

WebServices

.NET J2EE XML JOURNAL

WSJ2.COM

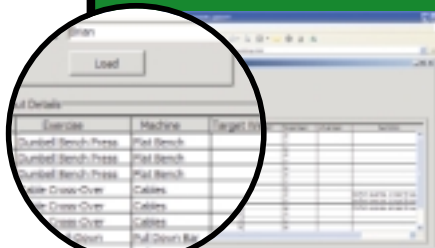
WEST
Web Services Edge 2003

web services **EDGE**
conference & expo

SEPT. 30 - OCT. 2, 2003
Santa Clara, CA

From the Editor
Service with a Smile
by Sean Rhody pg. 3

Product Review
WebFace & WebFace Studio by Vultus
by Brian Barbash pg. 50



Industry Commentary
Federated Identity Management Addresses E-Business Challenges
by John Worrall
& Jason Rouault pg. 58

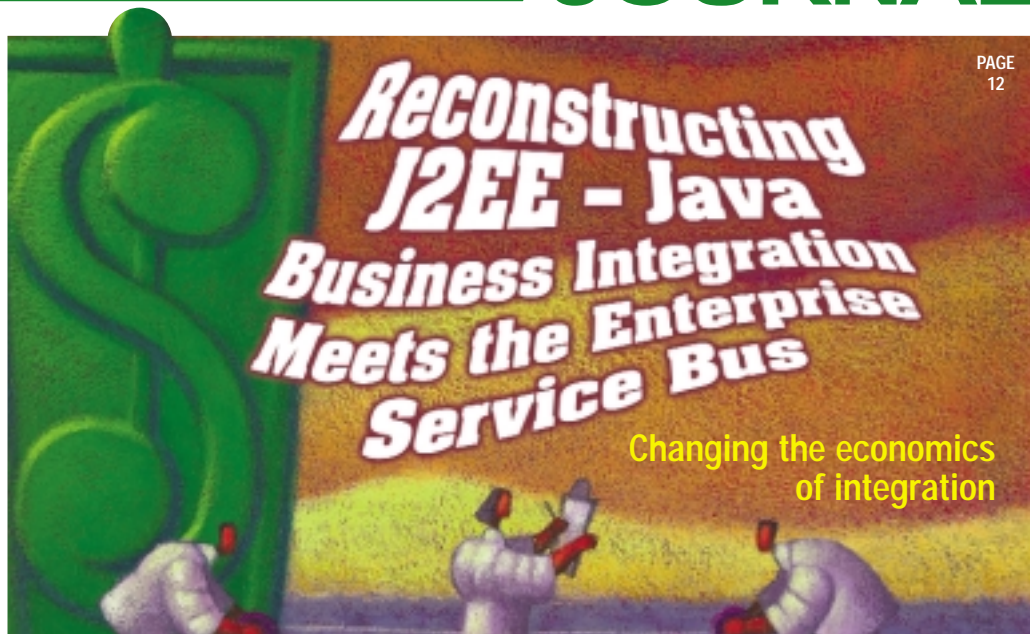
RETAILERS PLEASE DISPLAY
UNTIL JULY 31, 2003

\$6.99US \$7.99CAN






06>

0 71486 03420 9

SYS-CON MEDIA



FOCUS ON SOA

- **Strategies for Achieving Real-Time Enterprise Application Integration** *On the path to a service-oriented norm*  **Bob Zurek 6**
- **Developing J2EE 1.4 Web Services on the Fly** *Providing more design time*  **Arulazi Dhesiaseelan 22**
- **Intelligent Architectures for Service-Oriented Solutions** *Moving toward a dramatic reduction in cost & time*  **Steve Bailey & Abdul Kayam 36**
- **Unlock the Power of the Mainframe** *Leveraging CICS as a Web services hub*  **Paul Roth 44**
- **Managing the Impact of Change in an Enterprise Web Services Network** *A rapid response system for change*  **James Phillips 52**

Security: The Differentiation of Web Services Security *How can you leverage your investment in enterprise security?*  **Dave Stanton 18**

WSJ Feature: Introducing WS-Transaction, PART I *The basis of the WS-Transaction protocol*  **Dr. Mark Little & Dr. Jim Webber 28**

First Look: BEA WebLogic Workshop 8.1 *J2EE-based Web services development made easy*  **Joseph A. Mitchko 34**

IBM

www.ibm.com/websphere/seeit

INTERNATIONAL ADVISORY BOARD

Andrew Astor, David Chappell, Graham Glass, Tyler Jewell,
Paul Lipton, Anne Thomas Manes, Norbert Mikula,
Frank Moss, George Paolini, James Phillips, Simon Phipps

TECHNICAL ADVISORY BOARD

Steve Benfield, Bernhard Borges, JP Morgenthal, Andy Roberts,
Ajit Sagar, Michael A. Sick, Simeon Simeonov

EDITORIAL

EDITOR-IN-CHIEF

Sean Rhody sean@sys-con.com

EDITORIAL DIRECTOR

Jeremy Geelan jeremy@sys-con.com

INDUSTRY EDITOR

Norbert Mikula norbert@sys-con.com

PRODUCT REVIEW EDITOR

Joe Mitchko joe@sys-con.com

.NET EDITOR

Dave Rader davidr@fusiontech.com

TECHNICAL EDITORS

Andrew Astor aastor@webmethods.com

David Chappell chappell@sonicsoftware.com

Anne Thomas Manes anne@manes.net

EXECUTIVE EDITOR

Gail Schultz gail@sys-con.com

EDITOR

Nancy Valentine nancy@sys-con.com

ASSOCIATE EDITORS

Jamie Matusow jamie@sys-con.com

Jean Cassidy jean@sys-con.com

ASSISTANT EDITOR

Jennifer Van Winckel jennifer@sys-con.com

PRODUCTION

PRODUCTION CONSULTANT

Jim Morgan jim@sys-con.com

LEAD DESIGNER

Richard Silverberg richards@sys-con.com

ART DIRECTOR

Alex Bolero alex@sys-con.com

ASSOCIATE ART DIRECTOR

Louis Cuffari louis@sys-con.com

ASSISTANT ART DIRECTOR

Tami Beatty tami@sys-con.com

CONTRIBUTORS TO THIS ISSUE

Steve Bailey, Brian Barbash, Dave Chappell, Arulazi Dhesaseelan,
Abdul Kayam, Mark Little, Joe Mitchko, James Phillips,
Sean Rhody, Paul Roth, Jason Rouault, Dave Stanton,
Jim Webber, John Worrall, Bob Zurek

EDITORIAL OFFICES

SYS-CON MEDIA

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.

Periodicals postage pending

Montvale, NJ 07645 and additional mailing offices

POSTMASTER: Send address changes to:

WEB SERVICES JOURNAL, SYS-CON Publications, Inc.

135 Chestnut Ridge Road, Montvale, NJ 07645

©COPYRIGHT

Copyright © 2003 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 per-
mission. For promotional reprints, contact reprint coordinator. SYS-CON
Publications, Inc., reserves the right to revise, republish, and authorize
its 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 cov-
ered in Web Services Journal.

Service with a Smile

Written by
Sean Rhody



Author Bio:

Sean Rhody is the editor-in-chief of Web Services Journal and managing editor of WebLogic Developer's Journal. He is a respected industry expert and a consultant with a leading consulting services company.
SEAN@SYS-CON.COM



My dictionary defines service as the work pro-
vided by one that serves. Sometimes it
seems that you have to define something by
itself, even when you don't want to.

Web services provide the ability for an organization to expose its business process-
es for consumption, either public or private. And they do so in a vendor-, platform-,
and language-neutral format, as opposed to proprietary attempts at similar solutions
over the past two decades.

But it should come as no surprise to anyone that the way that Web services get
organized is, or should be, based upon an architecture that emphasizes the
provisioning of services as the core paradigm. Yes, Web services design is based upon
service-oriented architecture.

This concept has been around for decades. As soon as it became possible to split
work off from a main computer and perform part of the processing on a separate
machine, the concept of a service was born. Because the inevitable question that
comes from this ability is "Well, what do I put where, and how do I use it?". Service-
oriented architecture attempts to answer that fundamental question. No one seems
to ask the more basic question – "Why do I want to do that?", but most of the time the

answer appears self evident.

The reality is that certain hardware is more adept at certain tasks. A mainframe, for example, is
superlative at I/O, and is often used as the data source of record in multitiered systems. Mainframes are
expensive, however, and the past decade has seen a great deal of effort geared toward providing individ-
ual processing power to the desktop, in the form of the PC. Of course the problem was, and still is to a
certain extent, the fact that the mainframe widely outstrips the capacity of a PC for things like databases
and I/O, so the question has become one of "how do I work with the mainframe from my PC" rather than
how to replace the frame.

Throughout all of the various shifts in computing paradigms, client/server, the Internet, occasionally
connected computing, and so on, one facet has become clear – computing is about providing services.
Early on in computing, copy books and operating systems provided a form of services – common rou-
tines that people used over and over again to accomplish higher-level tasks. Later, the concepts of
distributed computing added the notion that not all services need to be provided by the same machine,
and that services could be layered, federated, aggregated into more coarse-grained operations.

For many developers, the concept of a service seemed slightly foreign to the way they were taught to
approach computing. One generation thinks in procedures, the next thinks in objects, but they both have
the common tendency not to think in differences. The old saying goes "when all you have is a hammer,
everything looks like a nail," and it holds true in programming as well, as developers tend to use what they
know and ignore the possibilities that they are less familiar with.

Until Web services, it was difficult to make use of such services. Without discounting CORBA or DCE,
or even COM, there were serious flaws either in the interface or the implementation of each approach
that made them difficult to use.

But no matter how the concept came about, use of services required a fundamental shift in program-
ming – from program to function. User interface became optional, not essential. The focus shifts to
creating business logic that can be used by some unknown, arbitrary system, rather than on code that will
only be called inside my Visual Basic or PowerBuilder program.

This issue focuses on service-oriented architecture and different aspects of the distributed program-
ming paradigm. SOA is an important facet of application development under the Web services paradigm.

Now, off to more work. Ironically, the word service derives from the Latin *servitum*, which the
dictionary defines as the condition of a slave. Such is life. ©

Mindreef

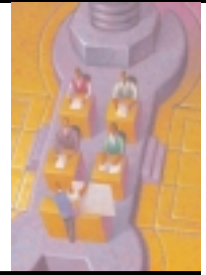
www.mindreef.com/nl0306

Mindreef

www.mindreef.com

Strategies for Achieving Real-Time Enterprise Application Integration

On the path to a service-oriented norm



Prior to the development of Web services, many enterprises were faced with very complex and expensive integration projects that were intended to tie a variety of enterprise business applications together with the goal of providing a seamless integrated business application platform inside the enterprise. Today, enterprises are rethinking their approach to integration strategies now that they are seeing numerous successful cases where companies are moving forward with the use of Web services.

AUTHOR BIO:



Bob Zurek is responsible for Ascential's overall product strategy, including new product development and technology acquisition. With more than 23 years of high-tech experience, Bob is instrumental in developing and driving Ascential's enterprise integration strategy, including the company's parallel processing framework, data quality and Web services strategies. Previously, Bob was a senior analyst with Forrester Research, where he was responsible for emerging technologies, including CRM, knowledge management, and wireless/speech.
BOB.ZUREK@ASCENTIALSOFTWARE.COM

Although Web services are well on their way to success, enterprises continue to leverage the combined capabilities of enterprise application integration suites, message brokers, and data integration platforms to achieve everything from connecting the disparate applications together to building data warehouses in order to apply analytics and reporting to better understand the data collected in the various applications, whether they were packaged applications or custom built. Successes point to having teams of professionals with deep experiences and expertise on the technical side of integration technologies. The failures are primarily due to lack of expertise and unrealistic goals associated with integration projects. More important, the failures were due to the time-consuming and difficult processes in developing scalable and reliable integration projects that connected enterprise applications together.

Today, the new enterprise architecture is a real-time, service-oriented architecture that leverages open and widely adopted standards such as XML, SOAP, Java, and JCA. Nothing says that services should always be "Web services" based on common components, including XML, SOAP, and UDDI. Services can also be exposed as Java, JavaBeans, JSP, and .NET services.

The development and deployment of a service-oriented architecture is seen by many,

including major research organizations like Gartner, as the next big thing. We are already experiencing the "early adopter" phase of the move toward this architecture, especially when it comes to integration. In fact, a recent industry survey indicated that integrating systems and processes is enterprises' most strategic IT priority. As standards mature and IT organizations gain experience with these new standards, we will move into the mainstream. This is not some overnight phenomenon; it will take time, as did the adoption of client/server computing. We will likely see this approach really accelerate over the course of the next year based on the number of early adopters seeing success with their efforts around Web services.

Standards

With Web services standards quickly maturing, enterprises are seeing early signs of a much simpler means for being able to orchestrate business processes across both operational and decision support systems. With interfaces in enterprise business applications exposed as Web services, the norm will be to orchestrate and integrate services exposed in both operational and decision support systems using enterprise business process managers. Enterprises will need a "service-oriented integration platform" that simplifies the process of integrating the services deployed within service-oriented architectures to achieve this goal.

The vision of many enterprise integration vendors is to be the standard service-oriented integration platform provider for the enterprise. In fact, it is a large opportunity for many of these vendors to extend their leadership in the integration market. A recent ZapThink survey indicates that the market opportunity will be about \$6.2 billion by 2006. Already most of the major integration vendors have made strong progress in adapting standards like XML, SOAP, Java, and JCA to their platforms. This approach enables them to extend their platforms to achieve a service-oriented integration platform. Enterprises pursuing service-oriented architectures must now steer clear of any integration offering that lacks support for Web services as it is considered required functionality in an integration vendor's offering.

Beyond helping with integration, how can you justify building a service-oriented architecture? One of the key motivations for leveraging a Web services-based, service-oriented IT architecture is to help the enterprise reduce latency in their business operations in order to become what many call a real-time enterprise. Pursuing this strategy will enable the enterprise to build better relationships with customers and business partners, eliminate process bottlenecks, streamline business processes, and leverage real-time critical data to better predict and optimize the key metrics in their business.

Parasoft

www.parasoft.com/ws2

An Example for the Future

What business would not like to achieve these goals? One of my favorite examples of a company that will benefit from a service-oriented approach is Kinko's. In a recent announcement, Kinko's said that in the future it will expose a Kinko's printing service as a Web service to Microsoft Word users. With the Kinko's service enabled, it is envisioned that Word users, choosing the print command, would get the standard print dialog, but in addition to seeing all the printers the user would also see a "Print at Kinko's" option. Selecting the service would then allow users to pick the Kinko's closest to their physical location to have the job printed. I'm sure that a message would travel back to the user indicating that the print job was complete and ready to be picked up. Who knows, may be even a map from MapQuest will be displayed showing the exact Kinko's location. Imagine how, by taking this approach, Kinko's could expand their business model by making it convenient for their customers to use this service.

Expanding the concept further, imagine Kinko's interacting with Federal Express when the user simply sends a list of mail addresses to the Kinko's service to request that the final print jobs be sent to Federal Express for delivery to the specific addresses. This is quite visionary on behalf of Kinko's. With success and good user adoption, this service could serve as a "poster child" to demonstrate how, by taking a service-oriented approach to their business, Kinko's can better service their customers and further expand their business visibility.

Spurred by such visions, many enterprises are now moving their business IT infrastructure to a service-oriented architecture to help achieve these goals. That's the good news. The bad news is that it will take time and effort and an investment to retrofit existing systems with services-oriented interfaces. One obstacle preventing some from moving forward is a concern for where standards are going. Other enterprises have overcome these concerns as they actively monitor standards efforts, even to the point of participating in the standards bodies.

Both business and industry are encouraged by the progress of these standards as they become more mature. This is showcased by the efforts of the Web Services Interoperability Organization (WS-I), which describes itself as an open industry effort chartered to promote Web services interoperability across platforms, applications, and programming languages. It

brings together a diverse community of Web services leaders to respond to customer needs by providing guidance, recommended practices, and supporting resources for developing interoperable Web services. Most of the major enterprise application vendors now actively participate in this effort. These efforts will go far in helping enterprises achieve a service-oriented architecture leveraging Web services as the primary vehicle for integration of enterprise business applications. With this in mind, another strategy on the path to a service-oriented architecture is to become actively engaged in organizations like WS-I. At this writing, the WS-I is underrepresented by the Global 3000 enterprises. Strategically, enterprises should consider joining organizations like the WS-I. This will go far in helping architects inside the enterprise to gain considerable insight and develop a great network of very smart technologists and strategists who will be invaluable when it comes to pursuing a service-oriented architecture using Web services.

Industry Leaders

Actively engaged in standards efforts and moving forward with service-oriented architectures are the industry-leading packaged business application vendors, including Siebel, SAP, Microsoft, and PeopleSoft. These companies are moving in this direction to help enterprises achieve the benefits of the service-oriented architecture. In addition, many vendors are promoting strategies and solutions to help enterprises integrate their offerings with other packaged and custom applications through the use of Web services as a primary enabling technology in their offerings.

Siebel Systems

One of the most highly visible efforts by an enterprise application vendor today is Siebel Systems. Siebel is investing over \$100 million in a project called Siebel Universal Application Network (UAN). By taking a very services-oriented approach to UAN, Siebel can provide a model for how enterprises can rapidly connect both Siebel and non-Siebel business applications. Siebel UAN takes a process-centric approach to integrating applications. In fact, Siebel provides a library of prepackaged cross-industry and industry-specific business processes, such as quote to cash, campaign to lead, and others, that form the heart of Siebel UAN.

Siebel is unique in their approach as they are providing a framework (process, common

objects, and transformations) that can be used by any integration vendor that supports UAN. Other application vendors will either work independently of the integration vendors or build their own integration stack. SAP is one vendor that has built their service-oriented integration stack but also works with integration vendors.

Siebel does not provide a native integration server as part of UAN. Instead, they defined all of the business processes using the BPEL4WS (Business Process Execution Language for Web Services) standard. With the processes defined in BPEL4WS, Siebel is promoting an "integration server-neutral model" for its business processes and has teamed up with integration vendors including SeeBeyond, TIBCO, Vitria, IBM, and Microsoft to help ensure that enterprises that have adopted these integration brokers can leverage the capabilities of UAN.

Although UAN has a strong vision, Siebel is just getting started. Enterprises that use Siebel will want to investigate the capabilities of UAN and determine how it might fit into their service-oriented architecture. Integration brokers supporting UAN are also in the early phases of adoption. For example, many of the integration brokers don't have native support for BPEL4WS and will probably resort to importers and exporters to bring the defined processes into their integration broker business process environment. (Part of the reason for this approach is that some believe that BPEL4WS may go through further changes or be incorporated as part of a superset standard. It is hard to predict where the BPEL4WS standard will go.) Because Siebel is in its early stages with UAN, they offer only a limited number of predefined processes. With this limited number, many enterprises might want to take a wait-and-see approach until UAN matures further.

One other obstacle Siebel is likely to encounter is the wide range of custom scenarios required by the enterprises adopting Siebel. We live in a world of heterogeneous systems and customized packaged applications. This will require enterprises to adopt UAN to take a path of customizing UAN. Although Siebel has done a good job in defining common business objects like customer, employee, product, and order, not all enterprises will be able to directly align with these objects and in fact will likely have to customize these base objects to more directly align to their business requirements. The good news is that Siebel does provide an object base to get started. They also support customization of these objects, which gives us

Quest Software

www.java.quest.com/jcsr/ws

a sense that Siebel understands the issues of customization required in an enterprise.

Due to the highly competitive nature of the packaged enterprise application space, Siebel will face challenges when selling UAN in a world of heterogeneous applications, including SAP and PeopleSoft. Both companies have competitive solutions and a battle has already begun for enterprise mind share. Certainly SAP and PeopleSoft will be pushing their UAN-like approach over Siebel.

SAP

Competing directly with Siebel UAN is SAP's NetWeaver platform. According to SAP, "SAP NetWeaver is the foundation of SAP xApps and mySAP Business Suite solutions. It's also the enabler of SAP Enterprise Services Architecture, which combines the enterprise applications experience of SAP with the flexibility of Web services and open technologies – for complete and services-based business solutions." When you peer into NetWeaver you unveil a very comprehensive service-oriented architecture that leverages the capabilities of open standards and Web services. NetWeaver claims to "provide all the capabilities that an integration and applications platform requires to develop, integrate, and run solutions following SAP Enterprise Services Architecture."

Part of the SAP NetWeaver offering is the SAP Exchange Infrastructure, which includes an integration broker that enables XML/SOAP interactions between services exposed by various vendors' source systems. The SAP Exchange Infrastructure also supports a business process manager, which enables an enterprise to model and execute business processes within a service-oriented architecture.

Although deep in its functionality, SAP also

understands that organizations will have existing integration solutions in place. With this in mind, SAP exposes services in NetWeaver that are accessible for other integration broker solutions. Enterprises, however, may adopt NetWeaver components alongside these integration brokers with the assumption that they will avoid having to work with multiple vendors. It isn't unusual for an enterprise to support multiple integration brokers. In fact, Gartner predicts that more than 67% of large enterprises will have two or more different integration brokers in production by year-end 2003, and half will have four or more by year-end 2005 (0.7 probability).

PeopleSoft

Right up there with UAN and NetWeaver is PeopleSoft with their AppConnect and PeopleSoft Internet Architecture. The PeopleSoft AppConnect product suites include an enterprise-class portal, a standards-based integration broker, and an enterprise warehouse. The primary communication mechanism for PeopleSoft AppConnect is Web services. Like SAP, PeopleSoft works closely with integration broker vendors such as Vitria and SeeBeyond with the understanding that enterprises will want to ensure that these integration brokers are "certified" to interoperate with PeopleSoft's offerings. With this in mind, PeopleSoft as well as Siebel and SAP have certification processes that validate that the integration broker is a "good citizen" in interoperating with the particular vendors' offerings. PeopleSoft delivers Web services-based integration by exposing Web services so that PeopleSoft applications can invoke services in other systems. In addition, systems can easily invoke PeopleSoft Web services from other systems to access content and

data in PeopleSoft applications. PeopleSoft also supports messaging in their offering, where third-party systems can initiate services in PeopleSoft and subscribe to messages by using baseline Internet technologies inherent in the PeopleSoft Internet architecture.

Integration vendors will also vie for the position of controlling the enterprise standard for service-oriented integration inside an enterprise. Although most enterprises' application vendors have integration capabilities built in, many take an integration server-agnostic approach to their integration strategy and have attracted the major integration vendors into their appropriate camps. Most integration vendors see the value of "cooperating" with these vendors as the vendors give them tremendous visibility into a very large base of enterprise customers. The problem with this approach is that integration server vendors now claim that they are the standard for these solutions. I've seen this in action in public symposiums where several integration vendors have claimed that the enterprise application vendors themselves use their solution to build and test processes, giving their offering a strategic advantage over others. Many enterprise application vendors respond that they work with all the integration vendors' products to ensure the success of their solutions. With this in mind, enterprises must take this off the decision-making table as it brings little differentiation. What enterprises must look to is a proven track record of successful implementation using technologies like Siebel UAN, SAP NetWeaver, and PeopleSoft AppConnect that represents real-world scenarios.

