concert with the network, enabling rapid deployment and successful delivery of XML Web services.

The iControl SDK (software developer's kit) can be downloaded for free at [www.f5.com/iControl/developer/index.html](http://www.f5.com/iControl/developer/index.html).

## Actional Utilizes .NET for Deployment of Enterprise-class Web Services

(Mountain View, CA) – Actional, a provider of cross-platform XML Web services infrastructure software, has been named one of 12 Microsoft .NET Emerging Business Partners.

Actional's groundbreaking Web services product, SOAPswitch – the Web Services Gateway, builds on that expertise to help organizations more easily accelerate their move into XML Web services – and benefit from XML-based transactions.

SOAPswitch works by residing within an organization's IT infrastructure, and allowing internal staff, customers, or suppliers to access existing packaged, legacy, and custom-built applications such as accounting, inventory, or booking via Internet technology within a centralized security and management model. With SOAP switch, companies can introduce and expand their Web service offerings while centrally maintaining their choice of robust security and management capabilities across the enterprise.

[www.actional.com](http://www.actional.com).

## Software AG And T4 Consulting Group Announce Strategic Alliance

(Reston, VA) – Software AG, Inc., the U.S. subsidiary of Software AG, a systems software provider and pioneer in XML technologies, and T4 Consulting Group, Inc. (T4CG), a provider of Java



and XML-based solutions, have announced a strategic alliance to deliver a complete solution leveraging Software AG's product suite and T4CG's XML integration experience. T4CG will provide XML integration capabilities with Software AG's Tamino native XML server and EntireX(TM) enterprise application integration products for its clients.

Under this agreement, Software AG and T4CG will actively market the combined solution. The benefit of the alliance is that enterprise customers can rapidly access and display information from multiple, heterogeneous sources and data formats. Customers benefit from this type of integration because they can access a wide variety of enterprise data from a single application rather than needing to learn multiple systems for each data source.

[www.t4cg.com](http://www.t4cg.com), [www.softwareagusa.com](http://www.softwareagusa.com)

## PolarLake and SpiritSoft Deliver Messaging-based Solutions

(Boston) – PolarLake, the enterprise XML and Web services company, has announced a strategic partnership with SpiritSoft, a leader in open-framework Java messaging. This alliance provides enterprises with a complete platform for creating and deploying messaging-based XML and Web services solutions.

The PolarLake development and deployment platform combines with SpiritSoft's messaging technology to offer a scalable, standards-based, business process integration solution. The combined solution allows Web services and XML offerings to be integrated quickly with legacy systems, and greatly simplifies the integration of highly complex workflow and business processes. This is applicable to a wide range of business problems including B2B e-commerce, straight-through processing in the financial services industry, and more general application integration issues faced by enterprise architects.

[www.polarlake.com](http://www.polarlake.com),  
[www.spiritsoft.com](http://www.spiritsoft.com)





# WebServices JOURNAL

.NET J2EE XML

COMING IN THE  
May ISSUE

## FOCUS ON EAI

- e** Web Services for Enterprise Application Integration  
What solutions are offered for enterprises today?
  - e** Powering Web Services through Integration Technology  
Agreement on standards will lead to a new era
  - e** J2EE, EAI, and Web Services  
New challenges for your information systems
  - e** Will Web Services Mean the End for EAI?  
Complementary approaches for peaceful coexistence
- Plus*
- e** Data: A Key Part of Web Services  
Keep the fundamentals for current environments and tools

## WSJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Actional	www.actional.com/soap2	800-808-2271	6
Altova	www.altova.com		23
engindata Research	www.engindata.com		61
Hewlett-Packard Company	www.hpmiddleware.com/download	856-638-6000	67
Dynamic Buyer	www.ibm.com/smallbusiness/dynamicbuyer	800-426-7235	41
iWay Software	www.iwaysoftware.com/guaranteed	866-297-4929	15
Java Developer's Journal	www.sys-con.com	800-513-7111	37
JDJ Edge Conference & Expo	www.sys-con.com	201-802-3069	35
JDJ Store	www.jdjstore.com	888-303-JAVA	57, 62, 63
Sams Publishing	www.sampublishing.com		11, 25
Simplex Knowledge Company	www.sk.com	845-620-3700	45
Sitraka	www.sitraka.com/jclass/ws		19
Sonic Software	www.sonicsoftware.com		2, 3
SpiritSoft	www.spiritsoft.com/climber		4
SYS-CON Media	www.sys-con.com	800-513-7111	57
Web Services Edge Conference & Expo	www.sys-con.com	201-802-3069	31
Web Services Edge World Tour 2002	www.sys-con.com	201 802-3069	32,33
Web Services Journal	www.sys-con.com	800-513-7111	51
WebLogic Developer's Journal	www.sys-con.com	800-513-7111	61
Wireless Business & Technology	www.sys-con.com	800-513-7111	39
XML Edge Conference & Expo	www.sys-con.com	201-802-3069	27
XML Global Technologies	www.xmlglobal.com/newangle	800-201-1848 ext. 210	68
XML-Journal	www.sys-con.com	800-513-7111	47

Advertiser is fully responsible for all financial liability and terms of the contract executed by their agents or agencies who are acting on behalf of the advertiser.

SYS-CON Media, the world's leading publisher of i-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of WebSphere.