Conclusion

With standards maturing, integration brokers supporting service-oriented architectures, and mainstream enterprise application vendors supporting Web services, all of these efforts are leading enterprises down a path where a service-oriented integration platform will become the norm over the course of the next few years. This services-oriented integration platform will help enterprises deliver the right data at the right time and will be an important key to achieving a real-time enterprise. As these efforts take shape and enterprises achieve success, we'll begin to see a fundamental shift in how enterprises integrate their applications. Business process will then be the key driver to help enterprises achieve real-time, cross-business application integration with Web services. ©

Key Strategies

1. Investigate the benefits of taking a services-oriented approach to your next-generation architectures.
2. Develop key value propositions and examples tailored to your business to showcase how the business will benefit from taking a service-oriented approach to your future architectures. Some of these value propositions will be streamlining latency in the business, reducing time and cost to integrate business applications, expanding the business model, and developing better customer relationships.
3. Investigate the capabilities of your existing vendor's service-oriented offerings and ensure that they have existing or near-future capabilities.
4. Identify early adopter candidates inside the enterprise that will quickly benefit by adopting a services oriented architecture.
5. Establish processes for developing and deploying a service-oriented architecture.
6. Develop cost/benefit analysis and if the results are positive move forward.
7. Develop and deploy the service-oriented architecture in cooperation with your strategic vendors.
8. Analyze and publish findings in order to gain support for moving forward with other candidates in the organization.

Crystal Decisions

www.crystaldecisions.com/cr9/220

Reconstructing J2EE - Java Business Integration Meets the Enterprise Service Bus

Changing the economics of integration



Web services have given newfound importance to service-oriented architectures and promise to drive down the cost of integration by providing a standards-based approach to interoperability between applications. The trouble is, what people really want is a new way of doing integration. Until now, we haven't really had a way to incorporate Web services into a meaningful architecture for integrating applications and services into a fabric that spans the extended enterprise in a large-scale fashion. With the advent of the enterprise service bus we have that architecture.

The Java Business Integration (JBI) specification, JSR 208, has set out to define a loosely coupled integration model that aligns with Web services-style distributed computing. The JBI expert group hopes to utilize existing J2EE integration components such as the Java Message Service (JMS), J2EE Connector Architecture (JCA), and the J2EE 1.4 Web services APIs, and add a host of new capabilities as well. The charter of the expert group promises to define an open-ended, pluggable architecture that will support both industry-standard integration components as well as proprietary

elements through the use of standardized interfaces.

Coincidentally, the ESB addresses similar goals. It is based on today's established standards, and has real implementations that have been shipping for at least a year. JBI can learn a great deal from what the ESB approach offers.

The Enterprise Service Bus, an Industry Phenomenon

The enterprise service bus (ESB) is a rapidly emerging technology category that is gaining industry attention from analyst firms and the vendor community alike. More than just a new architecture for integration, the ESB promises to change the economics of integration projects by bringing the fabric of integration into the hands of the everyday IT professional. Gartner Group predicts that an ESB infrastructure will be running in the majority of enterprises by 2005 (DF-18-7304). IDC likens the advent of the ESB to the invention of the steam engine or the telegraph. In IDC's

report (#29132), Sally Hudson went on to say, "The ESB is an open standards-based technology concept that will revolutionize IT and enable flexible and scalable distributed computing for generations to come. This flexible, reliable infrastructure can enable a new type of corporation – one that is not held captive to rigid IT capabilities, but rather can respond almost instantaneously to meet business opportunities."

Infrastructure vendors are "getting on the bus" at a regular pace now, with early pioneers including Sonic Software, and early adopters including SeeBeyond, IONA, SpiritSoft, Kenamea, and Cape Clear. The ESB has received front-page attention several times from industry weeklies such as *InfoWorld* and *IT-Week*. ESB has also been written about in *Web Services Journal* and *WebServices.org*, and debated on the *theServerSide.com*. An employee of a very large ERP vendor stood up in one of my presentations recently and announced to the audience that they are currently in the process of "ESB-enabling" their ERP suite.

The ESB is an architectural concept that is already being deployed in real production environments. Customers like ABNA, a subsidiary of \$1B Associated British Foods, have adopted the ESB to provide standards-based integration between its internal systems and electronic trading partners, utilizing the ESB infrastructure for XML document exchange between its supply manage-

AUTHOR BIO:



Dave Chappell, technical editor of *Web Services Journal*, is vice president and chief technology evangelist for Sonic Software with over 20 years of industry experience building software tools and infrastructure for application developers.

He is a regular speaker at the Web Services Edge Conferences & Expos. Dave represents Sonic as a member of the JSR 208 Expert Group.

CHAPPELL@SONICSOFTWARE.COM



ment systems and ABF subsidiaries, as well as other trading partners.

Changing the Economics Through Standards-Based Integration

Integration brokers have long suffered from being highly proprietary in nature. This proprietary nature has been very problematic to organizations trying to integrate on a large scale because it leads to unwanted vendor lock-in, and usually carries with it a high cost in consulting fees throughout the project. In the end, what is usually accomplished is an expensive island of integration that can essentially only be viewed as an ongoing "integration tax" to be written off as a cost of doing business.

In contrast, the introduction of standards throughout an integration fabric – which in part defines the ESB approach – offers business benefits through the following characteristics:

- **Increased ability to integrate with** business partners using standard interfaces and standard protocols.
- **Leverage existing IT staff:** The amount of information available on Java standards such as JAXP, JCA, and JMS, and XML and Web services standards such as SOAP, WSDL, XSLT, XPath, and XQuery is expanding at an ever-increasing rate. This means that the average IT professional can readily attain the expertise he or she

needs to become the in-house integration architect.

- **Reduce proprietary vendor lock-in:** With a proprietary integration stack, an organization can't simply say, "Well, vendor A wasn't what we expected, so let's try vendor B."
- **Projects become more predictable:** With a standards-based integration approach, common design patterns and success factors can be carried from project to project without expensive consultants and vendor lock-in. As the manager of a large IT department put it: "we no longer have to worry about how long the next integration project will take, or even if it's possible."

Characteristics of the ESB Employ Key Standards for Integration

There are a significant number of standards, all of which have reached a fair enough maturity level in implementation that fit together very nicely in an ESB architecture. It could even be posited that the evolution of the standards have helped to catalyze the creation of the ESB. Table 1 shows a sampling of the main characteristics that make up the ESB, and the standards that support those traits.

(Note: In Table 1 the items highlighted with a *, such as WS-Reliability and WS-Security, are not yet part of the ESB since they have not yet reached a sufficient level of maturity and widespread adoption. However they are considered strong future candidates." The ESB is all about standards-based integration based on what is available today, in an architecture that is well suited to support the maturing technologies of tomorrow.)

Getting on the Bus

A common interpretation of the term "bus" is the notion of an architecture that one simply plugs into. Think of a hardware bus in a PC motherboard, with multiple slots that any type of compatible card can be plugged into. The use of the term "bus" in the ESB case is analogous to that in the way that one interfaces with the bus. From the perspective of the owner of a service or an application domain, you need only be concerned with three operations – you plug into the bus, you

post data to the bus, and you receive data from the bus. The Enterprise Service Bus takes care of getting the data to the applications that need it, in the target data formats that they need to see it in.

Expanding the Reach of Integration Services

Roy Schulte of Gartner Group summarizes the ESB as "combining Message Oriented Middleware (MOM), Web Services, and Transformation and Routing Intelligence." These are concepts that are found in many traditional integration brokers. The ESB concept incorporates those same concepts, yet applies standards to every aspect possible, and does it in a way that enables a highly distributed, yet centrally managed SOA. Think of the contrast between conventional integration broker stacks versus the ESB approach as centralized and proprietary versus highly distributed and standards based.

An ESB is based on a distributed, federated network model. It takes on the responsibility of reliable communication within the network with a common security model, and provides the integration capabilities that allow applications to easily interoperate. Integration capabilities such as data transformation are themselves implemented as services and can be independently deployed anywhere within the network. The result is precise deployment of integration capabilities at specific locations, which can then be scaled independently as required. It also means that services can easily be upgraded, moved, or replaced without affecting applications.

TABLE 1: Characteristics & Standards of the ESB

Web services	XML, SOAP, WSDL
Content-based routing and transformation	XSLT, XPath, XQuery, JAXP
Enterprise Messaging	JMS
Connectivity	JMS, JCA Adapters, HTTP/S
Reliable SOAP	WS-Reliability, WS-ReliableMessaging
Highly distributed Services Container	- Apache Soap, Apache Axis SOAP envelope APIs, JAXP - JSR 208?
Enterprise Manageability	JMX, JNDI, LDAP
Security	- SSL, HTTP/S - PKI - JSSE, JAAS - LDAP

Intelligent routing based on content is accomplished through specialized services that apply XPATH expressions to identify a document type and route it based on values within the document. Routing of messages is also accomplished using a *message itinerary*. Think of a message itinerary as a list of destinations that travels with the message as it moves from service to service, or application to application. The ESB evaluates the itinerary at each step along the way, without the requirement of a centralized rules engine. More complex orchestrations, including stateful, long-duration conversations, are also possible using orchestration facilities. An ESB architecture is well suited for supporting choreography languages such as BPSS or BPEL4WS.

In your role as IT's new integration architect, your job is to tie applications and services together in a service-oriented architecture using itinerary-based routing, content-based routing, and orchestration. The interface through which this is done is the ESB services container.

Highly Distributed Services Container

Perhaps the most important characteristic of the ESB is the highly distributed services container, which allows the selective deployment of discreet services exactly when and where you need them. The services container is the interface between the bus and the applications and services that communicate through the bus (see Figure 1).

The services container can house an application adapter via JCA, or it may expose an MDB or JMS interface to talk to a J2EE app server. It may house an XSLT transformation engine that converts a particular XML dialect into an agreed-upon centralized common format, or act as a content-based routing service. A service container may be an XML caching, staging, and aggregation service that gathers data from multiple sources prior to rendering them together using an XSLT transformation, in preparation for feeding into an ERP system. A service container may also act as an interface to external Web service invocations, both inbound and outbound. Combined with the

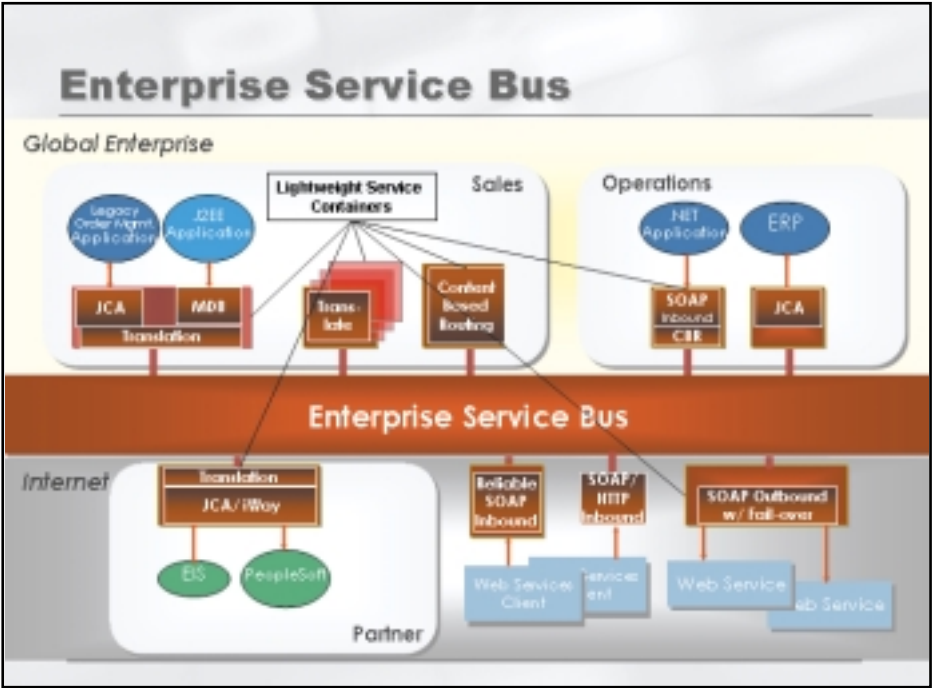


FIGURE 1 Lightweight distributed-service containers allow highly distributed, selective deployment of services across the enterprise.

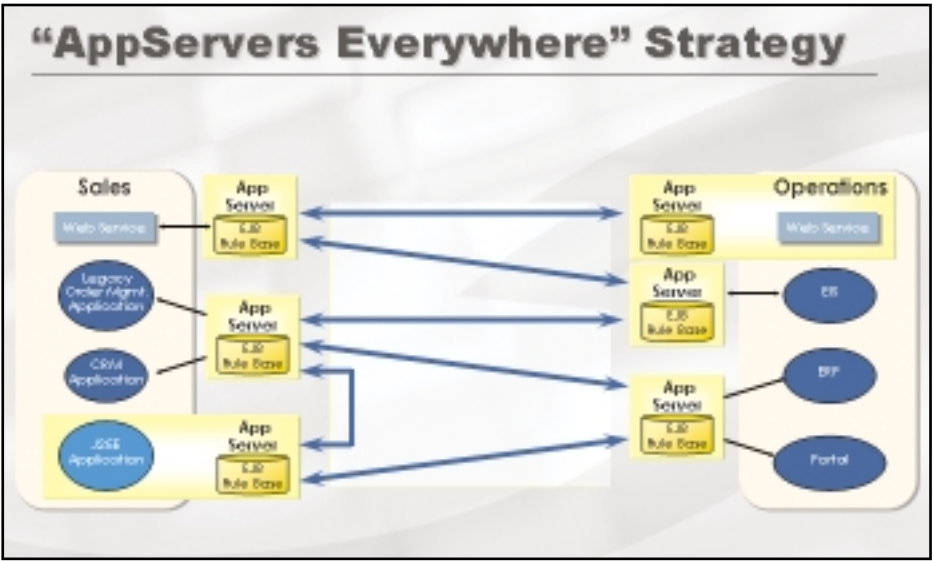


FIGURE 2 Appservers Everywhere means significant overkill in many cases.

staging service, it could be used as a portal cache for a portal server based on J2EE.

XML transformations can be very CPU intensive. As indicated by Figure 1, services containers can be independently scaled across as many machines as necessary to support the load required.

HTTP and SOAP are no problem for the ESB, as it is capable of exposing SOAP-based Web services to the outside world, and also capable of calling out to external Web services across the Internet. The ESB takes care

of mapping SOAP-over-HTTP to SOAP-over-JMS for both inbound and outbound Web services invocations. Emerging WS-Reliab* SOAP specifications, such as WS-Reliability and WS-ReliableMessaging, can be easily adapted into the ESB architecture as they become widely adopted.

Reconstructing the J2EE Stack

The current approach taken by integration brokers that are built on app server platforms is to install the entire app server stack in every

place that a particular piece of functionality is needed. This is referred to as the “AppServers Everywhere” strategy (see Figure 2).

The “Appservers Everywhere” strategy can be viewed as overkill in some cases. Think of installing an entire integration broker or app server stack in a particular department, or remote location, simply to act as a conduit to a JCA container which houses an adapter to an ERP system. Why do that when you can simply place a services container there, and have it act as the JCA container directly?

Don’t get me wrong. App servers are good, and have their place in the ESB, for things like managing Web-facing portals and housing EJBs. Through JMS and MDB, all app servers, such as WebLogic, WebSphere, and JBoss, can hook directly into the bus and become a part of its service-oriented architecture. However, the integration functionality is delegated to the ESB itself.

The requirements of integration based on standards are vastly different from the application server view of the world. Today’s integration projects require a substrate that is built from the ground up to enable loosely coupled, document-centric XML interchange. This does not mean that the entire J2EE stack needs to be discarded. The ESB is built upon Java standards such as JMS, J2EE Connector Architecture, EJB, MDB, WSEI, JAX-RPC, SAAJ, Java servlets, JMX, JAXP, JAAS, and JSSE. It simply means that a radical rearranging, or “Re-constructing,” is in order.

“
the ESB promises to
...change the economics
of integration projects
by bringing the fabric
of integration into the
hands of the everyday
IT professional”

The services containers are connected together through a messaging backbone using

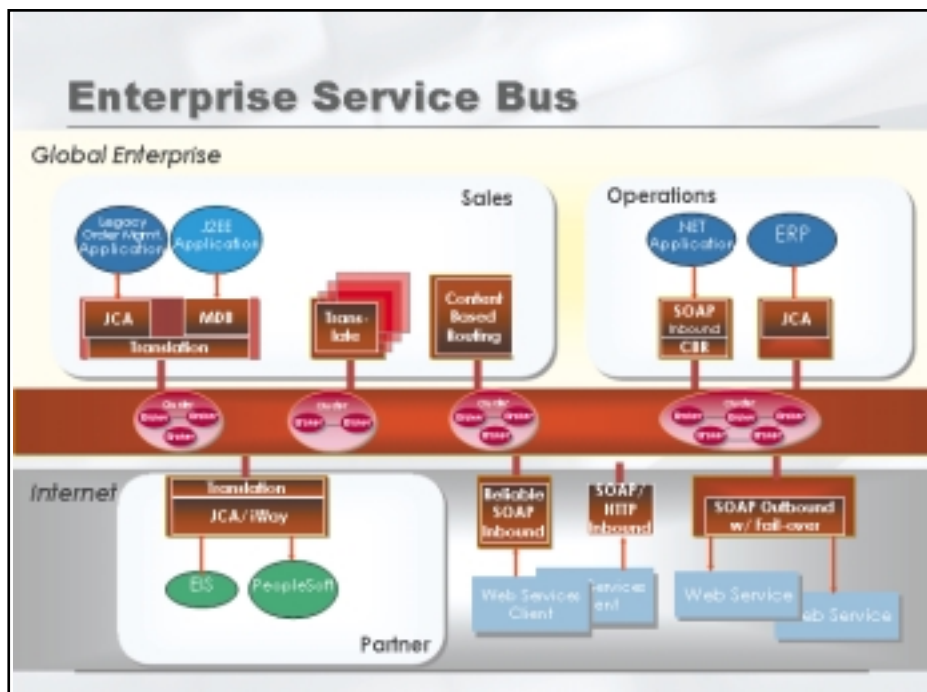


FIGURE 3 Scalable clusters of JMS servers at the heart of the ESB infrastructure.

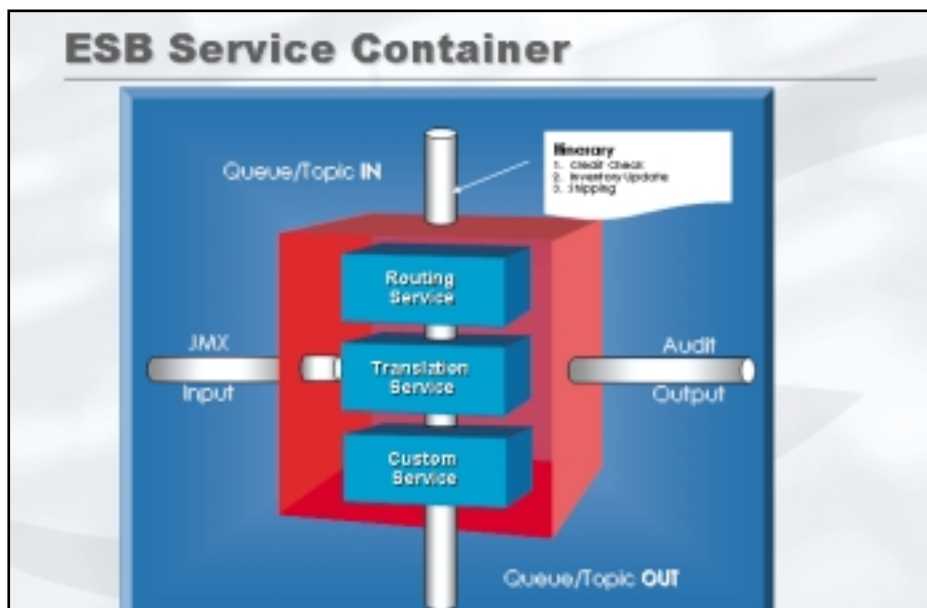


FIGURE 4 Service Container is a JMX-managed MBean

JMS. As illustrated in Figure 3, the JMS backbone can support scalable clustering across a large scale global deployment.

The services container is both a JMS client and a JMX MBean container that is capable of housing integration services (see Figure 4). The container receives JMX management input, and has special output endpoints for error handling and auditing.

The services container uses a Java servlet-like inbox/outbox processing metaphor. The

destinations that the endpoints eventually map to are administratively defined. Underneath the covers, the in/out endpoints are mapped to JMS topics and queues that carry SOAP envelopes. The topic or queue can be directed at an MDB or a JCA container, or could be mapped administratively to call out to an external Web service using a mapping to SOAP over HTTP, or even WS-Reliable*.

The service API can be implemented using an open-source SOAP stack such as

Apache Soap or Apache Axis, and could conform to Java standards such as JMS, JAX-RPC, or SAAJ. It could be a combination of those. That is probably something that should be standardized by the JBI/JSR208 EG. There is already some API work being done by Sonic and IBM in Apache Axis to build an asynchronous callback API for SOAP services based on combining the JAX-RPC with JMS. The goal was to not reinvent the wheel and to take advantage of existing standards. Perhaps we can draw from this work and use this approach in JBI/JSR208.

Technology Catalysts Driving Business Models

Mature standards such as the Java Message Service (JMS) for messaging, and the J2EE Connector Architecture (JCA) mean that IT can now pick and choose from best-of-breed implementations and combine them together. Proprietary application adapters are also becoming standardized. Companies such as iWay are selling ESB-enabled adapters, in an a la carte model, that are based on standard interfaces such as JMS and JCA. This is creating a highly competitive market that competes on price, performance, scalability, and functionality. De facto standard SOAP stacks such as Apache Soap and Apache Axis are licensed freely, yet have strong backing from industry vendors who help build these SOAP stacks for use within their own commercial offerings, and offer their contributions back into the open source community.

According to Gartner Group, an essential characteristic of the ESB approach is "high function, low cost." Low cost is certainly reflected in the licensing, but more importantly in the overall cost of configuration, deployment, and cost of ownership over time. Contemporary integration broker vendors have built their businesses by charging initial license fees ranging from \$500K-\$1M, and consulting fees typically five times the license cost. In contrast, the licensing fees for an ESB approach can range from one tenth to one half of that cost, with consulting fees an average one tenth of the cost of conventional integration consulting fees. The competitive

landscape being created by up-and-coming ESB companies, combined with the business benefits of standards-based integration stated earlier, will take a heavy hit on the business model of the traditional integration broker and the app server vendor alike. Even if the existing vendors can somehow re-architect their products to create a low-cost, highly distributed ESB offering, it's questionable whether their business model can be adapted to survive the new economics of integration.

“...today's integration requirements call for a vastly different approach from the app server model that was once built for processing Web requests”

As ESBs become more widespread, system integrators may adapt to the lower cost/consulting ratio by adopting the ESB approach and taking advantage of its proliferation.

And one last point: although this article focuses on J2EE, the ESB is platform agnostic. A good ESB should be capable of supporting direct access through .NET clients and C++ clients.

I'm very excited about the formation of a JSR in the area of business integration, as is Sonic, because we feel that today's integration requirements call for a vastly different approach from the app server model that was once built for processing Web requests. The ESB approach, along with JSR 208, will "change the economics of integration" and breath some fresh ideas into the J2EE stack. ©

WebServices JOURNAL

PRESIDENT AND CEO

Fuat A. Kircaali fuat@sys-con.com

VP, BUSINESS DEVELOPMENT

Grisha Davida grisha@sys-con.com

ADVERTISING

SENIOR VP, SALES & MARKETING

Carmen Gonzalez carmen@sys-con.com

VP, SALES & MARKETING

Miles Silverman miles@sys-con.com

ADVERTISING DIRECTOR

Robyn Forma robyn@sys-con.com

DIRECTOR, SALES & MARKETING

Megan Ring-Mussa megan@sys-con.com

ADVERTISING SALES MANAGER

Alisa Catalano alisa@sys-con.com

ASSOCIATE SALES MANAGERS

Carrie Gebert carrie@sys-con.com

Kristin Kuhnle kristin@sys-con.com

SYS-CON EVENTS

PRESIDENT, SYS-CON EVENTS

Grisha Davida grisha@sys-con.com

CONFERENCE MANAGER

Michael Lynch mike@sys-con.com

CUSTOMER RELATIONS/JDJ STORE

CIRCULATION SERVICE COORDINATORS

Niki Panagopoulos niki@sys-con.com

Shelia Dickerson shelia@sys-con.com

JDJ STORE MANAGER

Rachel McGouran rachel@sys-con.com

SYS-CON.COM

VP, INFORMATION SYSTEMS

Robert Diamond robert@sys-con.com

WEB DESIGNERS

Stephen Klimurray stephen@sys-con.com

Christopher Croce chris@sys-con.com

ONLINE EDITOR

Lin Goetz lin@sys-con.com

ACCOUNTING

FINANCIAL ANALYST

Joan LaRose joan@sys-con.com

ACCOUNTS RECEIVABLE

Kerri Van Achen kerri@sys-con.com

ACCOUNTS PAYABLE

Betty White betty@sys-con.com

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

1-888-303-5282

FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,

PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT

COVER PRICE: \$6.99/ISSUE

DOMESTIC: \$69.99/YR (12 ISSUES)

CANADA/MEXICO: \$89.99/YR

ALL OTHER COUNTRIES: \$99.99/YR

(U.S. BANKS OR MONEY ORDERS)



macromedia

www.macromedia.com/go/cfmxad

The Differentiation of Web Services Security

How can you leverage your investment in enterprise security?



Security is cited as the number one concern in building and deploying Web services today. Web services are inherently a different architecture that presents a whole new set of challenges.

You will have to reexamine many of the security aspects of your infrastructure, such as confidentiality, integrity, authentication, non-repudiation, and cohesion. The bottom line is, you can build robust, secure systems today by leveraging your existing investment in security and taking advantage of new and evolving security standards. The business benefits of Web services are too great to hold up their adoption because of security concerns. Understanding the architectural issues will allow you to take advantage of this in your enterprise.

The Traditional Approach

Traditional applications are connection oriented, allowing many security details to be implemented at the connection level, and requiring a direct connection between the service provider and consumer. However, Web services are message oriented and lack the guarantee of a direct connection between service provider and consumer, so many traditional connection-oriented approaches to common security challenges are inappropriate or insufficient for a Web services security architecture. In addition, the introduction of Web services into an organization may be the

first time that organization has ever allowed external users access to their business applications; the security practices and standards which are appropriate for inside use may be completely unacceptable when outside users are introduced. Creating an effective Web services security architecture requires a focus on achieving business requirements for your security architecture in the face of these two primary differences between Web services and traditional applications.

Confidentiality, ensuring that communications between a sender and receiver are private, is traditionally solved using connection-oriented security protocols such as HTTPS (or HTTP over SSL). However, Web services communications are message oriented, and may traverse many different "network hops" before they reach their final destination. Thus, the use of a connection-oriented protocol cannot guarantee confidentiality between a message's first producer and its final consumer.

Integrity, ensuring that communications are not altered between a sender and a receiver, is also traditionally solved at the connection level using error correction systems (such as CRCs and checksums) and connection-oriented security protocols (such as SSL). Connection-oriented security protocols that guarantee communications integrity have the same problem in ensuring integrity across all processors of a message as they do in ensuring confidentiality across those "network hops." They also suffer from an entirely different problem when used to guarantee communi-

cation integrity: message-oriented integrity should be confirmable for an extended period in the future. In other words, if a message is stored over a period of time, it should be verifiable at any point in the future that it still has not been modified. Connection-oriented approaches cannot guarantee this.

Authentication in a traditional application is usually solved using some type of user name/password combination while establishing a connection. The application can then assume that for the remainder of the connection, the authenticated user is still on the other end of that connection. Since Web services are message-based, and there is no single connection that connects a message sender with a message receiver, entirely new approaches to authentication must be employed to ensure that every message contains authentication information, and that this information can be confirmed for every message.

Nonrepudiation can be considered to be the combination of integrity (that a message was not altered in transit) and authentication (that a particular entity sent a message), and refers to the principle that a sender cannot deny that he or she sent a message, and a receiver cannot deny that he or she received it.

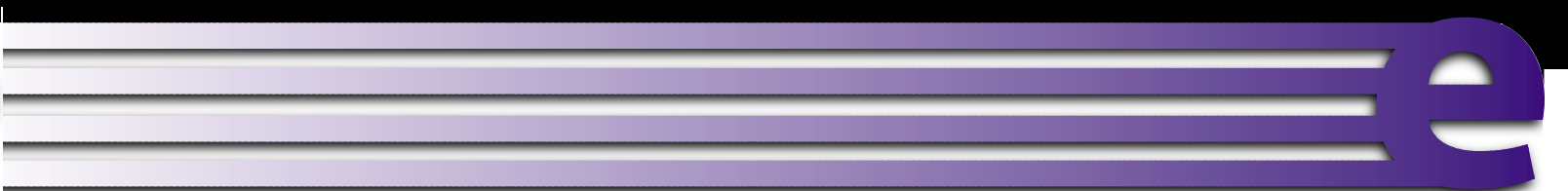
The Web Services Approach

Perhaps the most important requirement for Web services security architecture is one of cohesion. Developing a Web services architecture requires quite a bit of coordination between different systems, particularly when it comes to security. A typical organization may have legacy applications that are to be Web service enabled; external Web services; and enterprise applications that provide Web services interfaces to combine into a Web services architecture. Each of these might have a different set of security information, or even worse, an entirely different security architecture.

With the proliferation of security users, roles, domains, and realms that the Web services architecture

Author Bio

Dave Stanton is a serial entrepreneur. Prior to founding Talking Blocks, he was CTO of Xpedior and founded Sage IT Partners (sold to PSInet). Dave has held management positions at Metropolis Software and Andersen Consulting, and is an MBA and graduate of the Air Force Academy.
DAVE.STANTON@TALKINGBLOCKS.COM



will entail, it will be difficult to make enterprise-wide decisions and verify that enterprise-wide security standards are being adhered to. Therefore, a critical aspect of any Web services architecture must be a security management framework, allowing a centralized way to organize and coordinate the different security systems that will be present in the service-oriented architecture. Such a framework should enable setting and enforcing security policies across all Web services present in the organization.

The Use of Standards

Although the standards used in Web services are still evolving, some have already been proposed and adopted by the developer community (such as SOAP). Since the message-oriented nature of Web services communication and distributed nature of Web services networking pose entirely new security challenges, new standards had to be developed to ensure that Web services architectures can meet enterprise security requirements.

Because Web services share many technical similarities with Web applications, some techniques for securing Web applications can be directly utilized in Web services architectures. For example, the use of HTTP to transmit Web services messages is common, as is the use of HTTP over SSL (or HTTPS) to ensure connection-level security and the HTTP/Basic challenge-response authentication scheme for username/password connection-level authentication. Furthermore, the use of HTTP headers and cookies has also been applied to the challenge of communicating security and session information from the sender to the receiver.

However, because Web services usually operate over XML rather than HTML, an entirely new set of standards has been under development to provide XML-specific security features. The XML-Encryption standard defines how to include encrypted data (which might be other XML elements) in an XML document, and the XML-Digital Signature standard defines how to encode a digital signature (which might be signing other XML elements) in an XML document. While these have direct applications to Web services security, they are specifications for the generic problem of encoding complex data structures (such as a digital signature) in XML.

An important part of a security system is its ability to propagate security context between processing nodes. This might happen between an end-user's machine and the first Web server it communicates with (in which case cookies might be used), or it might occur between two servers in a portal environment. A new standard, the Security Assertion Markup Language (SAML), being managed by OASIS, exists to allow the propagation and validation of security assertions in Web services architecture. The use of SAML as a security assertion specification provided the basis for the Liberty Alliance specifications. These specifications describe a system that allows federated user definition across a range of organizations with single sign-on from a particular identity provider.

With so many different standards defining different components of the Web services security puzzle, an effort was required to determine how to apply them to Web services. That effort, initially led by Microsoft, IBM, and VeriSign, resulted in the WS-Security specification, now being coordinated by OASIS. WS-Security attempts to provide a comprehensive approach to message integrity, authentication, and confidentiality by specifying how to include features such as digital signatures and encryption in SOAP documents, and is expected to encompass other standards, such as SAML, for security assertions.

The Security Architecture

A Web services security architecture should decouple the semantic meaning of your services from their provision, allowing you to manage a complex security infrastructure without changing your applications. This means that your services need to be concerned only with context-sensitive authorization decisions, rather than having to understand the vast scope of Web services security standards.

By decoupling service provision from service control, you control access to your Web services using different mechanisms for different consumers. This could be achieved by using a message router or other security intermediary. For example, most enterprises will have different security requirements for services being consumed within the company than for those being consumed by business

partners. This allows you to deploy the service once, and manage your security requirements based on the characteristics of the consumers of your service, rather than the characteristics of the service you're deploying.

Furthermore, this approach allows you to "future-proof" your applications. Standards surrounding Web services security are constantly evolving. By delegating security, you allow your enterprise the ability to constantly keep up with new standards, providing better levels of functionality and interoperability without changing your services or redeploying your applications.

A Web service security architecture supports message confidentiality through:

1. Encryption of the message itself
2. Transportation-level confidentiality schemes

When a message is encrypted to ensure message confidentiality, any portion of it can be encrypted subject to the requirements of the encoding scheme used to include the encrypted data in the message. A Web service security architecture should support the XML-Encryption standard for encoding arbitrary encrypted data in an XML document, applied either through the WS-Security standard for encapsulating encrypted data in a SOAP message, or by arbitrarily replacing portions of the SOAP envelope (including the whole envelope as necessary) with an encrypted data element.

When sending messages, a managed external service can generate WS-Security-encoded and encrypted message bodies, which ensures that the message itself is confidential. It may also replace the entire SOAP body with a single XML-Encryption-encoded element for communicating with external services that don't comply with the WS-Security standard. Either system, when combined with proper certificate management, will ensure complete confidentiality between message producer and message consumer.

Transportation-level confidentiality is appropriate for applying confidentiality between point-to-point communication and for use when an external managed service does not support message-level confidentiality. In order to provide transportation-level security and authentication, SSL may be used as an option on any message transportation system



that may use it. This can be used with client-side certificate presentation and validation for additional transportation-level authentication, and will ensure that even if message-level encryption is not used at least the transmission of messages will be confidential.

A Web service security architecture supports message authentication through the use of:

1. Digital signatures included in the message itself
2. Message-level Proof of Possession authentication schemes
3. Transportation-level authentication schemes
4. Security assertions made by trusted partners

In the case of a digital signature included in the message itself, a message router may approve a digital signature encoded using the XML-Digital Signature standard and included in the message using either the W3C SOAP Security Extension Digital Signature recommendation or the WS-Security standard for message-level security. An external service may also provide a digital signature in either one of these formats. In order to avoid "replay attacks," you should only sign and accept signatures that include the complete SOAP body of the message being transmitted.

When cryptographic message-level authentication schemes aren't used, the Web service security architecture should comply with the WS-Security standard for Proof of Possession Claims for authentication. While these are generally considered less secure, as they typically rely on unencrypted username/password schemes for authentication, they are invaluable when interacting with services that don't support more secure message-level authentication schemes. Organizations may also wish to deploy a Proof of Possession system where service performance is critical.

Transportation-level authentication schemes are most appropriate when the performance of a particular service invocation is critical, or a persistent, cryptographic authentication scheme is not desirable. Using one of these schemes, a client of the service would provide security credentials using a mechanism appropriate for the transportation scheme being employed (such as a user name and password or digest when using HTTP/Basic over SSL). While these schemes don't provide

all of the benefits of a message-oriented authentication system, they are usually faster and easier to support with legacy clients.

When a trust relationship exists between two organizations, they may choose to allow each other to share sign-on responsibilities and trust assertions made by each other about principal-level authentication. A Web service security architecture should accept SAML assertions in incoming messages, and evaluate the originator of the SAML assertions for an existing trust relationship. When one exists for a particular security realm, consider a valid security assertion by a trusted partner to be equivalent to a local authentication request.

Once a particular inbound message has been authenticated by a message router, a set of security assertions is created and assigned to that message (or appended to it in the case of assertion-based authentication). These also come in the form of SAML assertions, which will then be propagated through the network. If an inbound message then results in an outbound message external service, those security assertions will be translated into authentication information (including SAML assertions), which may be provided to the external service provider.

Web services by their very nature bring together different consumers and providers, each of which may have its own user, authentication, and authorization model. In order to seamlessly bring together these different security systems and user models, a Web service security architecture should provide user management at the message router level.

A Web service security architecture should allow a site administrator to define a security realm for each category of system that has a different and incompatible security model. These can be managed by an existing directory server (such as an LDAP server or Active Directory Domain). When services are enabled, they are assigned to a particular security realm, which will then be used for managing authentication and authorization decisions.

Site security administrators may also define mappings between different security principals. These mappings can then be used to allow seamless user translation between different security realms. For example, a mapping can be

used to translate from the user name allowed by a message router to the user name required by an external managed service, where each may be in a different security realm. This allows organizations to provide single sign-on functionality without modifying their existing applications, which is critical when some services are external to the organization.

While many enterprises have made an investment in a public key infrastructure to complement their other security initiatives, for those that have not, the reliance on public key cryptography in a Web services architecture can be daunting.

A Web service security architecture supports key management using:

1. An internal key management facility
2. An organization's existing XKMS service
3. An organization's existing Java Key Store architecture.

Because many organizations have existing public key infrastructures that provide them with an XKMS service deployed inside their organization, or a Java Key Store-based system that allows access to central registration and certificate authorities, a key management solution allows the use of either one of these types of enterprise key management integration.

However, for organizations that don't have an existing public key infrastructure, but which want to use public key-based Web service security systems (such as XML Digital Signatures or Encryption), the organization should provide a key management facility internally that provides a centralized security realm based on certificate management, distributed private key management, and association of certificates and keys with user management facilities.

Conclusion

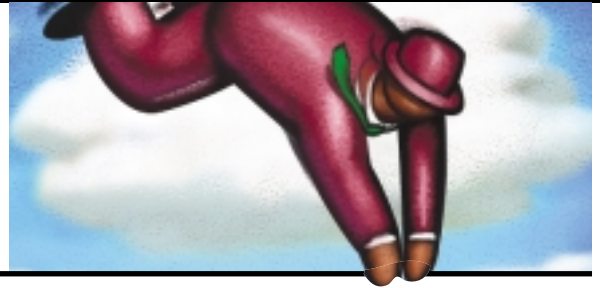
Security is one of the most critical aspects of your Web services management infrastructure. While the security standards are still rapidly evolving, they are certainly mature enough for enterprises to build robust, secure systems today. The benefit of exposing business systems as services is compelling. Understanding the differences in implementing a message-oriented architecture will allow you to take advantage of these new business opportunities. ©

Merant

www.merant.com

Developing J2EE 1.4 Web Services on the Fly

Providing more design time



As Web services technology becomes pervasive, the beta release of Java 2 Enterprise Edition (J2EE) 1.4, which focuses primarily on Web services, flags a milestone in the Web services developer community. Sun's J2EE Reference Implementation (RI) helps developers to easily understand the technology through its robust implementation of the specification. The J2EE 1.4 specification (JSR-151) adds a lot of functionality to the platform. The core theme of J2EE 1.4 circumvents Web services. It has incorporated most of the Web services-related JSRs in its specification, including JSR-109 (Implementing Enterprise Web Services), JSR-101 (Java APIs for XML-based RPC), JSR-67 (Java APIs for XML Messaging 1.0), and JSR-93 (Java API for XML Registries 1.0).

AUTHOR BIO:



Arulazi Dhesiaseelan has been involved in designing and building Java-based applications and SDK for more than 3 years. He was also involved in the API development of the UDDI4j project hosted at <http://uddi4j.org>. Arul works with Hewlett-Packard (India Software Operations) in Bangalore.

He is currently involved in the development of an open-service framework for mobile infrastructures.

ARULD@ACM.ORG

Basically, a Web service client can access J2EE applications in two ways. First, the client can access a Web service created with JAX-RPC. Second, the client can access an EJB Web service (only stateless session beans) through its Web service endpoints. The bottom line is that JAX-RPC uses a servlet to implement the Web service. This helps in exposing the existing stateless session beans as Web services.

In this article, I'll deal with the development of a Web service using the J2EE 1.4 platform. I'll use the JAX-RPC APIs that form the basis for developing J2EE 1.4 Web services on the fly.

Creating a Web Service with JAX-RPC

Remote Procedure Call (RPC) is a mechanism through which you invoke procedures (or services) from a remote client. In the Java world, we have the Java API for XML-based RPC (JAX-RPC) in which service invocation takes place through an XML-based protocol called Simple Object Access Protocol (SOAP) over HTTP transport. Since the J2EE 1.4 platform uses standard technologies such as

XML, SOAP, and HTTP, it is easy for developers to create Web services that are platform independent and interoperable.

“

The typical life cycle of a Web service has three stages: develop the Web service, deploy the Web service, and invoke the Web service ”

The typical life cycle of a Web service has three stages: develop the Web service, deploy the Web service, and invoke the Web service. We'll follow this life cycle in

developing our Web services using J2EE 1.4. We will consider the familiar stock quote application for our illustration. A stock quote service is deployed in the server. The client invokes the service by passing a stock symbol. Here, the service and the client are developed using the JAX-RPC APIs. Figure 1 shows the interactions between the Stock quote Web service and the stock quote client. Stubs are the client-side proxies used to communicate with the service. Ties are the classes the server needs to communicate with a remote client. Stubs and ties are low-level classes that perform similar functions, i.e., stubs on the client side and ties on the server side.

Building and Deploying the Service

The steps that are involved in building and deploying the service are:

1. Define the interface class for the service.
2. Implement the interface class.
3. Compile the classes and generate WSDL.
4. Package the classes as an EAR file.
5. Deploy the service.

Altova

www.xmlj.altova.com/authentic5

“

The core theme of J2EE 1.4 circumvents Web services ”

Service Interface

Our stock quote service has one method, `getQuote`, which takes the symbol as the argument and returns the value of the stock symbol. The interface definition for the stock quote service is shown in Listing 1.

Service Implementation

The service implementation `StockQuoteImpl` implements the interface `StockQuoteIF`. This connects to an Axis stock quote service that is running on the `www.xmethods.net` server. The `getQuote` request made to this service returns an XML document that contains the value of the stock. If you are behind a firewall, you need to specify the `proxyHost` and `proxyPort` properties in order to access the service. The implementation for the stock quote service is shown in Listing 2.

Generating the WSDL File

J2EE 1.4 provides a tool called `wscompile.bat`, which generates the WSDL file for the service. The Ant target `generate-wsdl` does this for you and runs this tool as follows:

```
wscompile.bat -define -d build/server
-nd build/server -classpath
build/server service-config.xml
```

The generated WSDL file `MyStockQuoteService.wsdl` is shown in Listing 3.

The Ant target `build-service` compiles the service classes and the generation of the WSDL file for the service.

Packaging the Service

To package the service, use the ant target `package-service`, which packages the service and its dependent files in a `stockquote.ear` file. This EAR file bundles the

`stockquote-jaxrpc.war` file in it. The contents of these WAR and EAR files are:

```
stockquote-jaxrpc.war
META-INF/
META-INF/MANIFEST.MF
WEB-INF/
WEB-INF/classes/
WEB-INF/classes/stockquote/
MyHelloService.wsdl
WEB-
INF/classes/stockquote/StockQuoteIF.class
WEB-
INF/classes/stockquote/StockQuoteImpl.class
WEB-INF/mapping.xml
WEB-INF/web.xml
WEB-INF/webservices.xml

stockquote.ear
META-INF/
META-INF/MANIFEST.MF
```

```
stockquote-jaxrpc.war
META-INF/application.xml
META-INF/sun-j2ee-ri.xml
```

Deploying the Service

J2EE 1.4 provides a tool called `deploy-tool.bat` for deploying Web applications. The developed EAR file is deployed in the server using this tool. This can be done with the help of the Ant target `deploy-service`. As a precondition, you need to start the database server from the command prompt by typing

```
%J2EE_HOME%/bin/cloudscape.bat -start
```

Start the J2EE server from the command prompt by typing

```
%J2EE_HOME%/bin/j2ee.bat -verbose
```

to start the server in verbose mode. The Ant

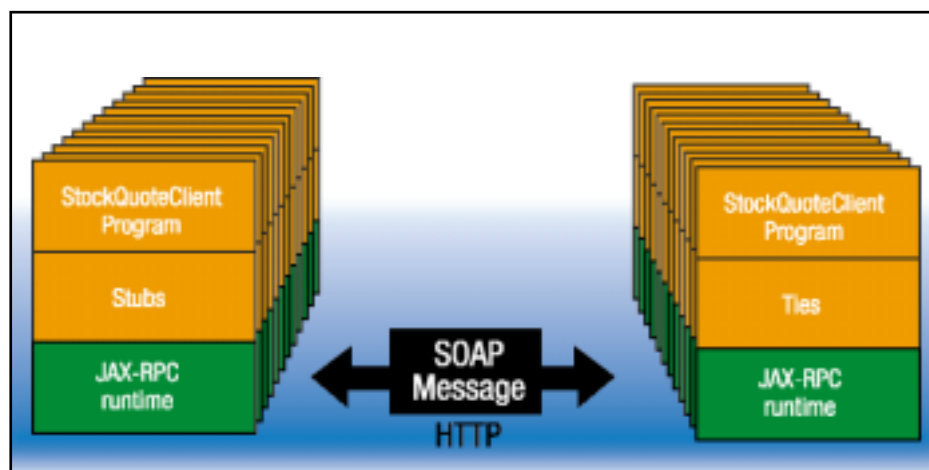


FIGURE 1 The StockQuote application at runtime

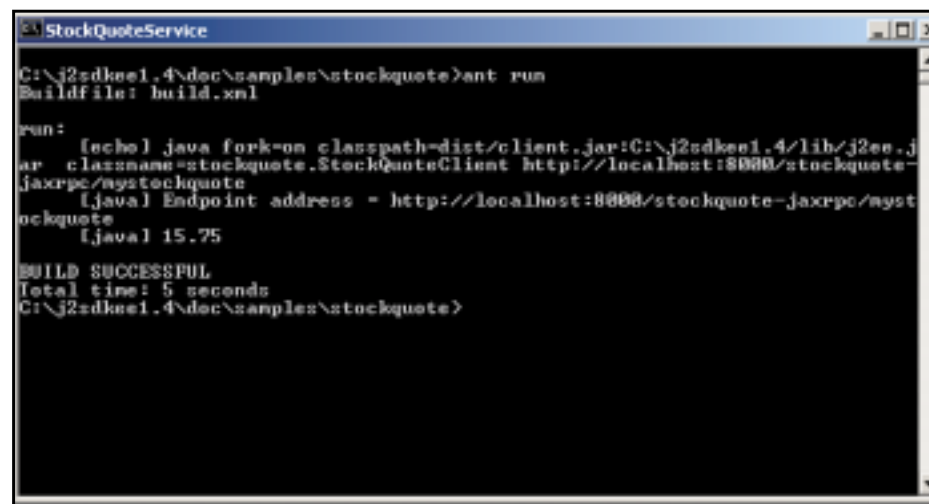


FIGURE 2 Output of the StockQuoteClient Program

target runs the deploy tool as follows:

```
deploytool.bat -id stockquote-jaxrpc -
deployModule stockquote.ear
```

This deploys the stock quote service in the server. To verify that the service has been successfully deployed, execute the Ant target “list”, which lists all the applications that are deployed on the server. You can verify the successful service deployment by typing the following URL in your browser:

```
http://localhost:8000/stockquote-jaxr-
pc/mystockquote?WSDL
```

If you can see the WSDL file on your browser, the service is deployed successfully.