**WebSphereDevelopersJournal.com**

**WebSphere**  
DEVELOPER'S JOURNAL



**Introductory  
Charter Subscription**  
**SUBSCRIBE NOW AND SAVE \$31.00  
OFF THE ANNUAL NEWSSTAND RATE**  
**ONLY \$149 FOR 1 YEAR (12 ISSUES) REGULAR RATE \$180**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

**Do You Have  
Access to the  
Internet?**

The  
World's  
Leading  
Independent  
WebSphere  
Developer  
Resource

**Then  
Subscribe  
Online and  
Save \$31!**  
**It's that easy**

# Standard Java APIs for Web Services

## *Writing portable SOAP applications*

WRITTEN BY



**Anne Thomas Manes**

Anne Thomas Manes is the CTO of Systinet, a Web services infrastructure company. Anne is a recognized industry spokesperson and has published on a range of technology issues.  
ATM@SYSTINET.COM

**H**ave you played with SOAP yet? If so, which SOAP implementations have you used so far? Even though SOAP is on its way to becoming a standard, one thing you'll notice is that every SOAP implementation has a different programming interface. SOAP only defines the format of the messages that are sent across the wire. It doesn't define the way that a Java application interacts with SOAP. What this means is that when you build a SOAP application, you have to select your SOAP implementation up front. There are no SOAP tools available that allow you to write a generic SOAP application that can use any SOAP implementation.

So the Java Community Process (JCP) is developing a set of standard Java APIs for Web services. The theory is that all Java SOAP implementations will support these APIs, so you should be able to write a portable SOAP application. There are two APIs in development: the Java API for XML Messaging (JAXM) and the Java API for XML-based RPC (JAX-RPC). Both APIs support both RPC-style and document-style services. A more accurate way to distinguish the two APIs is to describe JAXM as a low-level SOAP API, and JAX-RPC as a high-level WSDL-based SOAP framework.

### JAXM

The JAXM API consists of two packages. The `javax.xml.soap` package provides a low-level Java abstraction for the core elements of a SOAP message. You use this package to create a SOAP message and to construct its elements. You also use this package to create a connection to a SOAP service and to call that service, passing the constructed message. The call method blocks until a response is returned. The `javax.xml.messaging` package provides a slightly higher-level abstraction for SOAP messaging, which includes a `MessageFactory` object that you use to construct a SOAP message structure (the envelope, header, and body) based on a predefined profile. Once the message structure has been constructed, you populate the message with your payload data. When your message is ready, you connect to your service and send the message using either the asynchronous send method or the synchronous call method.

### JAX-RPC

JAX-RPC provides a very high-level abstraction of SOAP communications. For the most part, the developer never works with any SOAP constructs. Instead, JAX-RPC provides a remote method invocation-style interface. JAX-RPC uses a WSDL file to generate the code that maps an RMI call to its SOAP message representation. JAX-RPC supports both RPC-style and document-style operations. In both cases, JAX-RPC

is responsible for mapping all Java data types to XML data types, and for marshalling the method parameters into the SOAP message. JAX-RPC also provides access to the low-level SOAP constructs using the `javax.xml.rpc.soap` package, which relies on the JAXM `javax.xml.soap` package.

JAX-RPC defines three different client APIs. The generated stub API relies on a WSDL-to-Java mapping tool to generate a proxy object that implements the SOAP interface code. This proxy object, which is bound to a specific binding and transport, is included in the client application. The dynamic proxy API generates the client proxy object from the WSDL at runtime. A dynamic proxy binds to the specified binding and transport at runtime. Both the generated stub and dynamic proxy APIs require the client application to know capabilities of the service at development time. The dynamic invocation interface (DDI) API allows an application to access a previously unknown service. It reads a WSDL file at runtime and dynamically constructs method invocations.

### API Readiness and Adoption

The JAXM 1.0 specification was released in November, 2001. To date there are two SOAP implementations that provide some level of support for this specification: the JAXM reference implementation (included in Sun's Java XML Pack), and Systinet WASP. Both are available as early-access releases. The JAX-RPC API was still in development as of March, 2002. The most recent release of the specification, 0.7, was published in February. To date there are two SOAP implementations that provide some level of support for this specification: the JAX-RPC reference implementation (included in Sun's Java XML Pack) and Apache Axis. The JAX-RPC RI is an early-access release. Apache Axis is available as an alpha release. Given that the specification is still changing, you should expect it to take a few months before any of the other Web service platform vendors adopt JAX-RPC.

Personally, I'm a big fan of JAX-RPC, and I'm looking forward to its completion. It would be nice, though, if the spec stopped at the client APIs and didn't attempt to define behaviors on the server side. For example, JAX-RPC defines a general-purpose mechanism called a message handler that allows you to intercept SOAP processing to bypass, alter, or add processing behavior. While it's necessary to have a mechanism to intercept SOAP processing, this "back door" approach is pretty crude and not particularly useful. First, it restricts the architectural design of a SOAP implementation, and second, the handlers are so completely undefined that they will never be portable across SOAP implementations. ☹

# **Hewlett-Packard Company**

**[www.hpmiddleware.com/download](http://www.hpmiddleware.com/download)**



# **XML Global Technologies**

**[www.xmlglobal.com/newangle](http://www.xmlglobal.com/newangle)**