Building and Running the Client

The steps involved in building and running the client are:

1. Generate the stubs.
2. Code the client class.
3. Compile the client class.
4. Package the classes as a JAR file.
5. Run the client.

Generate the Stubs

Stubs are the client-side proxies for the service. J2EE 1.4 provides a tool called `wscmcompile.bat` for generating these stubs. The Ant target `generate-stubs`, defined in our `build.xml`, does this task for you. This target runs the tool as follows:

```
wscmcompile.bat -gen:client -d
build/client -classpath build/server
client-config.xml
```

The `wscmcompile.bat` tool generates files based on the information that it reads from the `MyStockQuoteService.wsdl`. The `-gen:client` option instructs the tool to generate the client-side classes called stubs. The `-d` option instructs the tool in which directory the generated stubs are to be placed. The `-classpath` option instructs the tool where to find the input classes. The tool reads the config file “`client-config.xml`” which has the information about the location of the WSDL file. The client configuration file `client-config.xml` is shown in Listing 4.

Service Client

Once the service is defined, implemented, and deployed, it is ready for access by the clients. The client uses the generated stubs to interact with the service. This is a static client as it was created before runtime. The client takes the service endpoint as the input argument, and invokes the `getQuote` method by passing the symbol “HPQ”. The service returns the value of the symbol to the system console. This is the real-time quote of the symbol with a delay of 20 minutes. The client for accessing the stock quote service is shown in Listing 5.

Compiling and Packaging the Client

The Client sources along with the stubs are compiled and packaged as a JAR file by the Ant targets `compile-client` and `package-client`. This generates a file – `client.jar` that is used in invoking the stock quote service.

Setup Instructions for Running the Web Service

Required Software:

- Download Ant 1.5.3 from <http://ant.apache.org>
- Download J2SE 1.4.1 from <http://java.sun.com/j2se/1.4.1/download.html>
- Download J2EE 1.4 Beta from <http://java.sun.com/j2ee/1.4/download.html#sdk>

Setting Up Environment Variables

To run the StockQuote application, you need to set the following environment variables:

- `J2EE_HOME=C:\j2sdkee1.4`
- `JAVA_HOME=C:\j2sdk1.4.1_02`
- `ANT_HOME=H:\apache-ant-1.5.3`
- `PATH=%JAVA_HOME%\bin;%J2EE_HOME%\bin;%ANT_HOME%\bin;%PATH%`

Invoke the Service Using the Web Service Client

Download the source code of the developed Web service. Extract this under “`J2EE_HOME\doc\samples`”. The “`build.xml`” file bundled with this application defines all of the tasks discussed above as Ant targets. Before proceeding further, you need to change the proxy-related information in the `StockQuote-Impl.java` class.

As a precondition, you need to start

the database and J2EE servers by using the following commands:

```
%J2EE_HOME%\bin\cloudscape.bat -start
"%J2EE_HOME%\bin\j2ee.bat -verbose
```

This starts the server in verbose mode. Wait until the command window displays “J2EE server startup complete.” Now you are ready to execute the Ant targets that will help you build, deploy, and invoke the Web service.

Execute the following targets:

- **Ant build-service:** Compiles the service sources.
- **Ant package-service:** Packages the service classes and related files in an EAR file.
- **Ant deploy-service:** Deploys the service in the server.
- **Ant list:** Lists the deployed service.
- **Ant build-client:** Compiles the client sources.
- **Ant package-client:** Packages the client classes as a JAR file.
- **Ant run:** Invokes the service. Figure 2 shows the output of the client after invoking the service. It returns the value of the symbol with a delay of 20 minutes.
- **Ant undeploy:** Undeploys the service from the server.

Conclusion

Tools always ease the development of application software. With the help of robust tools, developers can spend more time designing the applications than developing the applications. The current release of the J2EE 1.4 platform provides some basic tools that ease the development of Web services. The J2EE 1.4 final release plans to have support for Web Services – Interoperability (WS-I) basic profiles. Developing and deploying Web services on the fly is a reality with the arrival of the J2EE 1.4 platform.

References

- *J2EE 1.4 Platform Home:* <http://java.sun.com/j2ee/>
- *J2EE 1.4 Specification:* <http://java.sun.com/j2ee/j2ee-1.4-pfd2-spec.pdf>
- *JSR 151:* www.jcp.org/en/jsr/detail?id=151
- *The J2EE Tutorial Addendum:* <http://java.sun.com/j2ee/1.4/docs/tutorial/index.html>



Listing 1: StockQuoteIF.java

```
package stockquote;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface StockQuoteIF extends Remote {
    public float getQuote(String symbol) throws
        RemoteException, Exception;
}
```

Listing 2: StockQuoteImpl.java

```
package stockquote;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.net.URL;

public class StockQuoteImpl implements StockQuoteIF {

    // 20 minute delayed stock quote
    public float getQuote (String symbol) throws Exception {

        System.setProperty("http.proxyHost", "rio");//Replace
        with your proxy host
        System.setProperty("http.proxyPort", "8088");//Replace
        with your proxy port
        URL url = new
        URL("http://services.xmethods.net/axis/getQuote?s=" + sym-
        bol);

        DocumentBuilderFactory dbf =
        DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();

        Document doc = db.parse( url.toExternalForm() );
        Element elem = doc.getDocumentElement();
        NodeList list =
        elem.getElementsByTagName("stock_quote");

        if (list != null && list.getLength() != 0) {
            elem = (Element) list.item(0);
            list = elem.getElementsByTagName("price");
            elem = (Element) list.item(0);
            String quoteStr = elem.getAttribute("value");

            return Float.valueOf(quoteStr).floatValue();
        }
        return(0);
    }
}
```

Listing 3: MyStockQuoteService.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions name="MyStockQuoteService"
targetNamespace="urn:Foo"
xmlns:tns="urn:Foo"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
    <types/>
    <message name="StockQuoteIF_getQuote">
        <part name="String_1" type="xsd:string"/>
    </message>
    <message name="StockQuoteIF_getQuoteResponse">
        <part name="result" type="xsd:float"/>
    </message>
```

```
</message>
<portType name="StockQuoteIF">
    <operation name="getQuote" parameterOrder="String_1">
        <input message="tns:StockQuoteIF_getQuote"/>
        <output
        message="tns:StockQuoteIF_getQuoteResponse"/></opera-
        tion></portType>
    <binding name="StockQuoteIFBinding"
    type="tns:StockQuoteIF">
        <operation name="getQuote">
            <input>
                <soap:body
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                use="encoded" namespace="urn:Foo"/></input>
                <output>
                    <soap:body
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    use="encoded" namespace="urn:Foo"/></output>
                    <soap:operation soapAction=""/></operation>
                <soap:binding
                transport="http://schemas.xmlsoap.org/soap/http"
                style="rpc"/></binding>
                <service name="MyStockQuoteService">
                    <port name="StockQuoteIFPort"
                    binding="tns:StockQuoteIFBinding">
                        <soap:address location="http://localhost:8000/stock-
                        quote-jaxrpc/mystockquote"/></port></service>
                </definitions>
```

Listing 4: Client-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
    xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
    <wsdl location="http://localhost:8000/stockquote-jaxr-
    pc/mystockquote?WSDL"
        packageName="stockquote"/>
</configuration>
```

Listing 5: StockquoteClient.java

```
package stockquote;

import javax.xml.rpc.Stub;

public class StockQuoteClient {

    private String endpointAddress;

    public static void main(String[] args) {

        System.out.println("Endpoint address = " +
        args[0]);
        try {
            Stub stub = createProxy();
            stub._setProperty(javax.xml.rpc.Stub.END-
            POINT_ADDRESS_PROPERTY,
                args[0]);
            StockQuoteIF stock = (StockQuoteIF)stub;
            System.out.println(stock.getQuote("HPQ"));
        } catch (Exception ex) {
            ex.printStackTrace();
        }

        private static Stub createProxy() {
            // Note: MyStockQuoteService_Impl is implementa-
            tion-specific.
            return (Stub)(new
            MyStockQuoteService_Impl().getStockQuoteIFPort());
        }
    }
}
```

Download the code at
sys-con.com/webservices

Quest Software

www.java.quest.com/jcsr/ws

Written by Dr. Mark Little & Dr. Jim Webber

Introducing WS-Transaction

Part 1

The basis of the WS-Transaction protocol

In July 2002, BEA, IBM, and Microsoft released a trio of specifications designed to support business transactions over Web services. These specifications, BPEL4WS, WS-Transaction, and WS-Coordination, together form the bedrock for reliably choreographing Web services-based applications, providing business process management, transactional integrity, and generic coordination facilities respectively.

In our previous article (*WSJ*, Volume 3, issue 5), we introduced WS-Coordination, a generic coordination framework for Web services, and showed how the WS-Coordination protocol can be augmented to support coordination in arbitrary application domains.

This article introduces the first publicly available WS-Coordination-based

protocol – Web Services Transaction – and shows how WS-Transaction provides atomic transactional coordination for Web services.

Transactions

Distributed systems pose reliability problems that are not frequently encountered in centralized systems. A distributed system consisting of a number of computers connected by a network can be subject to independent failure of any of its components, such as the computers themselves, network links, operating systems, or individual applications. Decentralization allows parts of the system to fail while other parts remain functioning, which leads to the possibility of abnormal behavior of executing applications.

Consider the case of a distributed system where the individual computers provide a selection of useful services that can

be utilized by an application. It is natural that an application that uses a collection of these services requires that they behave consistently, even in the presence of failures. A very simple consistency requirement is that of *failure atomicity*: the application either terminates normally, producing the intended results, or is aborted, producing no results at all. This failure atomicity property is supported by atomic transactions, which have the following familiar ACID properties:

- **Atomicity:** The transaction completes successfully (commits) or if it fails (aborts) all of its effects are undone (rolled back);
- **Consistency:** Transactions produce consistent results and preserve application specific invariants;
- **Isolation:** Intermediate states produced while a transaction is executing are not visible to other transactions. Furthermore, transactions appear to execute serially, even if they are actually executed concurrently. This is typically achieved by locking resources for the duration of the transaction so that they cannot be acquired in a conflicting manner by another transaction;
- **Durability:** The effects of a committed transaction are never lost (except by a catastrophic failure).

A transaction can be terminated in two ways: *committed* or *aborted* (rolled back). When a transaction is committed, all

AUTHOR BIO:



Dr. Jim Webber is an architect and Web services fanatic at Arjuna Technologies where he works on Web services transactioning and Grid computing technology. Prior to joining Arjuna Technologies, he was the lead devel-

oper with Hewlett-Packard working on their BTP-based Web Services Transactions product – the industry's first Web Services Transaction solution. An active speaker and Web Services proponent, Jim is the coauthor of "Developing Enterprise Web Services."

JIM.WEBBER@ARJUNA.COM



Dr. Mark Little is chief architect for Arjuna Technologies Limited, a spin-off from Hewlett-Packard that develops transaction technologies for J2EE and Web services. Previously, Mark was a distinguished engineer and architect at HP

Middleware, where he led the transactions teams. He is a member of the expert group for JSR 95, and JSR 117, is the specification lead for JSR 156, and is active on the OTS Revision Task Force and the OASIS Business Transactions Protocol specification. Mark is the coauthor of J2EE 1.4 Bible and Java Transactions for Architects.

MARK.LITTLE@ARJUNA.COM

changes made within it are made durable (forced onto stable storage such as disk). When a transaction is aborted, all changes made during the lifetime of the transaction are undone. In addition, it is possible to nest atomic transactions, where the effects of a nested action are provisional upon the commit/abort of the outermost (top-level) atomic transaction.

Why ACID Transactions May Be Too Strong

Traditional transaction processing systems are sufficient to meet requirements if an application function can be represented as a single top-level transaction. However, this is frequently not the case. Top-level transactions are most suitably viewed as short-lived entities, performing stable state changes to the system; they are less well suited for structuring long-lived application functions that run for minutes, hours, days, or longer. Long-lived, top-level transactions may reduce the concurrency in the system to an unac-

ceptable level by holding on to resources (usually by locking) for a long time. Furthermore, if such a transaction aborts, much valuable work already performed will be undone.

Given that the industry is moving toward a loosely coupled, coarse-grained, B2B interaction model supported by Web services, it has become clear that the semantics of traditional ACID transactions are unsuitable for Web-scale deployment. Web services-based transactions differ from traditional transactions in that they execute over long periods, they require commitments to the transaction to be negotiated at runtime, and isolation levels have to be relaxed.

WS-Transaction

In the past, making traditional transaction systems talk to one another was a rarely achieved holy grail. With the advent of Web services, we have an opportunity to leverage an unparalleled interoperability technology to splice together existing

transaction-processing systems that already form the backbone of enterprise-level applications.

WS-Coordination Foundations

An important aspect of WS-Transaction that differentiates it from traditional transaction protocols is that a synchronous request/response model is not assumed. This model derives from the fact that WS-Transaction (see Figure 1) is layered upon the WS-Coordination protocol whose own communication patterns are asynchronous by default.

In our last article, we looked at how WS-Coordination provides a generic framework for specific coordination protocols, like WS-Transaction, to be plugged in. Remember that WS-Coordination provides only context management – it allows contexts to be created and activities to be registered with those contexts. WS-Transaction leverages the context management framework provided by WS-Coordination in two ways. First, it extends

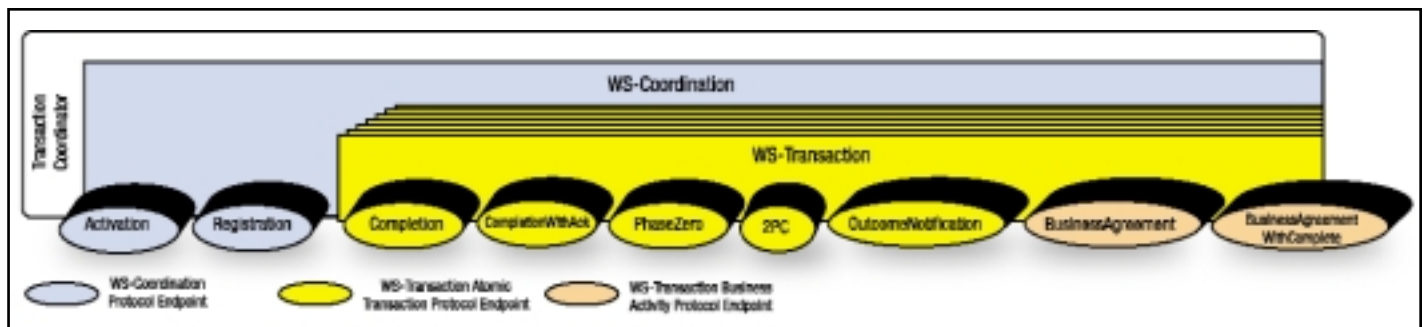


FIGURE 1 | WS-Coordination and WS-Transaction

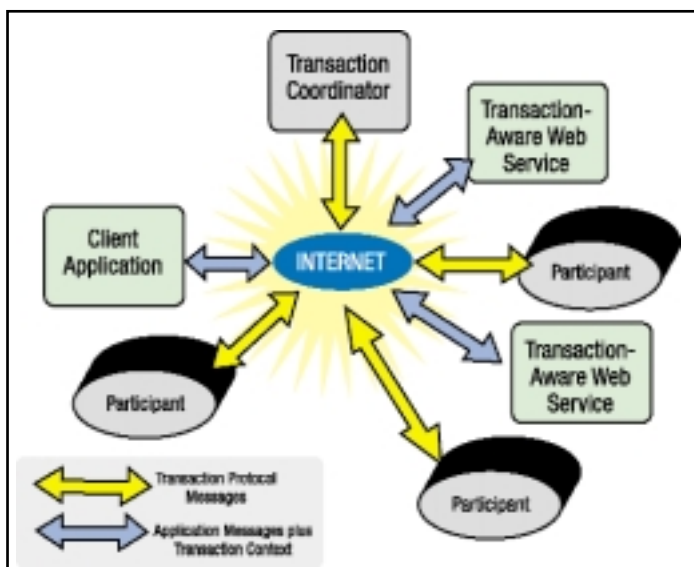


FIGURE 2 | WS-Transaction global view

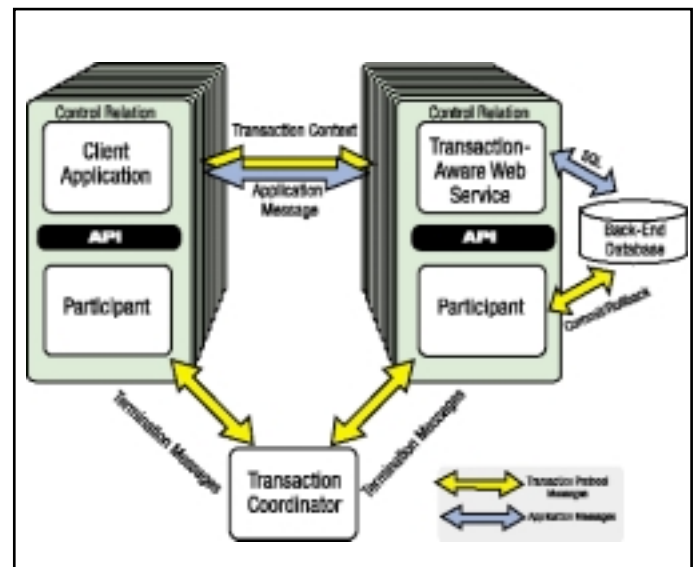


FIGURE 3 | Transactional service and participant

the WS-Coordination context to create a transaction context. Second, it augments the activation and registration services with a number of additional services (Completion, CompletionWithAck, PhaseZero, 2PC, Outcome Notification, BusinessAgreement, and BusinessAgreement-WithComplete) and two protocol message sets (one for each of the transaction models supported in WS-Transaction) to build a full-fledged transaction coordinator on top the WS-Coordination protocol infrastructure.

WS-Transaction Architecture

In common with other transaction protocols (like OTS and BTP), WS-Transaction supports the notion of the service and participant as distinct roles, making the distinction between a transaction-aware service and the participants that act on behalf of the service during a transaction: transactional services deal with business-level protocols, while the participants handle the underlying WS-Transaction protocols, as shown in Figure 2.

A transaction-aware service encapsulates the business logic or work that is required to be conducted within the scope of a transaction. This work cannot be confirmed by the application unless the transaction also commits and so control is ultimately removed from the application and placed into the transaction's domain.

The participant is the entity that, under the dictates of the transaction coordinator, controls the outcome of the work performed by the transaction-aware Web service. In Figure 2 each service is shown with one associated participant that manages the transaction protocol messages on behalf of its service, while in Figure 3, we see a close-up view of a single service, and a client application with their associated participants.

The transaction-aware Web service and its participant both serve a shared transactional resource, and there is a control relationship between them through some API – which on the Java platform is JAXTX. In the example in Figure 3, we assume that the database is accessed through a transactional JDBC database driver, where SQL statements are sent to the database for processing via that driver, but where those statements will be tentative and only

commit if the transaction does. In order to do this, the driver/database will associate a participant with the transaction which will inform the database of the transaction outcome. Since all transactional invocations on the Web service carry a transaction context, the participant working with the database is able to identify the work that the transactional service did within the scope of a specific transaction and either commit or roll back the work.

At the client end, things are less complex. Through its API, the client application registers a participant with the transaction through which it controls transaction termination.

WS-Transaction Models

Given that we've already seen that traditional transaction models are not appropriate for Web services, we must pose the question, "What type of model or protocol is appropriate?" The answer to that question is that no one specific protocol is likely to be sufficient, given the wide range of situations that Web service transactions are likely to be deployed within. Hence the WS-Transaction specification proposes two distinct models, each supporting the semantics of a particular kind of B2B interaction. In the following sections we'll discuss these two models, but for the sake of brevity we ignore possible failure cases.

Note: as with WS-Coordination, the two WS-Transaction models are extensible, allowing implementations to tailor the protocols as they see fit (e.g., to suit their deployment environments). For clarity, we'll discuss only the "vanilla" protocols and leave proprietary extensions out of the picture.

Atomic Transactions (AT)

An atomic transaction, or AT, is similar to traditional ACID transactions and intended to support short-duration interactions where ACID semantics are appropriate.

Within the scope of an AT, services typically enroll transaction-aware resources, such as databases and message queues, indirectly as participants under the control of the transaction. When the transaction terminates, the outcome decision of the AT is then propagated to each enlisted

resource via the participant, and the appropriate commit or rollback actions are taken.

This protocol is similar to those employed by traditional transaction systems that already form the backbone of an enterprise. It is assumed that all services (and associated participants) provide ACID semantics and that any use of atomic transactions occurs in environments and situations where this is appropriate: in a trusted domain, over short durations.

To begin an atomic transaction, the client application first locates a WS-Coordination coordinator Web service that supports WS-Transaction. Once found, the client sends a WS-Coordination CreateCoordinationContext message to the activation service specifying <http://schemas.xmlsoap.org/ws/2002/08/wstx> as its coordination type and will get back an appropriate WS-Transaction context from the activation service. The response to the CreateCoordinationContext message, the transaction context, has its CoordinationType element set to the WS-Transaction at namespace, <http://schemas.xmlsoap.org/ws/2002/08/wstx>, and also contains a reference to the atomic transaction coordinator endpoint (the WS-Coordination registration service) where participants can be enlisted, as shown in Listing 1 (the code for this article can be found online at www.sys-con.com/webservices/sourcecode.cfm).

After obtaining a transaction context from the coordinator, the client application then proceeds to interact with Web services to accomplish its business-level work. With each invocation on a business Web service, the client inserts the transaction context into a SOAP header block, such that the each invocation is implicitly scoped by the transaction – the toolkits that support WS-Transaction-aware Web services provide facilities to correlate contexts found in SOAP header blocks with back-end operations.

Once all the necessary application-level work has been completed, the client can terminate the transaction, with the intent of making any changes to the service state permanent. To do this, the client application first registers its own participant for the Completion or Completion-WithAck protocol. Once registered, the

Global Knowledge

www.globalknowledge.com

participant can instruct the coordinator either to try to commit or roll back the transaction. When the commit or rollback operation has completed, a status is returned to the participant to indicate the outcome of the transaction. The CompletionWithAck protocol goes one step further and insists that the coordinator must remember the outcome until it has received acknowledgment of the notification from the participant.

While the completion protocols are straightforward, they hide the fact that in order to resolve to an outcome several other protocols need to be executed.

The first of these protocols is the optional PhaseZero. The PhaseZero protocol is typically executed where a Web service needs to flush volatile (cached) state, which may be in use to improve performance of an application to a database prior to the transaction committing. Once flushed, the data will then be controlled by a two-phase aware participant.

All PhaseZero participants are told that the transaction is about to complete (via the PhaseZero message) and they can respond with either the PhaseZeroCompleted or Error message; any failures at

this stage will cause the transaction to roll back. The corresponding interfaces through which the participant and transaction coordinator exchange PhaseZero messages are shown in Listing 2.

After PhaseZero, the next protocol to execute in WS-Transaction is 2PC. The 2PC (two-phase commit) protocol is at the heart of WS-Transaction atomic transactions and is used to bring about the consensus between participants in a transaction such that the transaction can be terminated safely.

The 2PC protocol is used to ensure atomicity between participants, and is based on the classic two-phase commit with presumed abort technique. During the first phase, when the coordinator sends the prepare message, a participant must make durable any state changes that occurred during the scope of the transaction, such that these changes can either be rolled back or committed later. That is, any original state must not be lost at this point as the atomic transaction could still roll back. If the participant cannot prepare then it must inform the coordinator (via the aborted message) and the transaction will ultimately roll

back. If the participant is responsible for a service that did not do any work during the course of the transaction, or at least did not do any work that modified any state, it can return the read-only message and it will be omitted from the second phase of the commit protocol. Otherwise, the prepared message is sent by the participant.

Assuming no failures occurred during the first phase, in the second phase the coordinator sends the commit message to participants, who will make permanent the tentative work done by their associated services.

If a transaction involves only a single participant, WS-Transaction supports a one-phase commit optimization. Since there is only one participant, its decisions implicitly reach consensus, and the coordinator need not drive the transaction through both phases. In the optimized case, the participant will simply be told to commit and the transaction coordinator need not record information about the decision since the outcome of the transaction is solely for that single participant.

To place the 2PC protocol concepts into a Web services context, the interfaces of the transaction coordinator and corresponding 2PC participant are defined by the WSDL shown in Listing 3. The two WSDL portType declarations are complementary; for instance, where the 2PC-ParticipantPortType exposes the prepare operation to allow a coordinator to put it into the prepared state; the 2PCCoordinatorPortType has the prepared operation to allow participants to inform the coordinator that they have indeed moved to the prepared state.

Figure 4 (redrawn from the WS-Transaction specification <http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnglobspec/html/ws-transaction.asp>) shows the state transitions of a WS-Transaction atomic transaction and the message exchanges between coordinator and participant; the coordinator generated messages are shown in the solid line, whereas the participant messages are shown by dashed lines.

Once the 2PC protocol has finished, the Completion or CompletionWithAck protocol that originally began the termination of the transaction can complete,

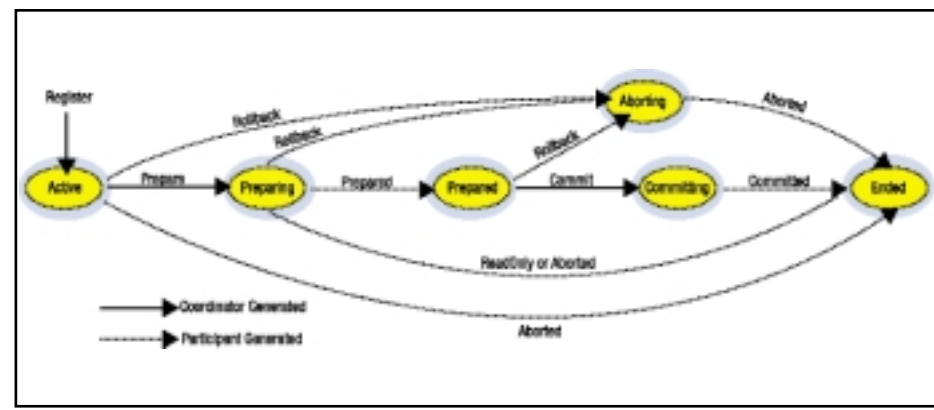


FIGURE 4 Two-phase Commit State Transitions

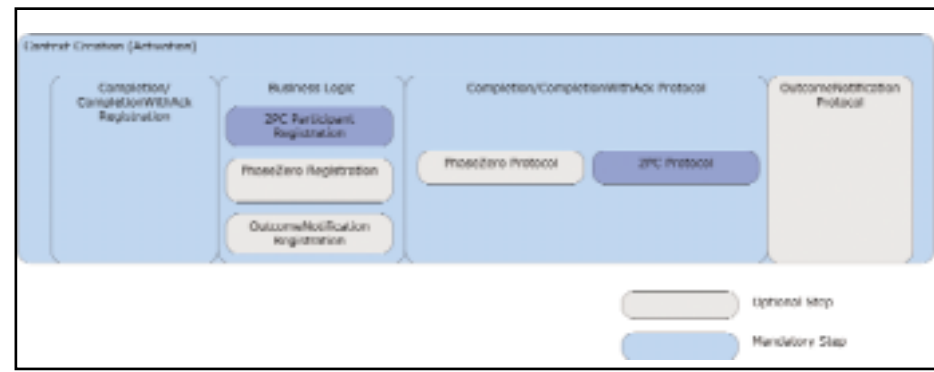


FIGURE 5 The AT Model

and inform the client application whether the transaction was committed or rolled back. In addition, some services may have registered an interest in the completion of a transaction, and they will be informed via the OutcomeNotificationProtocol.

Like the PhaseZero protocol, the OutcomeNotificationProtocol is an optional protocol that some services will register for so that they can be informed when the transaction has completed, typically so that they can release resources (e.g., put a database connection back into the pool of connections).

Any registered OutcomeNotification participants are invoked after the transaction has terminated and are told the state in which the transaction completed (the coordinator sends either the Committed or Aborted message). Since the transaction has terminated, any failures of participants at this stage are ignored – OutcomeNotification is essentially a courtesy and has no bearing on the outcome of the transaction.

Finally, after having gone through each

of the stages in an AT, we can now see the intricate interweaving of individual protocols that goes to make up the AT as a whole in Figure 5.

Coordinating Atomic Transactions on the Web

Transactions come to the fore when computational work with real-world financial implications must be executed. That being said, what better place to demonstrate the use of WS-Transaction than in online retail, where organizations live and die based on the quality of their customer service?

Take the situation where a customer needs to purchase a new set of formal-wear items, including a suit, tie, and shoes. Obviously it wouldn't be advisable for the customer to go into a formal situation without any of these, so the purchase of all three is a prerequisite for the completion of a business transaction.

In the first instance, let's consider the situation where a single retailer can offer a choice of all three items (see Figure 6).

In Figure 6 the retailer's Web service

acts as a gateway to some back-end services that it also hosts. In this case, since the trust domain is entirely within one organization it's safe to use an Atomic Transaction to scope the purchases that the client application makes into a single logical unit of work.

A typical use case for the architecture shown in Figure 6 is:

1. The client application begins its interaction with the online store, which creates an AT at the back end.
2. The client purchases items, which are then locked and other transactions cannot see them.
3. When the client application decides to buy the items, the AT is committed and its tentative work is made permanent, unless there are faults, in which case the work is rolled back.
4. The termination status of the transaction is reported back to the customer as a purchase successful/unsuccessful message.

Aside from the fact that we are using Web services to host application logic, this is a textbook transactions example, which goes to strengthen the view that ATs are meant to be used within the kinds of close trust domains that traditional transaction processing infrastructure operates within. The Web services aspects of the protocol simply mean that proprietary transaction processing systems can interoperate, but this does not change their fundamental trust characteristics – which must be borne in mind by developers lest they expose lockable resources to the Web!

Summary

In this article we've seen how WS-Coordination has been used to provide the basis of the WS-Transaction protocol. We have also discussed the first transaction model that WS-Transaction supports: Atomic Transaction. This protocol is suitable for supporting short-lived transactions between trusted Web services where the possibility for malicious locking of resources is low.

In our next article, we'll introduce the Business Activity protocol and show how it can provide the basis for higher-level business process management and workflow technology. ©

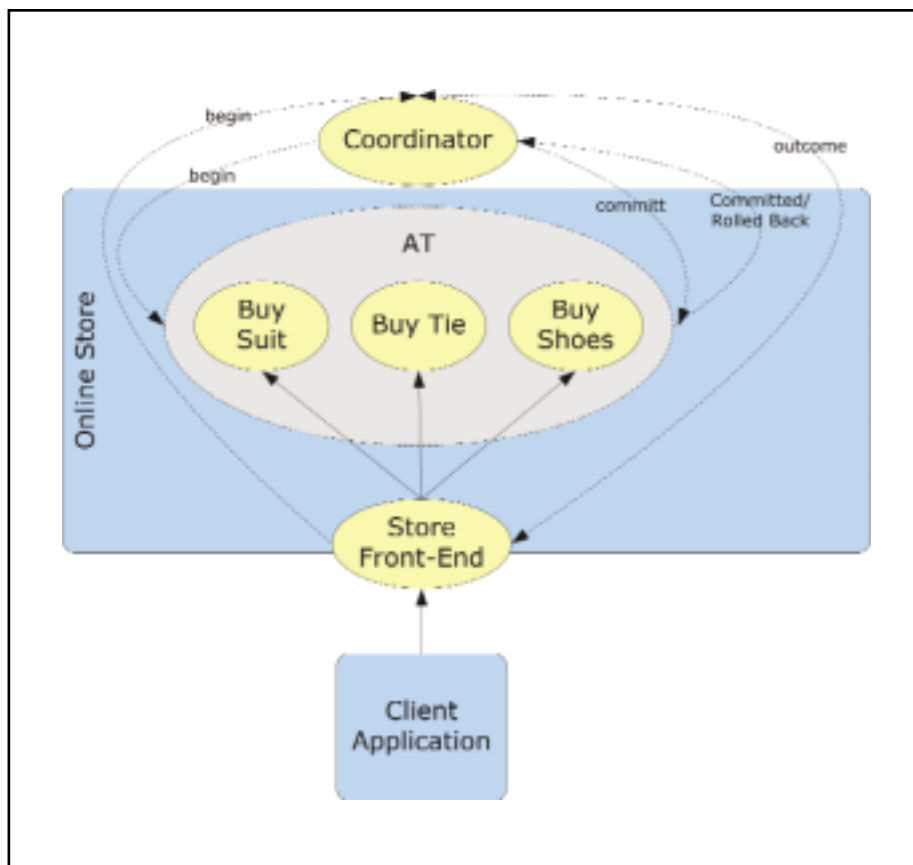


FIGURE 6 Using an AT to ensure all-or-nothing semantics for buying formalwear



Written by Joseph A. Mitchko

**Author Bio:**

Joe Mitchko is a technology specialist working for a leading Internet service company and is product review editor and contributing writer for Web Services Journal.

JMITCHKO@RCN.COM

BEA WebLogic Workshop 8.1

J2EE-Based Web services development made easy

Last year, BEA introduced WebLogic Workshop, a revolutionary product based on declarative annotations that took away most of the pain and aggravation of developing J2EE-based Web services on the WebLogic Application Server platform. Not being satisfied with just Web services, BEA extended this technology with Workshop 8.1 to include Web applications, portals, and other J2EE integration-based applications.

New Features

For development of loosely coupled applications that can maintain their public contract while underlying data structures change, WebLogic Workshop 8.1 now includes support for XML Schema and XQuery Mapping. Based on the XQuery XML standard, the visual mapping tool allows you to map XML elements to Java data elements by simply performing point-and-click operations. In addition to straight one-to-one mapping, you can also use a number of built-in XQuery functions such as "concat," allowing you to combine various fields into one. All of the hard work of handling the complex data transformations is performed automatically.

In addition to XQuery, Workshop 8.1 provides support for XMLBeans, a strongly typed Java object interface for XML data that allows a developer to manipulate raw XML data using the productivity and flexibility benefits of the Java language.

Web Services Improvements

For Web services, WebLogic Workshop 8.1 now supports both the RPC and document-literal style of SOAP requests, making it easy to integrate with .NET-based Web services. Here's where I couldn't resist testing this one out. Using the Order Entry Service example that comes with Workshop, I fired up a copy of Microsoft Visual Studio .NET on another machine and

within 5 minutes was able to create a .NET client for the Workshop example (using the generated WSDL) and successfully execute the service from a .NET ASP application. I need not say more.

Java Controls

WebLogic Workshop 8.1 includes a number of new Java Controls to help you connect to various IT assets, including FTP, e-mail, Tuxedo, Portal Server, Integration Server, and more. Remember that as a developer, interacting with a Java Control is the same for all types of back-end services. All you need to do is set various property settings and set up event handlers; the control itself handles all the hard stuff.

For all those proprietary legacy systems out there, Workshop 8.1 now enables you to create your own custom controls and also opens the door for custom third-party vendors and ISPs to develop Java Controls.

Java Page Flow View

One of the major new changes to Workshop is the introduction of a visual development interface for Java Page Flow (JPF) files. This approach, based on the Struts Model-View-Controller (MVC) architecture, allows you to visually see the flow of a Web application, including user action decision flow and business logic. Tag libraries and drag-and-drop wizards are included to help you bind information on each page to external data sources, including databases, Web services, and Java controls. Workshop automatically provides support for sessions and state management.

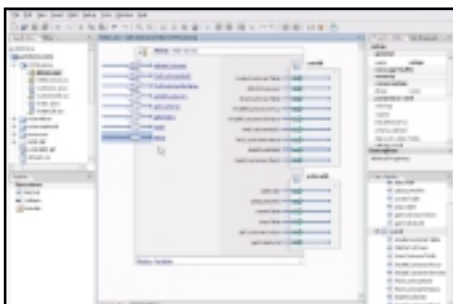


FIGURE 1 WebLogic Workshop 8.1 visual development environment

COMPANY INFO

BEA Systems, Inc.
2315 North First Street
San Jose, CA 95131
Tele: 1.800.817.4232
Web: www.bea.com
E-mail: sales@bea.com

DOWNLOAD INFORMATION

Free one-year development subscription at
<http://dev2dev.bea.com/subscriptions/index.jsp>

TESTING ENVIRONMENT

OS: Windows-XP
Hardware: 1GHz Athlon, 1G RAM



Workshop IDE

The WebLogic Workshop 8.1 IDE itself is well organized and provides you with various views, editors, property panels, etc., to assist you in your development work (see Figure 1). When using pre-built Java Controls to your back end, you can literally create a Web service as fast as you can drag and drop Java Controls and methods into the design view panel. The IDE includes framework extensions for both WebLogic Portal 8.1 and WebLogic Integration 8.1.

Conclusion

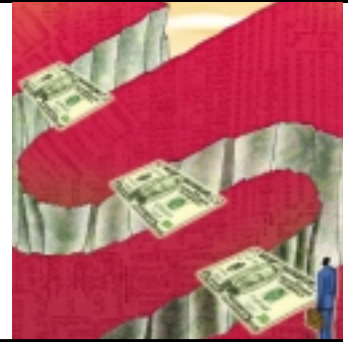
BEA WebLogic Workshop 8.1 is a powerful tool for developing sophisticated J2EE-based applications requiring integration with Web service-based assets both within the enterprise and abroad. Workshop 8.1 allows anyone with minimal Java coding skills to do some fairly complex J2EE development. Its power and ease of use take most of the drudgery out of J2EE development, and present a new level of competition for competing architectural platforms that make a similar claim – specifically Visual Studio .NET. ©

SpiritSoft

www.spiritsoft.com/climber

Intelligent Architectures for Service-Oriented Solutions

Moving toward a dramatic reduction in cost and time



Since the dawn of the software industry, technology has been evolving at a rapid rate to meet business needs. We have now entered a new stage in evolution with the adoption of Web services, which will only bring a quantum leap in productivity for businesses if a number of key characteristics are adopted that enable a revolution in software development.



Abdul Kayam is CTO of hyfinity, an intelligent Web services platform company (www.hyfinity.com). Before hyfinity, Abdul worked for AT&T and Software AG for 10 years. Abdul held the position of chief architect, responsible for the design and development of large-scale architectures for mission-critical systems employing Web and XML technologies.
ABDUL.KAYAM@HYFINITY.COM



Steve Bailey is chief e-business architect at hyfinity, an intelligent Web services platform company (www.hyfinity.com). Steve's 14 years in IT have focused on the delivery of dynamic self-service websites, systems, and B2B integration. He consults with enterprise customers developing real-world business solutions harnessing Web services.
STEVE.BAILEY@HYFINITY.COM

This article will define the essential characteristics for software development for Web services. None of the characteristics presented are new; in fact some of them date back more than 20 years. The fundamental difference is the "morphing" of these elements into a revolutionary software development approach for service-oriented solutions.

Software Development Building Blocks

If we examine software development over the past 40 years, software has become more complex as business needs and opportunities for exploiting technology have grown. As the complexity and cost of using software increases, the developer's ability to manage the complexity decreases, resulting in a dramatic reduction in productivity.

If we examine any software development approach, there are three primary building blocks:

- Communication Mechanism

TABLE 1: Software Development Approaches: The Primary Building Blocks

	General	Communication Mechanism	Business Logic Implementation	Information/Data Representation
1970s	Key processing orientation. Mainframe centric. Predominantly offline/overnight.	Proprietary protocols, such as IBM SNA.	Assembler, Cobol - Implemented as single monolithic processing steps	Data passed as binary, and stored into basic hierarchical indexed files.
1980s	Departmental orientation. Client/server. Online GUI interfaces - LANs.	Proprietary protocols by operating system/platform - Flavors of Unix.	Predominantly 4GLs - Many based on SQL database products - Oracle, Ingres, etc. client/server Tiers.	Data passed as binary, and stored into relational databases.
1990s	Enterprise orientation. Heterogeneous environments - Mainframes, Unix, NT. WANs. Internal integration. business hours. Browser/GUI based.	Low-level component models - CORBA, DCOM, RMI, EAI/Middleware	Mixture of OO-based languages - Java, VB, C, etc. Procedural code n-tier architectures.	Data passed as binary, and stored into relational databases.
New Century	Business network orientation. Any platform, any location. On demand - 24/7 browser based.	Internet (HTTP) and XML. Open Web Services specifications - SOAP, WSDL.	Autonomous software agents. Content-based, declarative Business Rules	XML Documents - Readable by humans and machines.

- Business Logic Implementation
- Information/Data Representation

Technology Adoption Life Cycles

If we review the key evolutions of software development, we can categorize them by using these building blocks. Table 1 describes each of the key software development approaches of the past 40 years.

We also see that each approach follows the classic technology adoption life cycle identified by Geoffrey Moore in *Crossing the Chasm: Innovators, Early Adopters, Early Majority, Late Majority and Laggards* (see Figure 1).

We will see the majority shift to the New Century approach as the previous approaches will be unable to deliver on the promise of Web services because the developer community will be unable to manage the complexity of more traditional approaches.

The software development approach for the new century has been identified by the IT industry as service oriented.

Service-Oriented Foundations

If we examine the requirements of today's Web services landscape, there are

a number of key characteristics that should be considered in developing effective service-oriented solutions.

We have associated these characteristics with the primary building blocks required by a software development approach:

- **Communication Mechanism:** Distributed peer-to-peer
- **Business Logic Implementation:** Software agents with business rules

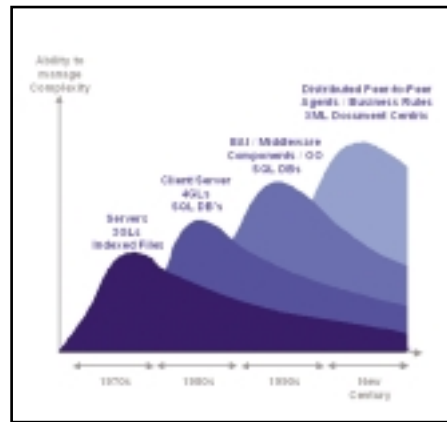


FIGURE 1 Software development approaches - adoption life cycles

- **Information/Data Representation:** XML document centric

These characteristics are essential for developing productive and agile Web service solutions that will have the longevity and adaptability that businesses require.

Communication Mechanism-Distributed Peer-to-Peer

In order for any two software elements to communicate, they must use common communication mechanisms. There are three essential elements: universal communication transport, dialogue language, and a unique identity system (see Figure 2).

Universal Communication Transport - The Internet

The Internet has revolutionized communication between people in the form of e-mail (SMTP protocol) and the publication of information (HTTP protocol). It is pervasive, can be reached from anywhere on the planet, and represents the ubiquitous communication network for businesses to interact over, whether internally or externally.

SHOP ONLINE AT **JDJSTORE.COM** FOR BEST PRICES OR CALL YOUR ORDER IN AT **1-888-303-5282**

BUY THOUSANDS
OF PRODUCTS AT
GUARANTEED
LOWEST PRICES!

GUARANTEED BEST PRICES
FOR ALL YOUR
WEB SERVICES
SOFTWARE NEEDS



N-ARY

\$299.00

n-ary Ticket System v2.0

This installable application is a clean-cut Web-based tool that enables you to log & track important information. It can be used as an intranet or extranet product, giving your customers access to see how their issue is progressing. A unique number ID is assigned to every ticket. The system is extremely flexible and simple to use and can be utilized in numerous different situations. With more features than before, The Ticket System can be used for any number of great applications, which you can tailor as much or as little as you like.

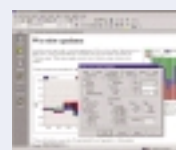


GREENPOINT INC.

\$1,350.00

WEB CHARTS 3D VER. 4.7

WebCharts 3D is a state-of-the-art visualization package designed for the professional Web developer. It provides general purpose and specialized 2- and 3-dimensional charts, grids, and heat maps that can be delivered as server-generated interactive images (PNG, GIF, JPG, SWF, SVG, WBMP) or applets to browsers and mobile devices.



CHUTNEY TECHNOLOGIES

\$755.25

Chutney SOAP+ Toolkit

The Chutney SOAP+ Toolkit is the industry's first Web services monitoring and optimization toolkit. The Toolkit fills the functionality gaps in the leading SOAP development libraries by providing the ability to accurately pinpoint Web services bottlenecks and eliminate them through optimization techniques such as caching. The Chutney SOAP+ Toolkit acts purely as a supplement to these libraries, so no changes to the existing application logic are required. Flexible in its feature set, the Toolkit provides value to applications serving as either Web service consumers or providers.



ALTOWEB

\$2,500.00

Application Platform Release 2.8

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.



JALERTS

\$1,495.00

JAlerts Deployment (ICQ)

If you need to send real-time Instant Messaging Alerts and Notifications to your employees or customers over the public IM services such as AIM, MSN, ICQ and Yahoo than JAlerts is your solution. Based on Zion Software's popular JBuddy SDK technology, JAlerts provides an even simpler solution for sending real-time messages.



INFRAGISTICS

\$795.00

JSuite 6.0 Basic

Lightweight components with heavyweight features, the JFCSuite v2 extends the standard Java Foundation Classes (JFC) to provide a series of feature-rich UI JavaBeans components.



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

OFFERS SUBJECT TO CHANGE WITHOUT NOTICE

Universal Dialogue Language – XML

The Internet accelerated with the use of HTML, a simple markup language to publish information for people to view. Unfortunately, machines cannot understand the data presented within HTML markup. In order for businesses to enable conversations over the Internet, they need a common language. XML allows data to be defined with a markup language that enables both humans and machines to understand the content of a page or document. It is already universally accepted as the syntax for conversations. Many industry bodies are defining standard business documents to be used for electronic business interactions.

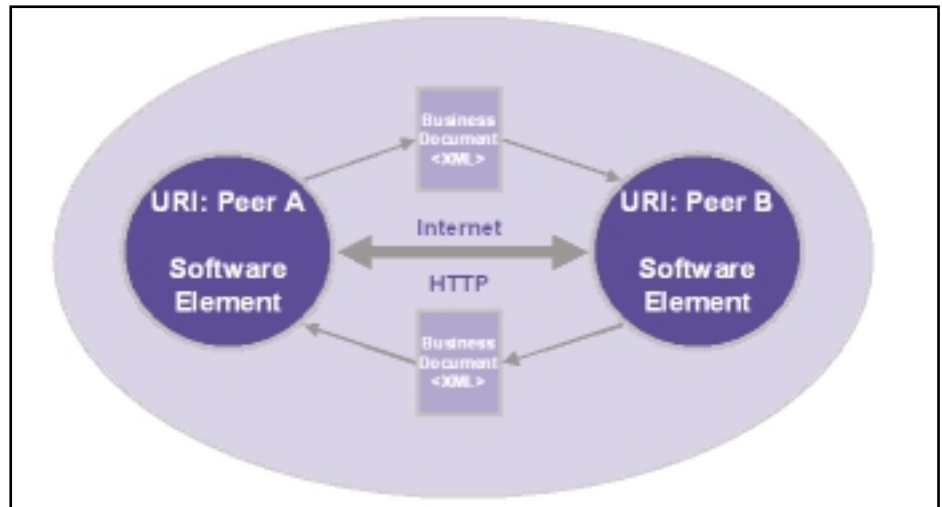


FIGURE 2 | Communication mechanisms - distributed peer-to-peer

Universal Identification – URI

In order to find information or services it is essential to have a globally unique identification mechanism. This ensures every Web page or document can be identified with a standard address mechanism.

Business Logic Implementation – Software Agents Using Business Rules

When two software elements communicate they are undertaking particular tasks on behalf of a business. Those tasks need to be implemented as a set of business services and processes. The services are the access points to a business, for example, to place a hotel reservation. The service would then call upon one or more business processes and consume or use resources to undertake the request. In the case of the hotel reservation, this may include checking the customer's status, the room availability, and recording the booking.

Each software element that delivers a service or process must be implemented with characteristics that ensure a loosely coupled, proven, and reliable implementation.

Contract of Use – Services-Oriented Architecture

If all business services are exposed through universal communication mechanisms, they can be consumed internally, externally by trusted organizations, or even publicly on the Web. Many of today's existing applications have not been constructed with this in mind.

Some traditional middleware technologies suggest that you simply need to wrap the existing applications with a service-oriented interface, and this will allow you to plug-and-play them into the universal communication network. It is often valid to wrap existing applications in this manner, but they have not been designed for external or public consumption. They tend to be internally focused, and have inappropriate data inputs/outputs, processing steps, exception handling, and security models.

Business services must be constructed by clearly defining the contract of the business processing that they expose, and the information that can be communicated via the exposed business service.

The Web Service Description Language (WSDL) standard allows the definition of service contracts, and is already accepted as a de-facto standard.

Autonomous Interactions – Role-Oriented Software Agents

The software elements need to be autonomous and loosely coupled software agents. A software agent is a piece of software that has the capacity to autonomously conduct its work. Each software agent exists for a clearly defined purpose, performing a specific role or job for a business. It is essential that the software agents themselves are not aware of the implementation details of other services. They should not know exactly where they are physically located.

The software agent would simply call upon other services when it required

them as part of its business processing. A particular service may be an exposed service of another business, or a business process consuming internal resources, e.g., recording a reservation in an operational data store. This is essential to ensure that implementation details are not entangled with the primary business logic.

Infrastructure Independent – Network Enabled Agents

Another key issue is that most software elements cannot easily be redistributed because they are based on centralized enterprise models rather than a networked, role-based model.

The current dominant development approaches are object-oriented and component-based construction models, whether based on J2EE or .NET platforms. Unfortunately, these types of software elements are constructed and deployed on the basis of an application server, with procedural code tightly coupled to the infrastructure.

These approaches operate on the basis of encapsulating fixed interfaces or methods to a software element with low-level technical details mixed with business information. This results in the data input and output to a software element being rigid and very tightly coupled to the physical implementation's architecture.

Software should be constructed with autonomous software agents that have no embedded knowledge of the physical world they exist in. The definition of the

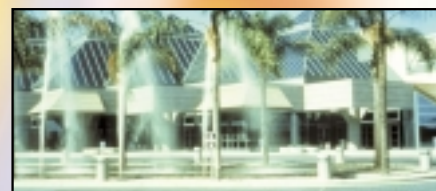
International Web Services Conference & Expo

Web Services Edge ^{WEST} 2003

web services
conference & expo 

SEPT. 30 - OCT. 2, 2003

Santa Clara, CA



**EXTENDING THE ENTERPRISE
WITH WEB SERVICES THROUGH JAVA,
.NET, WEBSHERE, MAC OS X
AND XML TECHNOLOGIES**



XML

WebSphere



Mac OS X



LINUX EDGE
conference & expo 

web services **EDGE**
conference & expo 

BOSTON

February 24-27, 2004

Featured technologies and topics will include:

- Focus on .NET
- Focus on Java
- Focus on WebSphere
- Focus on Mac OS X
- Focus on XML

For more information visit
www.sys-con.com

or call

201 802-3069



Over 100 participating companies will display and demonstrate over 300 developer products and solutions.

Over 2,000 Systems Integrators, System Architects, Developers, and Project Managers will attend the conference expo.

Over 60 of the latest sessions on training, certifications, seminars, case studies, and panel discussions will deliver REAL World benefits, the industry pulse and proven strategies.

Contact information: U.S. Events: 201 802-3069 or e-mail grisha@sys-con.com

WebServices
JOURNAL

JAVA
DEVELOPER'S JOURNAL

WebSphere
DEVELOPER'S JOURNAL

XML
JOURNAL

NET
DEVELOPER'S JOURNAL

SAMS

WebLogic
DEVELOPER'S JOURNAL

wireless
BUSINESS TECHNOLOGY

LINUX
BUSINESS TECHNOLOGY

CF Advisor

ColdFusion
DEVELOPER'S JOURNAL

PowerBuilder
DEVELOPER'S JOURNAL

Java is a registered trademark of Sun Microsystems, .NET is a registered trademark of Microsoft, Mac OS X is a registered trademark of Apple Computer, Inc., WebSphere is a registered trademark of IBM. All other product names herein are the properties of their respective companies.

**WEB SERVICES EDGE WEST 2003
CALL FOR PAPERS NOW OPEN**

Submit your papers online at:
www.sys-con.com/web-services2003west

PRODUCED BY
SYS-CON
EVENTS

physical communication mechanisms and location should be defined separately from the business processes. This allows software to be redeployed by redefining their locations to each other, independent of the business logic.

It should not be an issue if a software application needs to be deployed on a J2EE or .NET platform. The software should have no dependencies on its physical environment. The only open approach that can maintain this independence is to ensure that all the business logic is encapsulated within W3C XML architecture standards – URI, XML, XPath, XSL, and XSD.

This is critical if a business has a desire or need to be able to outsource particular business services and processes to third parties. A service-oriented solution does not require massive rewrites or integration activity to deliver this capability. It should simply be a case of redeploying the solution.

Separation of Concerns – Business Patterns of Interactions

The core principle behind the OO/Component development approach was to enable assembly of prefabricated components or objects. Unfortunately, this hasn't lived up to expectations because most businesses are different internally with respect to the infrastructure used – hardware, operating systems, languages, application servers, and so on. Some estimates suggest that only 5–10% reuse has really been achieved with business processes in these environments. Where reuse has been achieved, it is predominantly in areas that are horizontal rather than vertical in nature, such as security or auditing.

The fundamental problem is that the business processes and the physical infrastructure details are tightly coupled. It is more powerful to ensure that the business process definitions are completely independent of the physical implementation.

The business processes should be defined by selecting proven business patterns of interaction rather than focusing on technical interface requirements (see Figure 3). These patterns come in a number of forms, based on the type of business problems that they are de-

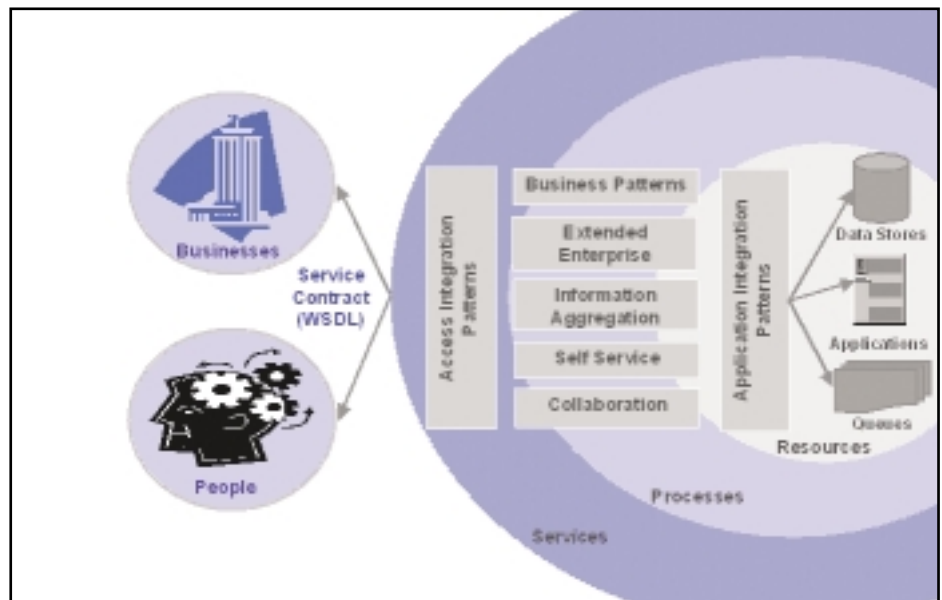


FIGURE 3 | Business logic implementation – patterns of software agents

signed to handle. IBM has undertaken an exhaustive study of 80,000 engagements and defined an extensive range of reusable, proven patterns for e-business solutions. The business patterns fall into four primary categories:

- **Self Service:** B2C multi-channel access
- **Collaboration:** User-to-user interactions
- **Extended Enterprise:** B2B integration
- **Information Aggregation:** Data integration

For example, an Information Aggregation business pattern defines how data should be acquired from another source. An example is a “Multi-Step Population” pattern, which uses a number of software agents to perform the following steps: extract, transform, and load data.

It is essential that software agents be defined based on reliable business patterns of interaction.

Separation of Logical and Physical Aspects

Many software development advances can be characterized as making the software code look more like the design specification.

It is essential to separate the business aspects of development from the implementation environment aspects. For example, the business aspect may be focused on the recording of a hotel reservation. The business requirement is to

store the reservation (business document) into an Operational Data Store for processing at a later time.

The implementation environment should be able to use an SQL database, XML data store, or a local file system without requiring a change in the business logic. The processing of this logic should be completely independent of the physical infrastructure.

For example, a generic SQL software engine would save, update, or get business documents from the particular type of operational store. The engines don't need to know anything about the specific business documents they are processing. The business logic could then call on the engine using a standard interface.

We then define transformations to convert the reservation business document dialogue to the application integration dialogue required by the operational data store. For example, the engine could transform the XML document into a SQL statement, and transform the results of the SQL query into an XML document.

Of course, if the operational data store is changed, it is only the transformation that needs to change, along with the selection of the relevant software engine. The business process and logic remain unaltered. This can significantly reduce development effort, and drastically reduce the cost of change, which is the most expensive element of a project life cycle.

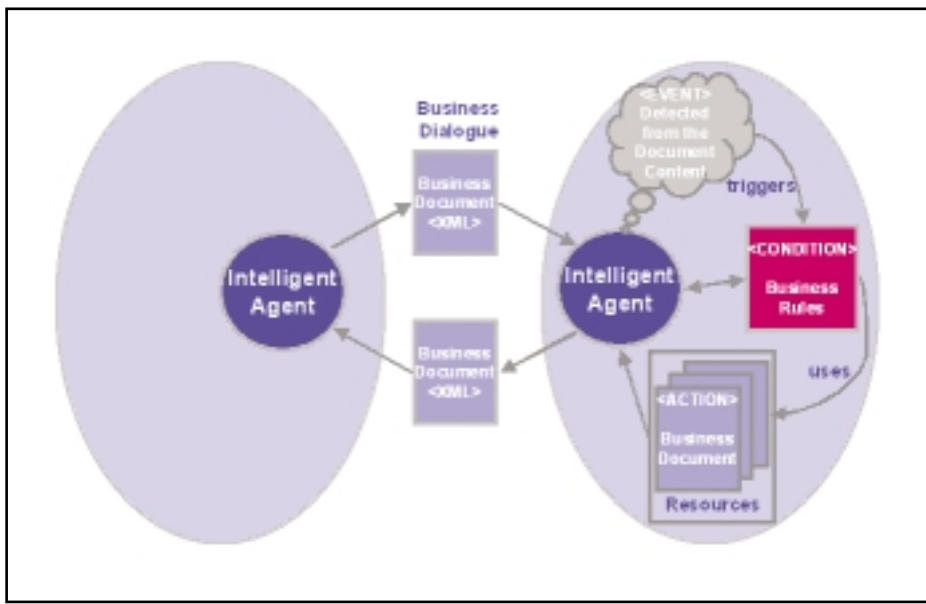


FIGURE 4 | Business Logic Implementation - Declarative Business Rules

Business Logic Visibility - Declarative Business Rules

Traditional software design and development methodologies have inherent weaknesses because knowledge leak-

age occurs when translating requirements and design specifications to software code. The primary reason for this is that most requirements for gathering and design methodologies are based on

declarative techniques, whereas software code is generally written in a procedural manner. Because there is no direct alignment between the business user's requirement definition and the software developer's implementation, it is very difficult to ensure accuracy and completeness.

The business logic should be defined with declarative business rules based on the content of the business document being processed. The rules should follow a simple structure that is intuitive for both business users and developers: <event>-<condition>-<action>. The rules are only fired if the <condition> is satisfied by the incoming <event> content of the business document. If the business rules change the data content, which is checked by other rules, then further rules will be fired. This is known as a forward-chaining business rules engine (see Figure 4).

This approach to business logic is inherently more maintainable as further declarative definitions can be added eas-

THE INSIDER
INTELLIGENCE
YOU NEED...

TO KEEP AHEAD OF THE CURVE

Go to www.SYS-CON.com

SELECT THE INDUSTRY NEWSLETTERS THAT MATCH YOUR NEEDS! CHOOSE ONE OR TRY THEM ALL!

- JAVA > Newsletter
- WebServices > Newsletter
- OXML JOURNAL > Newsletter
- wireless > Newsletter
- WebLogic > Newsletter
- WebSphere > Newsletter
- GoldFusion > Newsletter
- .NET Journal > Newsletter

FREE

E-Newsletters

SIGN UP

TODAY!

The most innovative products, new releases, interviews, industry developments, and plenty of solid i-technology news can be found in SYS-CON Media's Industry Newsletters. Targeted to meet your professional needs, each e-mail is informative, insightful, and to the point. They're free, and your subscription is just a mouse-click away at www.sys-con.com.

Exclusively from the World's Leading i-Technology Publisher

ily, particularly when compared to more complex nested procedural logic. It also allows a business user and a developer to work in collaboration by viewing the same business logic definition because it is not buried within the software code.

Information/Data Representation – XML Document Centric

Having considered the communication mechanisms and business logic implementation, we must finally examine the representation of data and information within applications.

Earlier, we established that Web services process XML messages based on XML standards. If most interactions are performed in this manner, it is simply logical that the business rules should be defined based on the XML business documents rather than on traditional software code method properties.

“

As business needs
change and grow,
this platform must
also be able to
evolve with them ”

Flexibility of the Dialogue – Dynamic Data-Driven Interfaces

Traditional OO/component development approaches require a mapping of the hierarchical content to methods and properties to process XML documents. There are many issues associated with this approach, including accurate mapping and marshalling of the data. The business logic is fixed internally within the methods of the OO classes based on the class properties.

The advent of Web services presents enormous challenges. If information is exchanged in the form of XML messages with multiple parties, it is increasingly likely that the information will take many forms. If the existing software develop-

ment approaches are utilized, there will be a proliferation of software classes and interfaces for each business document. If we consider how many different documents and versions each business process will require across an organization, it will simply become unmanageable. The software code will become impossible to maintain.

Software agents need the ability to receive multiple business documents, and then decide the appropriate processing that should be performed based on content within the business document.

Document Metaphor – Content Activated Business Rules

What becomes even more powerful is when the business rules can natively process the XML business documents and react to the content directly.

The W3C has defined the XPath language for querying and processing content within XML documents. If we combine this rules syntax with a “Document Metaphor” type of action operation to be applied to XML documents, it is a very intuitive way to define the business logic.

The processing actions should take the form of business document manipulation, for example, operations such as read, write, store, forward, and so on. In fact, business rules of this nature are similar to decisions people make when manually processing paper documents.

Conclusion

The three main considerations of any software development approach are the communication mechanism, the business logic, and the representation of information and data. The ubiquitous nature of the Internet and its related specifications make it the obvious choice for a communication mechanism. In order to make effective use of these standards (which include SOAP, WSDL, XML, and HTTP), they should be incorporated directly into the development process.

By using XML over HTTP as a standard communication mechanism, business can take advantage of peer-to-peer architecture. This peer-to-peer approach requires autonomous, intelligent software agents, and these agents will in turn need to be able to react to the content of

the incoming messages to be able to function efficiently.

The most effective way for the agents to do this would be to make use of XML-based rules using open standards such as XPath. By making these business rules declarative, we also gain business logic visibility. If we also ensure that the rules are directly activated by the incoming XML document content, we also gain a human- and machine-readable representation of information and data.

This article has examined what we believe to be the key characteristics required for a service-oriented development approach, and highlighted why this approach delivers a dramatic reduction of the cost and timescales for business integration. E-business projects should be driven by business needs and not by the coding capability of technical specialists. In fact, as more and more software elements are Web services-enabled, the cost of business integration will decrease further.

Traditional software development and middleware approaches adopted by the majority today simply cannot deliver effective, intelligent service-oriented architectures. A Web service-oriented architecture must be document-based, loosely coupled collaborations of autonomous intelligent software agents. A single development and runtime platform that supports all the above characteristics in a cohesive architecture is inherently more agile and productive. As business needs change and grow, this platform must also be able to evolve with them, making the software development approach outlined in this article the only suitable choice to take software development into the future.

References

- *W3C Architecture Domain: Enhancing the infrastructure of the Web and increasing its automation:* www.w3.org/Architecture
- “Patterns for e-Business – A Strategy for Reuse” (IBM Press www-106.ibm.com/developerworks/patterns)
- *OMG Model Driven Architecture:* www.omg.org/mda/
- *Aspect Oriented Development:* www.aosd.net ©

JavaOne

www.java.sun.com/javaone/sf

Unlock the Power of the Mainframe

Leveraging CICS as a Web services hub



Over many decades, IT organizations have invested billions of dollars in business logic and data housed in mainframes and within the CICS environment. Despite the age of these systems, the mainframe and CICS continue to provide the most scalable, reliable, and cost-effective platform upon which to conduct critical business operations.

AUTHOR BIO:



Paul Roth is CTO of CommerceQuest (www.commercequest.com). He is a leading advocate within the industry for open standards, e-business, and business process integration. Paul serves as the company's technology evangelist, speaking on the technical issues surrounding business integration across and inside enterprises.

PAUL.ROTH@COMMERCEQUEST.COM

Yet expanding the ROI of these systems by leveraging more modern technologies, such as Web services, continues to be a challenge.

Today's business environment requires an IT solution that will transform existing, rigid systems into flexible building blocks to enable a component-based, process-centric, and cross-platform integration environment. By adding an integration platform based on a service-oriented architecture, organizations can unlock critical mainframe resources and leverage data and business logic as services, allowing an organization to create a more flexible, dynamic, and agile infrastructure for competing in the 21st century.

This article presents the notion of CICS as an integration hub based on a component-based, service-oriented architecture supporting Web services. The challenges presented by most current approaches to implementing Web services to integrate with the mainframe are examined and contrasted with supporting Web services natively in CICS.

Among the technical issues discussed are:

- Transforming CICS resources into XML via an XML SQL metaphor
- Dynamically generating WSDL that effectively exposes Web services and subsequently accessing these services as SOAP-enabled Web services
- Building composite applications or busi-

ness processes that are composed of multiple CICS and non-CICS resources, and accessing the newly created business process as a Web service executing natively in CICS

Web Services Brings Ubiquity to Integration

The benefits of Web services have been touted for a few years now. Standards for communications (TCP/IP and HTTP), message exchange (SOAP), data formatting (XML), and

means of describing the access (WSDL) to services have emerged. Many vendor products support standards for creating Web services requestors and providers, and some tools can even generate code from WSDL for service requestors to access Web services.

The supporting runtime infrastructure and integration capabilities between Web services and legacy applications and data are the focus of this article. The standards don't deal with what is behind a Web service, nor should they. But behind the cur-

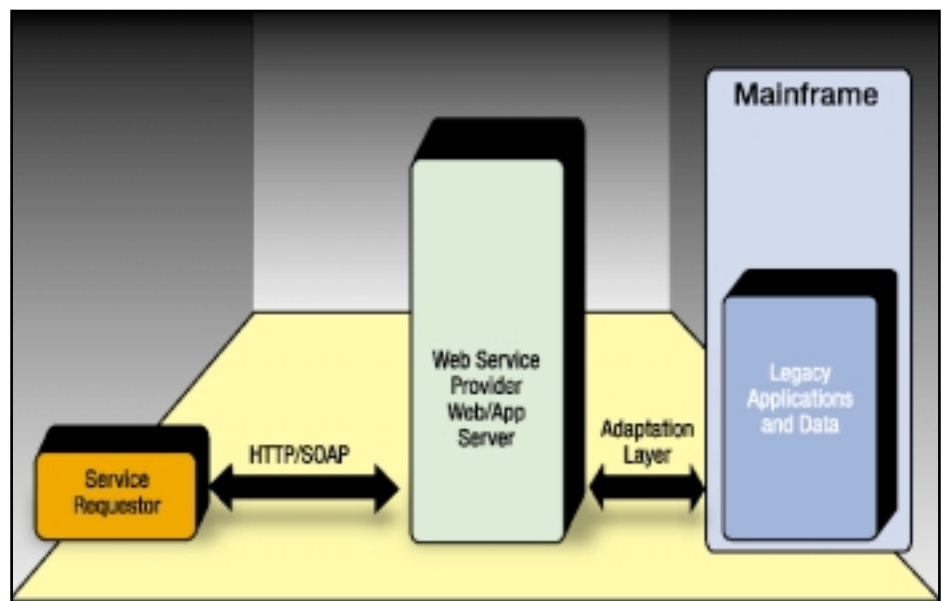


FIGURE 1 | Typical Web services/mainframe integration

tain lurks the dirty work that has to be done to make Web services real integration.

Service-Oriented Architecture: Bridging CICS with Web Services

Web services simply move the ability to create services into a standards-based approach that is independent of hardware/operating system platform and program language. Simply using Web services technologies to continue creating custom, point-to-point integration linkages produces just another tangled web of issues to address.

The real power of a service-oriented architecture is to think in terms beyond individual application-to-application linkages and more in terms of generic services that can be constructed for any and all applications to leverage. This is the key to effectively realizing the maximum benefit of Web services.

Typical Web Services Approach

Typical Web services products run on an outboard processor from the mainframe/CICS perspective, usually on a Unix or Windows server platform.

The ability to create Web services to wrap mainframe resources is limited by the ability of the chosen tool to integrate with the various types of resources found in CICS or accessible from CICS, including:

- COMMAREA programs
- VSAM files
- 3270 screens (BMS and others)
- DB2 tables and stored procedures
- IMS data and programs
- MQSeries-enabled applications

Not all vendor products provide the ability to access all of these resources and some are accessed using the same old tired EAI techniques of the past behind a facade of Web services.

There is still quite a bit of separation between the mainframe/CICS resources and the Web service provider platform. The issues in this type of an approach include:

- Security
- Transactional semantics
- Change management
- Availability and scalability
- Cultural differences

Security

Authentication and access control are critical to the security of CICS resources,

as well as for accounting purposes. What capabilities are offered to allow for users or programs to be authenticated to the back-end resources? HTTPS can be leveraged. Digital certificates can be crafted so that the identity of the Web service requestor will correspond to RACF userids, but the two security mechanisms must be integrated.

“

Today's business environment requires an IT solution that will transform existing, rigid systems into flexible building blocks ”

Transactional Semantics

CICS is a transaction manager. Many line-of-business systems depend upon transactional capabilities from order entry to order fulfillment and shipping to billing and so on. Virtually all financial institutions from banks to brokerage firms to insurance companies – depend upon CICS. ATM networks are controlled by CICS-based systems, whose reach encompasses the ATM terminal. It's important to understand the demarcation points of where transactions begin and end, and the impact on the end-to-end system. What happens if the communication link between the Web/application server or integration broker and the mainframe has a problem? How did the failure impact the Web service? Did the failure occur before initiating any CICS transactions or after? If the request got to CICS, did the CICS transaction complete? How is the Web service requestor or provider to know?

Change Management

All of the integration techniques of the past caused problems whenever some-

thing changed in the legacy application and data:

- Changes to 3270 screens broke any screen-scraping integration
- Printer output changes impacted applications tapping into printer drops
- Changes to data needed to be reflected in RPC client and server stub – requiring regeneration, recompilation, and redistribution to any client applications
- Changes to fixed message formats needed to be reflected in changes to code when applications were integrated using messaging and queuing

XML and Web services help in that additional XML tags and data can be added to a SOAP message body. Only those applications that need the new tags and data need to change.

Although changes can be isolated due to the loose coupling between Web service providers and requestors and the ability of Web service requestors to dynamically react to changes via WSDL, there is still an issue. An enormous problem common among the typical approaches to integrating the mainframe with Web services is managing changes to the underlying resources and the downstream impact to the Web services that have been created to wrap them. Because most approaches call for a Web/application server or integration broker running on a platform other than the mainframe, changes must be coordinated between the two platforms and typically between two organizations.

Availability and Scalability

Utilizing a separate platform to host a Web/application server and/or gateway requires two layers of systems that must be highly available and scalable and must also deal with potential communication failures between the two layers as well. This makes for a very complex operational situation. Clearly, tools and techniques exist to address the availability and scalability of Web servers that can also be applied to Web/application servers hosting Web services. And just as clearly, the mainframe/CICS environment also provides tools and technologies to address high availability and scalability of transactional systems. But they are two

SUBSCRIBE TODAY TO MULTIPLE

Go To WWW.sys-con.com/suboffer.cfm



and receive your
FREE CD Gift
Package VIA
Priority Mail



Each CD is an invaluable developer resource packed with important articles and useful source code!

Pick the CDs to go with your Multi-Pack order

► Pick one CD with your 3-Pack order

► Pick two CDs with your 6-Pack order

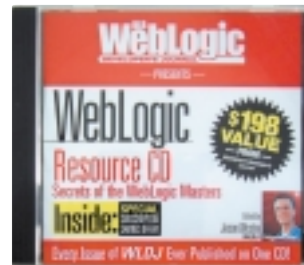
► Pick three CDs with your 9-Pack order



More than 1,400 Web services and Java articles on one CD! Edited by well-known editors-in-chief Sean Rhody and Alan Williamson, these articles are organized into more than 50 chapters on UDDI, distributed computing, e-business, applets, SOAP, and many other topics. Plus, editorials, interviews, tips and techniques!
LIST PRICE \$198



The most complete library of exclusive JDI articles compiled on one CD! Assembled by JDI Editor-in-Chief Alan Williamson, more than 1,400 exclusive articles are organized into over 50 chapters, including fundamentals, applets, advanced Java topics, Swing, security, wireless Java,... and much more!
LIST PRICE \$198



The most complete library of exclusive WLDJ articles ever assembled! More than 200 articles provide invaluable information on "everything WebLogic", including WebLogic Server, WebLogic Portal, WebLogic Platform, WebLogic Workshop, Web services, security, migration, integration, performance, training...
LIST PRICE \$198



The most complete library of exclusive CFDJ articles on one CD! This CD, edited by CFDJ Editor-in-Chief Robert Diamond, is organized into more than 30 chapters with more than 400 exclusive articles on CF applications, custom tags, database, e-commerce, Spectra, enterprise CF, error handling, WDDX... and more!
LIST PRICE \$198



The largest and most complete library of exclusive XML-Journal articles compiled on one CD! Edited by well-known Editors-in-Chief Ajit Sagar and John Evdemon, these articles are organized into more than 30 chapters containing more than 1,150 articles on Java & XML, XML & XSLT, <e-BizML>, data transition... and more!
LIST PRICE \$198



The most up-to-date collection of exclusive WSDJ articles! More than 200 articles offer insights into all areas of WebSphere, including Portal, components, integration, tools, hardware, management, sites, wireless programming, best practices, migration...
LIST PRICE \$198

- ☐ Web Services Resource CD
- ☐ Java Resource CD
- ☐ WebLogic Resource CD
- ☐ ColdFusion Resource CD
- ☐ XML Resource CD
- ☐ WebSphere Resource CD
- ☐ Linux Resource CD

Your order will be processed the same day!



An up-to-the-minute collection of exclusive Linux Business & Technology articles plus Maureen O'Gara's weekly LinuxGram, which keeps a finger on the pulse of the Linux business community. Edited by LBT's Editor-in-Chief Alan Williamson, these articles cover migration, hardware, certification, and the latest Linux-related products. Available June 2003!
LIST PRICE \$198

Subscribe Online Today

PEOPLE MAGAZINES ONLINE

AND SAVE UP TO \$400 AND RECEIVE UP TO 3 FREE CDs!



TO ORDER

• Choose the Multi-Pack you want to order by checking next to it below. • Check the number of years you want to order. • Indicate your location by checking either U.S., Canada/Mexico or International. • Then choose which magazines you want to include with your Multi-Pack order.

Pick a 3-Pack, a 6-Pack or a 9-Pack

<input type="checkbox"/> 3-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 6-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 9-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.

☐ Linux Business & Technology

U.S. - Two Years (24) Cover: \$143	You Pay: \$79.99 / Save: \$63 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 / Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 / Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 / Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 / Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$89.99 / Save: \$8

☐ Java Developer's Journal

U.S. - Two Years (24) Cover: \$144	You Pay: \$89 / Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 / Save: \$22
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 / Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 / Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 / Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$89.99 / Save: \$8

☐ Web Services Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 / Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 / Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 / Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 / Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 / Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 / Save: \$8

☐ .NET Developer's Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 / Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 / Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 / Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 / Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 / Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 / Save: \$8

☐ XML-Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 / Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 / Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 / Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 / Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 / Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 / Save: \$8

☐ WebLogic Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 / Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 / Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 / Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 / Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 / Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 / Save: \$1

☐ ColdFusion Developer's Journal

U.S. - Two Years (24) Cover: \$216	You Pay: \$129 / Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 / Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 / Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 / Save: \$20
Intl - Two Years (24) \$264	You Pay: \$189 / Save: \$75 + FREE \$198 CD
Intl - One Year (12) \$132	You Pay: \$129.99 / Save: \$2

☐ Wireless Business & Technology

U.S. - Two Years (24) Cover: \$144	You Pay: \$89 / Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 / Save: \$22
Can/Mex - Two Years (24) \$192	You Pay: \$139 / Save: \$53 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$79.99 / Save: \$16
Intl - Two Years (24) \$216	You Pay: \$170 / Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 / Save: \$8

☐ WebSphere Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 / Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 / Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 / Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 / Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 / Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 / Save: \$1

☐ PowerBuilder Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 / Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 / Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 / Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 / Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 / Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 / Save: \$1

RECEIVE
YOUR DIGITAL
EDITION
ACCESS CODE
INSTANTLY
WITH YOUR PAID
SUBSCRIPTIONS

3-Pack

Pick any 3 of our magazines and save up to \$275⁰⁰
Pay only \$175 for a 1 year subscription plus a FREE CD

- 2 Year - \$299.00
- Canada/Mexico - \$245.00
- International - \$315.00

6-Pack

Pick any 6 of our magazines and save up to \$350⁰⁰
Pay only \$395 for a 1 year subscription plus 2 FREE CDs

- 2 Year - \$669.00
- Canada/Mexico - \$555.00
- International - \$710.00

9-Pack

Pick 9 of our magazines and save up to \$400⁰⁰
Pay only \$495 for a 1 year subscription plus 3 FREE CDs

- 2 Year - \$839.00
- Canada/Mexico - \$695.00
- International - \$890.00

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

distinct environments and sets of tools/technologies solving the same basic problem in two places. These tools and technologies don't integrate with each other. This requires two skill sets on the part of operational personnel and complicates troubleshooting. Furthermore, the communications links between the two environments must also mesh well with the high availability/scalability solutions on either endpoint.

Web Services and CICS: Can These Cultures Coexist?

The predominant approaches to constructing Web services rely upon two fairly recent technologies – Sun Microsystems' Java 2 Platform Enterprise Edition (J2EE) and Microsoft's .NET. Virtually all vendor products require the use of a Windows or Unix-based server. Many require some form of a gateway and/or third-party adapters to interface to the mainframe.

The result is still two independent camps, much like in past EAI efforts, that require cooperation and communication. Worse yet is the ownership issue. Which-ever group "owns" the integration between the two environments is critical to the operations team. One of the greatest challenges to any EAI effort between the mainframe and other platforms is communication and cooperation among the different development and support organizations.

The Solution: CICS as the Web Services Hub

To overcome these issues, as well as provide for a more robust integration environment, CICS itself should host the Web services provider and integration layer. Furthermore, the task of creating and managing services should be performed by the same organization that maintains and manages the legacy applications and data being wrapped to ensure consistency and eliminate communications problems typically found in most integration scenarios.

This approach offers a single point of control for both the Web services and the CICS resources behind them. The Web service provider is, in effect, also the CICS resource provider. When problems arise, there is no issue over who owns the problem. And because there are no longer two distinct environments separated by a network, problem determination is simpler.

All sorts of operational issues are eliminated. Another layer of hardware/software to maintain and manage is removed, reducing operational costs. The same management tools currently being employed to support CICS can also be used to deal with the Web services also running in CICS. No retraining of operational staff is necessary.

The Web service provider, by virtue of the fact that it is under the control of CICS, immediately inherits all the benefits of CICS, including transactional

semantics, availability, and scalability. Security can be more tightly integrated with CICS from the Web service provider as well.

Putting the Pieces Together

CICS offers a feature known as CICS Web Support (CWS), which allows a Web browser to connect directly to CICS to invoke a CICS transaction using the EXEC CICS LINK call and passing data via the COMMAREA. EBCDIC/ASCII translations are automatically handled. And decode/encode functions may be plugged in to manipulate the data being passed between the Web interface and COMMAREA of the invoked transaction. All the hooks necessary to support Web services via HTTP exist in the CICS environment.

When combined with the capability to form Sysplex and CICSplex configurations, CICS Web Support provides a highly available and scalable platform for creating Web services second to none.

What Is Needed?

Additional capabilities are needed to enable CICS to operate as a Web services provider and integration hub:

- A mechanism for translating native data formats and record layouts from/to XML
- The ability to deal with SOAP messaging
- A repository to define services and mappings as meta-data
- The ability to dynamically generate WSDL for a given service based upon the meta-data
- The ability to access data and programs using a structured approach

Each item should be controlled under the auspices of CICS, easily understood by the mainframe/CICS Systems Programmer, and easily linked with CICS resources.

XML Translation

The ability to translate between XML and native data types (COBOL, DB2, etc.) and record formats (DB2 tables, VSAM files) is necessary. An XML parser is available for COBOL. However, there are a number of issues involved:

- Requires an upgrade to COBOL 3
- Only supports reading XML, not creating XML
- Efficiency concerns exist
- Cumbersome for COBOL programmers to work with

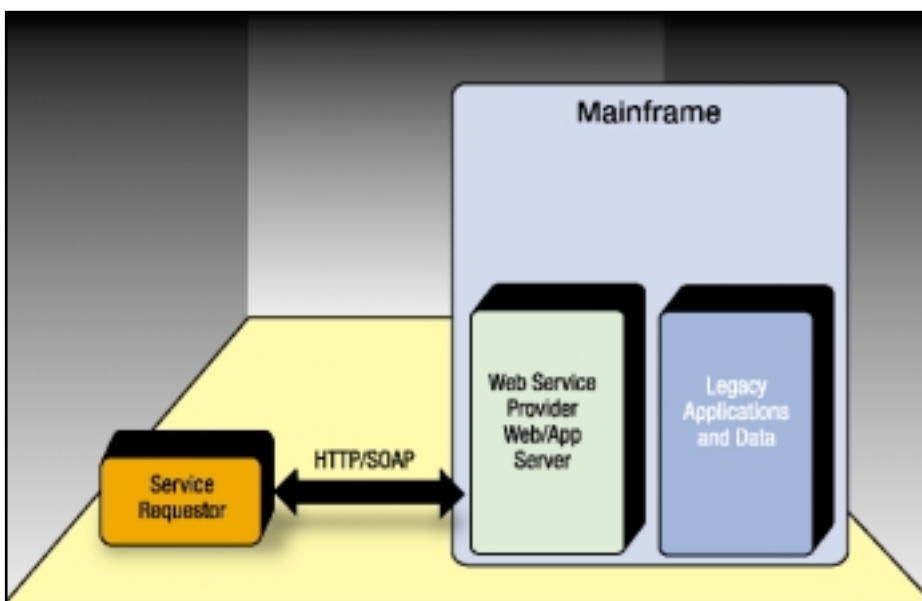


FIGURE 2 | CICS as the Web services hub

SOAP Messaging

Decoding/encoding of SOAP message formats for both inbound and reply messages requires the use of a SOAP processor.

Repository

A repository of metadata should be maintained to define Web services and their mappings to various CICS resources.

Dynamically Generate WSDL

With the Web services defined via metadata in a repository accessible via CICS, it's easy to configure the Web service to respond to inquiries and generate WSDL on the fly to specify available services and interfaces.

Accessing Programs and Data Using a Structured Approach

Simply providing a CICS-resident means of supporting Web services is necessary but not sufficient to allow IT organizations to realize the benefits of Web services/CICS integration.

For simple resources such as VSAM files and DB2 tables, one recommendation is to offer an SQL-like mechanism to

define the methods or operations that may be invoked on the service. The methods should include select(), insert(), update(), and delete(). Programs, including DB2 stored procedures, can be executed using the executeProcedure() method.

By defining a standard set of methods, it becomes easier to generate code to support the service level interfaces, especially when combined with the ability to map XML to/from native CICS structures.

Setting the Stage for Composite Services and Beyond

Once various CICS-accessible resources have been wrapped as Web services, they may also be invoked within CICS itself, thereby creating the opportunity for realizing composite services at the source. Furthermore, with the appropriate building blocks in CICS, business processes can be constructed and exposed as Web services as well.

Conclusion

This article explains the benefits and advantages of supporting Web services

directly inside CICS instead of the traditional approach most vendors have taken by providing this support on a middle tier. A number of vendors are moving in this strategic direction – one of which is IBM, the CICS provider. IBM's traditional approach to providing Web services access to CICS resources is via the WebSphere Application Server and the CICS Transaction Gateway. IBM recently announced a technology preview that supports SOAP enablement directly from CICS Transaction Server 2.2. While not a complete implementation of a Web services integration platform as prescribed in this article, it is a first step toward leveraging CICS as the hub for Web services and validates the approach taken in this article.

References

- Gisolfi, Dan. "Use Web services to integrate with OS390/CICS." *IBM eServer Developer Domain*: www-1.ibm.com/servers/esdd/articles/os390/index.html.
- SOAP for CICS: www-3.ibm.com/software/http/cics/soap ©

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.



WebSphereDeveloperJournal.com



Do You Have Access to the Internet?

Then Subscribe Online and Save \$31!

It's that easy.



Introductory Charter Subscription

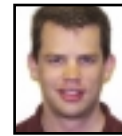
SUBSCRIBE NOW AND SAVE \$31.00 OFF THE ANNUAL NEWSSTAND RATE

ONLY \$149 FOR 1 YEAR (12 ISSUES) REGULAR RATE \$180

OFFER SUBJECT TO CHANGE WITHOUT NOTICE



Reviewed by Brian Barbash

**Author Bio:**

Brian R. Barbash is a consultant for the Consulting Group of Computer Sciences Corporation. He specializes in application architecture and development, business and technical analysis, and Web design.
BBARBASH@CSC.COM

WebFace & WebFace Studio by Vultus

A good alternative for Internet Explorer-based applications

As Web services proliferate and more and more applications expose their business logic in this manner, the need for effective user interfaces to consume these services grows. WebFace and WebFace Studio, from Vultus, provide a means of creating rich, Windows-like user interfaces distributed in a browser for Web services. The interface behaves like a traditional client/server application and does not require browser refreshes. The technology is built upon XML, HTML, and JavaScript, and requires that the user be running at least Internet Explorer 5.0. For this review, the focus will be on WebFace Studio and the development process.

Development in WebFace Studio


WebFace Studio is an IDE that provides a graphical environment in which to build applications using an event-driven model. The user interface itself, and each of its controls, comprise an XML document following Vultus' WebFace Application Markup Language (WAML) specification. At runtime, client-side WebFace JavaScript libraries interpret the XML document to paint the windows and controls for the user in the browser.

To start developing an application, the SOAP Application Builder wizard is executed. From an existing WSDL specification, the wizard generates a basic set of display, data, and communications components, along with supporting code to execute the service and display its results. As an example, I've created a sim-

ple Web service that allows you to manage your gym workouts. To demonstrate the wizard, I have requested it to generate the components necessary to retrieve a specific workout definition from the system. Figure 1 shows the results of the operation.


The parameters of the request include a workout ID and a user name and are shown by the two text boxes on the left side of the generated window. The result is an XML document that defines a specific workout and will be placed in the text box located in the right portion of the window.

Behind every GUI component or combination of components that display data is a DataSet object. These DataSets are part of WebFace's Model-View-Control (MVC) architecture and function similarly to the Java Swing GUI components and their models (i.e., JTable and DefaultTableModel). In the case of



COMPANY INFO
Vultus, Inc.
355 South 520 West, Suite 150
London, UT 84042

SALES
sales@vultus.com
801.852.0880



the workout ID and username text boxes, a DataSet with one row and two columns (one for each field) exists behind the scenes. This DataSet is the source of the data for the SOAP call, not the controls themselves. To establish a relationship between the DataSet and the controls, a DataController object is defined. This ensures that changes to the data in the

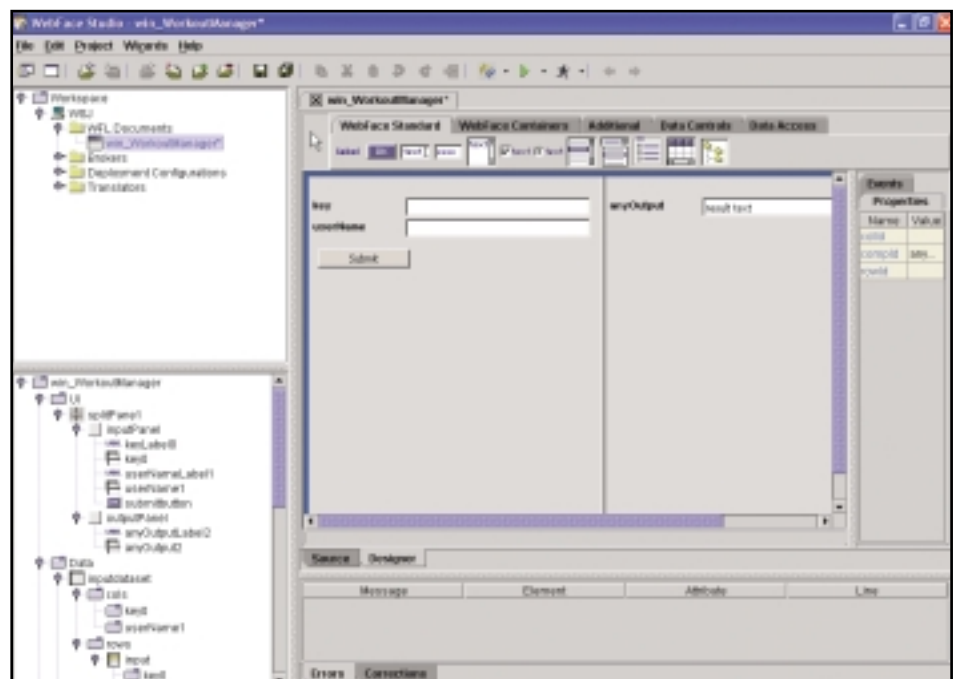


FIGURE 1 | SOAP Application Builder result

DataSet are reflected in the controls and vice versa.

The workhorse of WebFace applications is the Task object. Several types of tasks exist within WebFace:

- **Load Task:** Retrieves partial WAML documents from a host server. This may be used to request GUI components on demand.
- **Open Task:** Retrieves a complete window document from a host server.
- **Search Task:** Executed to retrieve data from a server, for example the result of a Web service.
- **Submit Task:** Submits data to the host server with no returned data.
- **Conduit Task:** Allows data to be transferred between DataSets.
- **Window Conduit Task:** Allows data to be transferred between DataSets across WebFace windows.

All tasks may be executed synchronously or asynchronously depending on the process requirement. A Task object contains the definitions of and source control references for the input parameters, the destination control references for its results, and the operations that it must perform.

In the workout example, the SOAP wizard created a Search Task to execute the appropriate Web service for retrieving the workout definition. When the submit button is clicked, the associated onClick event executes the Task to retrieve the data. The resulting XML document is displayed in the output textbox.

While tasks handle data associated with the request/response of service calls, a Request Broker object handles the details

of the communications process. Three broker types may be used in WebFace: Local, SOAP, and HTTP. The Local broker may only be used in conjunction with an Open Task to retrieve a window definition from the local machine. The remaining tasks may be used to execute remote services either via SOAP calls or HTTP calls.

In the workout example, the result of the Web service call is an XML document that is displayed in a text box. In order to work with the data, however, a more sophisticated view of the information, such as a grid, must be presented to the user. To accommodate a grid display, a DataSet must be defined to hold the information. Figure 2 shows the DataSet Editor dialog in WebFace Studio. This allows the developer to specify the columns and rows for the data grid. Constraints may be placed on the data, including data types, whether a value in a column is required, or its nullability. Once the DataSet is defined, a Data-Controller is created to link the DataSet to the display grid.

To populate the new DataSet, the information from the XML document produced by the Web service must be mapped appropriately. Recall that WebFace windows and their controls are themselves XML documents. Therefore, an XSL stylesheet may be used to transform the result of the Web service into the appropriate DataSet object (note that parameters for a request are also transformed via XSL stylesheets from their source DataSets into the service call). To assign a stylesheet to perform the transformation, the task is edited to include an xsltReplyTranslator that points to the desired XSL stylesheet. Figure 3 is a new window created in WebFace Studio that leverages the communications components and input fields created by the SOAP Wizard, but with the output transformed by an XSL stylesheet to fill a data grid.

As mentioned earlier, WebFace applications are event driven. Event handlers and all supporting code are written using JavaScript. In the WebFace Studio IDE, the developer may switch to the source view of the window to reveal the WAML document. From here, the CDATA ele-

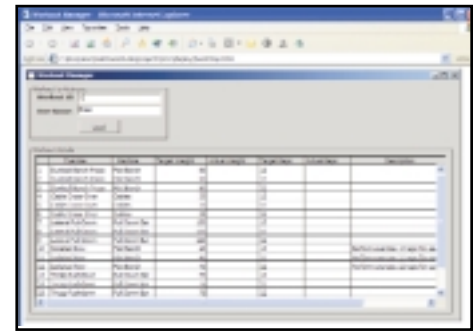


FIGURE 3 | Workout transformed to data grid

ment of the customCode tag may be edited to include any supporting JavaScript. All controls created are accessible as objects from the JavaScript code and may be manipulated or inspected at runtime. To wire an individual event handler to a control, the function name is assigned to the desired event in the properties editor for the control.

Deploying Applications

WebFace Studio provides a wizard to deploy applications for testing. The wizard may be used to deploy locally or remotely to a server. The local deployment simply copies the application files to a local directory on the developer's machine. Internet Explorer may be pointed to this directory using a file URL to launch the application. For server deployments, the destination machine must provide FTP write access. At deploy time, the application's configuration files are updated to reflect the runtime environment.

Summary

WebFace and WebFace Studio from Vultus provide the means to deliver rich, Windows-like user interfaces for Web services to a browser. By leveraging XML, HTML, and JavaScript they negate the need for browser refreshes when executing service calls and manipulating data. It does require some time to become comfortable with the development environment and the relationship between the various controls and components, but once acclimated, application development is relatively straightforward. For Internet Explorer-based applications, the WebFace product suite from Vultus is a good alternative. ©

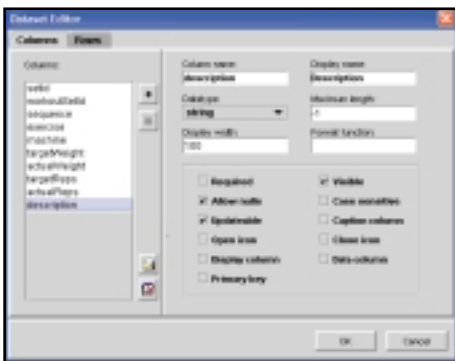


FIGURE 2 | DataSet creation dialog

Managing the Impact of Change in an Enterprise Web Services Network

A rapid-response system for change



A growing number of organizations are adopting a service-oriented architecture (SOA) approach to their IT infrastructure because they want to enhance their ability to change the enterprise application landscape cost effectively and rapidly, in support of changing business requirements. An SOA offers many potential benefits, but capturing the true value of an SOA is virtually impossible without the ability to manage the change inherent in Web service networks.

AUTHOR BIO:



James Phillips is chief strategist and senior vice president of product marketing and management at Actional Corporation. Actional offers a Web services management platform that is uniquely architected to help organizations manage the impact of the constant change inherent in enterprise Web service networks. Actional's patented active-management technologies provide users with a complete view of the entire enterprise Web services environment.

JAMESP@ACTIONAL.COM

The Nature of an Enterprise Service Network: Big, Dynamic and Rife with Interdependencies

The result of an SOA approach is a rapidly expanding network of enterprise Web services. SOA breaks monolithic applications into a greater number of smaller, more focused application modules that are easily understood, modified, and linked to form and reform applications as the needs of the business change.

By nature, this network of Web services is highly dynamic. Since facilitating change is easier, change occurs more frequently. The applications of acquired companies are integrated more rapidly. Customers, suppliers, and partners are connected faster.

Moreover, enterprise service networks are characterized by complex interdependencies. In the past, monolithic applications had very clear boundaries. In a service network, "applications" have fluid boundaries – they overlap and they are often assembled by leveraging components and services provided by other organizations within the enterprise, or even across enterprise boundaries. Dependencies can be direct, indirect, and circular.

The Impact of Change in an SOA

What do you get when you combine change, big networks of application components linked together in complex chains,

and interdependencies that are often unclear? You get exploding costs associated with keeping the network running and meeting performance expectations.

As a service network grows, the marginal cost of change in the network grows at an accelerating pace, fueled by both expected and unexpected changes.

Planned Change to a Service in the Network

In many cases, intentional IT actions may cause unintentional impact to the service network. These intentional actions can include introducing new applications, replacing and enhancing existing applications and Web services, upgrading and maintaining systems, and more.

An example of intended change with unintended consequences:

- **Change:** An aircraft manufacturer creates a new portal that will display real-time manufacturing process status. This portal leverages a Web service that returns the current status of a given aircraft assembly project. The Web service is changed to accept a more detailed query and to return a more detailed response.
- **Impact:** Systems bound to the old version of the service suddenly find their message formats invalid. Some systems that relied on the old version of the service were not

even known to the Web service owner, putting those systems at risk as well.

Unexpected Change to a Service in the Network

In a growing network of connected applications, service changes can occur suddenly and unexpectedly. For example:

- **Change:** A manufacturer faces a sudden increase in market demand for a product. Traffic increases to an e-commerce site as customers place orders. More calls come into the call center where customer service representatives take product orders. Both of these systems link to a customer management Web service being provided by a CRM system.
- **Impact:** There is an unexpected decrease in the response time of the customer management Web service.

This is just the beginning of a ripple effect of change throughout the service network, which will eventually impact systems that are several times removed from the initial problematic service. Specifically, any application that depends on the customer management Web service will now also begin to experience performance degradation.

Any upstream service that depends on the (now performance-degraded) customer management service may now itself display

Premiering
June 2003
at
JavaOne

www.sys-con.com



Millions of Linux Users One Magazine

Linux Business & Technology

There is no escaping the penetration of Linux into the corporate world. Traditional models are being turned on their head as the open-for-everyone Linux bandwagon rolls forward.

Linux is an operating system that is traditionally held in the highest esteem by the hardcore or geek developers of the world. With its roots firmly seeded in the open-source model, Linux is very much born from the "if it's broke, then fix it yourself" attitude.

Major corporations including IBM, Oracle, Sun, and Dell have all committed significant resources and money to ensure their strategy for the future involves Linux. Linux has arrived at the boardroom.

Yet until now, no title has existed that explicitly addresses this new hunger for information from the corporate arena. *Linux Business & Technology* is aimed squarely at providing this group with the knowledge and background that will allow them to make decisions to utilize the Linux operating system.

Look for all the strategic information required to better inform the community on how powerful an alternative Linux can be. *Linux Business & Technology* will not feature low-level code snippets but will focus instead on the higher logistical level, providing advice on hardware, to software, through to the recruiting of trained personnel required to successfully deploy a Linux-based solution. Each month will see a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

Regular features will include:

- Advice on Linux Infrastructure
- Detailed Software Reviews
- Migration Advice
- Hardware Advice
- CEO Guest Editorials
- Recruiting/Certification Advice
- Latest News That Matters
- Case Studies

LINUX EDGE
conference & expo

June 3-5 LONDON
June 24-26 BERLIN
September HONG KONG
October CALIFORNIA

**SYS-CON
MEDIA**

The World's Leading IT-Technology Publisher

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE \$71.76

YOU PAY ONLY

\$49.99

12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE TODAY!

WWW.SYS-CON.COM

OR CALL

1-888-303-5282

FOR ADVERTISING INFORMATION:

CALL 201.802.3020 OR

VISIT WWW.SYS-CON.COM

ALL BRAND AND PRODUCT NAMES USED ON THIS PAGE ARE TRADE NAMES, SERVICE MARKS, OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES.

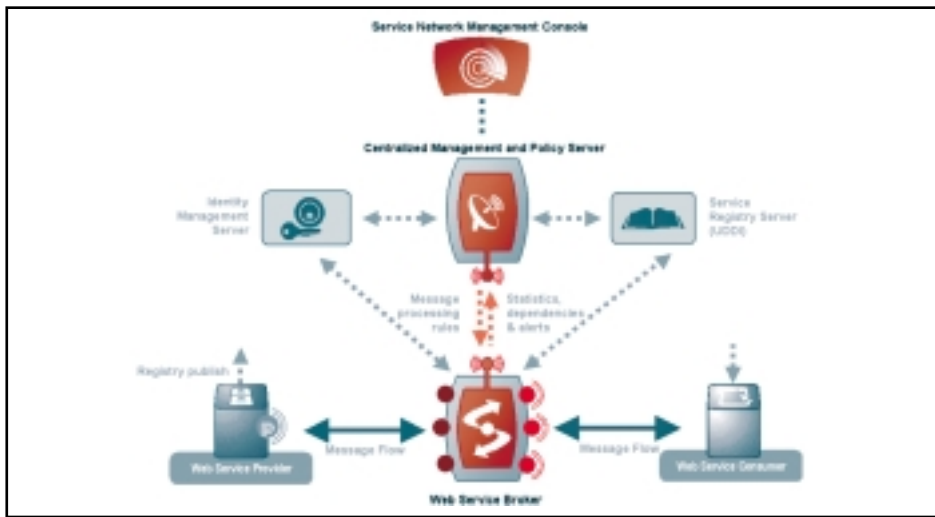


FIGURE 1 Service brokers (or an agent on a Web service provider) sit in the message flow between Web service providers, monitoring activity and enforcing centrally managed policy.

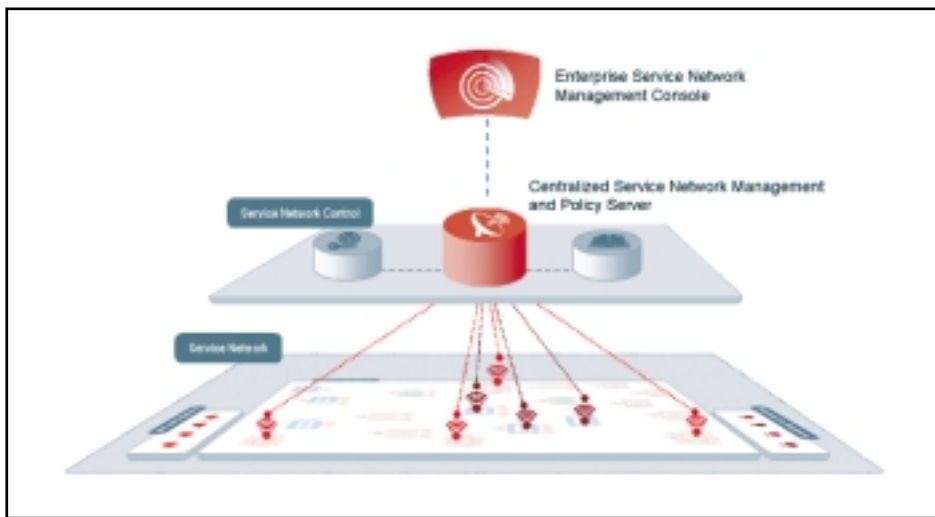


FIGURE 2 The effective Web Services Management Platform combines centralized visibility and policy management with distributed active control of the enterprise service network.

performance degradation in response time. Ripple effects continue until there are no more upstream systems acting as service providers – much like a rolling brownout effect in a power grid. The result is an overall decrease in service levels for all dependent Web services within the network.

While this series of events paints an ugly picture, what IT experiences is even uglier. There are no lines painted on the floor of the data center highlighting the interdependencies among the loosely coupled systems in a service network. There are no performance gauges and no alarm bells that sound when service performance moves outside normal bounds. There is no means of tracing failures back to the root of a service network problem.

Instead, those responsible for keeping the service network up and running face a

bewildering set of seemingly coincidental system failures. These failures are expensive as orders are lost, customers turn to competitors, and valuable IT personnel are frantically troubleshooting problems.

In order to truly reap the benefits of the service-oriented approach to application architecture, organizations must have a way to manage the impact of change in their enterprise service network.

Web Services Management Platform Requirements

A Web services management platform is critical to the continued profitable growth of a service network and must address both planned and unexpected change in an enterprise service network, providing a facility to:

- Understand the makeup of the service network

- Know when there is a service network problem
- Identify the root cause of the problem
- Insulate downstream applications from the root failure
- Predict the impact of a change
- Distribute policy, rules, and behaviors into the network of Web services to alleviate or avoid unintended consequences of change

Figure 1 provides an architectural overview of a solution that meets the requirements highlighted above.

This architecture combines the power of centralized visibility and policy management with distributed monitoring and policy enforcement. In this approach, active “in-network” components reside within the enterprise service network, logically positioned between the providers and consumers of Web services. A centralized management and policy server, working in concert with service registries and identity management solutions, communicates with these in-network components, both receiving and sending information. Information received from in-network components includes service performance statistics, alerts, and service interdependency information. Policy, in the form of rule sets, is sent to the in-network components to define in-network component behavior – when to raise alerts, how to process transiting message traffic, where to route messages, how to enforce security policy.

Successive deployment of in-network capabilities for each new Web service project results in a fabric of control woven into the enterprise service network. The central policy and management server taps into this in-network fabric of control (see Figure 2).

This solution provides a complete answer to the challenges faced in dealing with change in an enterprise service network – both unexpected and planned.

Those enterprises that deploy a Web services management platform in support of their adoption of a service-oriented enterprise software architecture will get what they expect – movement toward operation as a real-time enterprise and an IT organization able to rapidly respond to, and even enable, changes in the operation of the business. ©

WebServices

.NET J2EE XML JOURNAL

LEARN WEB SERVICES. GET A NEW JOB !

SUBSCRIBE TODAY TO THE WORLD'S LEADING WEB SERVICES RESOURCE

The Best
.NET
Coverage
Guaranteed!

Get Up to Speed with the Fourth Wave in Software Development

- **Real-World Web Services:** XML's Killer App!
- How to Use **SOAP** in the Enterprise
- Demystifying **ebXML** for success
- **Authentication, Authorization, and Auditing**
- **BPM** - Business Process Management
- Latest Information on **Evolving Standards**
- Vital technology **insights** from the nation's leading Technologists
- Industry **Case Studies** and **Success Stories**
- Making the Most of **.NET**
- **Web Services Security**

**Only \$69.99 for
1 year (12 issues)***
* Newsstand price \$82.00

*** Newsstand price \$83.88 for 1 year**

Subscribe online at
www.wsj2.com or
call 888 303-5252
**Offer subject to change without notice*

**Offer subject to change without notice*

**Offer subject to change without notice*



DreamFactory Software Accelerates Development of Interfaces for SOA

(Los Gatos, CA) – DreamFactory Software has announced the first full commercial release of an integrated tool suite designed to speed the development of sophisticated portals and Web applications built with native XML documents and powered by pure Web services transactions. The technology features standards-based integration with J2EE and .NET servers and enables desktop deployment of portals built with native XML documents and powered by pure Web services transactions.

www.dreamfactory.com



Liberty Alliance Releases Specs

(San Francisco) – The Liberty Alliance, a global consortium formed to develop open standards for federated network identity, has unveiled drafts of its Phase 2 specifications – an important step in creating a commonly accepted and more trusted way of building and managing identity-based Web services.

The Phase 2 draft specifications and related Privacy and Security documents are available at www.projectliberty.org. The Liberty Alliance expects to incorporate comments and finalize the specifications in Q3 2003.

VeriSign Announces Integrated Solution for Securing Web Services

(San Francisco) – VeriSign, Inc., has announced the VeriSign Trust Gateway, a solution to help companies simplify the process of deploying trusted Web services and managing the enterprise application integration and security. It simplifies application security by eliminating the need for developers to write security code into every application. Instead, it performs XML security operations on behalf of the applications,



based on policies set by enterprise administrators.

www.verisign.com/products/trustgateway/index.html

Global Computer Enterprises Wins E-Government Contract

(Gaithersburg, MD) – Global Computer Enterprises (GCE) has been awarded a contract for the development, implementation, and operation of the Federal Procurement Data System – Next Generation (FPDS-NG) from the General Services Administration (GSA). The current system will be reengineered to take advantage of new technology that allows millions of transactions to be recorded and reported upon in real-time, while providing tremendous cost savings to the federal government. FPDS-NG is the first system in the federal government designed from the ground up with large scale integration capabilities – it will integrate with every government procurement system in real time.

www.gce2000.com

Kavi Software Powers Expanded Web Portal for OASIS

(Portland, OR) – Kavi Corporation has announced the launch of an expanded Web portal for OASIS (www.oasis-open.org). Kavi's applications manage OASIS's membership and support its technical committee process for developing standards. Members can share documents and proposals, build consensus, and vote on specific proposals in conformance with OASIS bylaws and formal processes.

www.kavi.com

New IBM Software Lowers Costs While Increasing Performance

(Somers, NY) – The new IBM Server Allocation for WebSphere Application Server will help companies get more value from their existing IT resources by allowing them to automatically manage multiple applications running on multiple clusters of servers as a single environment. IBM WebSphere becomes the first application server to offer this grid capability built in.

By virtualizing the resources available across an entire grid of WebSphere servers, the new technology allows customers to significantly and simultaneously increase application performance and resource utilization.

www.ibm.com



Borland Closes Gap Between .NET and J2EE

(Scotts Valley, CA) – Borland Software Corporation has launched Janeva, a fast, flexible software solution



for enterprises integrating Microsoft .NET Framework

applications with J2EE and CORBA-based back-office software systems. Based on J2EE, Janeva supports all Microsoft .NET Framework-based languages, including C#, J#, Delphi, and Microsoft Visual Studio .NET, enabling greater team productivity to accelerate the application life cycle from development to deployment.

www.borland.com

www.microsoft.com



Q-Link Version 5.0 Allows Visual Assembly of Enterprise Applications

(Los Angeles) – Q-Link Technologies has announced version 5.0 of their application development platform, designed to reduce the overall



complexity, time, and cost of developing

J2EE applications. The Q-Link platform eliminates the complexity of traditional J2EE development by bringing together a new application model, a suite of integrated business process management services, and a scalable architecture built on standards-based technologies such as J2EE, XForms, and XPath. A primary differentiator of the Q-Link platform is the combination of component assembly and design-time integration capabilities.

www.qlinktech.com

The World's Leading Java Resource!

JAVA DEVELOPER'S JOURNAL

Here's what you'll find
in every issue of *JDJ*:

- Industry insights
- The latest software trends
- Technical expertise
- Career opportunities
- In-depth articles on Java technologies

Subscribe Today &
SAVE
30% Off
the annual cover price

ANNUAL COVER PRICE	
\$71.88	YOU PAY
\$49.99	YOU SAVE
30%	Off the annual cover price

Sign up **ONLINE** at
www.javadevelopersjournal.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

WSJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altova	http://xmlj.altova.com/authentic5		23
Ektron	www.ektron.com/ws		59
Global Knowledge	www.globalknowledge.com	1-800-COURSES	31
IBM	www.ibm.com/websphere/seit		2
Java Developer's Journal	www.javadevelopersjournal.com	888-303-5282	57
Java One	java.sun.com/javaone/sf		43
JDJ Store	www.jdjstore.com	888-303-5282	37
Linux Business & Technology	www.sys-con.com	201-802-3020	53
Macromedia	www.macromedia.com/go/cfmxad		17
Mind Reef	www.mindreef.com	603-465-2204	5
Mindreef News	www.mindreef.com/hl0306		4
Parasoft	www.parasoft.com/ws2	888-305-0041	7
Quest Software	http://java.quest.com/jcsc/ws	800-663-4723	9
Quest Software	www.java.quest.com/jcsr/ws	800-663-4723	27
Spirit Soft	www.spiritsoft.com/climber		35
SYS-CON Media	www.sys-con.com/suboffer.cfm	888-303-5282	46, 47
SYS-CON Media	www.sys-con.com	888-303-5282	41
Web Services Edge West 2003	www.sys-con.com	201-802-3069	39
Web Services Journal	www.ws2.com	888-303-5282	55
Web Sphere Developer's Journal	www.sys-con.com	888-303-5282	49

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. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

IN THE NEXT ISSUE OF *WSJ*...

Focus: BPM, Orchestration

"Smart" BPM – The Who, What, Where and When for Web Services

"Smart" BPM, Web services and portals are colliding to create an opportunity for companies to take common Web services to the next level of intelligent automation.



BPM to the Rescue

BPM relies on Web services to enable new processes that respond quickly to key business events as well as tap little-used business process functionality trapped within existing applications.



Web Services Orchestration

Web services orchestration and choreography standards are efforts that can be long-term solutions for business connectivity.



BPEL4WS: Make Your Services Flow

The impact of BPEL is significant, transforming application integration from a risky and expensive endeavor into a mainstream development practice based on Web services standards.



Business Process Automation

In the ongoing search to improve IT productivity, many companies are looking at BPM as a means to increase productivity and extend system functionality



Plus:

Why Web Services Work

The next column on "Web Services in the Real World" explores the reasons project executives said they chose Web services.



Web Services
JOURNAL





John Worrall & Jason Rouault

John Worrall is with RSA Security.
Jason Rouault is with Hewlett-Packard. Both companies are founding members of the Liberty Alliance Project.
WWW.PROJECTLIBERTY.ORG

Federated Identity Management Addresses E-Business Challenges

E-business initiatives – such as enterprise, B2B, and B2C applications – typically reach throughout and beyond an enterprise, requiring users to move across networks, applications, and security domains. To be effective, this movement must be transparent to the user. Consider what's involved in this: a single identity with one registration process and one login procedure. Not easy, considering hundreds, thousands or even millions of users require access to a growing number of applications which may or may not be under the direct control of the enterprise.

A single organization cannot effectively manage or control an e-business initiative from beginning to end, especially when multiple partners are involved. Even within the enterprise, different business units often manage distinct sets of users and



resources. That's why organizations are turning to federated identity management to address their e-business challenges.

duces a Web services-based identity service infrastructure that enables users to manage the sharing of their personal information across identity and service providers as well as the use of personalized services. For example, a user may authorize a service provider to access their shipping address while processing a transaction.

Built on top of the ID-WSF is a collection of interoperable identity services, the Identity Services Interface Specifications (ID-SIS). The ID-SIS might include services such as registration, contact book, calendar, geo-location, presence, or alerts. Through Liberty protocols and a standard set of attribute fields and expected values, organizations will have a common language to speak to each other and offer interoperable services. The services defined in the ID-SIS are designed to be built on top of Web services standards, meaning they are accessible via SOAP over HTTP calls, defined by WSDL descriptions, and use agreed-upon schemas.

So what does this mean to businesses? Let's look at a Web services-enabled supply chain environment in which the order management system automatically triggers back-end transactions that span inventory control and accounts payable. A trusted partner of a large electronics manufacturer accesses the ordering system through a designated portal. To access this application, the user's digital credentials and role are shared in a federated fashion, allowing the user to place an order without entering login information that is unique to the partner's domain. The user's order – and role-based authority – triggers a query to the inventory management system to determine whether inventory levels are sufficient to fill the order. Based on the inputs provided by the requesting application, the inventory management system earmarks product for this order and notifies shipping to process it. The accounting system registers the order and starts the invoicing process.

Thanks to Web services, most of this is done without human involvement. But what would happen without a federated user identity? There is no telling the extent of havoc that would result if an unauthorized or "untrusted" user had access to this process. Federated identities and Web services together are primed to take e-business processes to the next level.

The Liberty Alliance unites more than 160 firms representing more than 1 billion consumers. Organizations like this will continue to strive to achieve digital identity standards that will facilitate e-business processes around the globe. ©

resources. That's why organizations are turning to federated identity management to address their e-business challenges.

In a federated environment, a user logs on through his identity provider and then leverages that authentication to easily access resources in external domains. Federated identity standards form an *abstraction layer* over local identity and security environments of diverse domains. This abstraction layer provides for interoperability between disparate security systems inside and across domains, enabling true federation. Each domain maps to the agreed-upon policies without divulging sensitive user information. This trust is the foundation of any federated environment, and the organizations that work together within a domain are a *circle of trust*. A circle of trust connotes that both a business relationship and technical infrastructure are in place to assure secure access.

The Liberty Alliance is developing and delivering the first open architecture and specifications to enable federated identity management. At its core is the Identity Federation Framework (ID-FF), which facilitates identity federation and management through features such as identity/account linkage, single sign-on, and session management. ID-FF is fundamental to underpinning accountability in business relationships and Web services; providing customization to user experience; protecting privacy; and allowing adherence to regulatory controls.

The Liberty Alliance is also specifying an Identity Web Services Framework (ID-WSF) that will utilize the ID-FF. This framework intro-

Ektron

www.ektron.com/ws

Swingtide

www.swingtide.com/testdrive