

WebServices

.NET J2EE XML JOURNAL

WSJ2.COM

NOW SHIPPING



SEE DETAILS ON PAGE 49

From the Editor Tools of the Trade

by Sean Rhody pg. 3

Guest Editorial Enabling Trusted Web Services

by Phillip Hallam-Baker pg. 5

Product Review Parasoft SOAPtest

Reviewed by Prasad Joshi pg. 18

First Look AltioLive 3.0

by Joe Mitchko pg. 24

.NET

Eclipsing .NET: A Free .NET IDE

by Kyle Gabhart pg. 26

Web Services Adoption Navigating Web Services

by Bernhard Borges pg. 36

Industry Commentary WS as the Catalyst for an IT Economic Bounce Back?

by Tyler McDaniel pg. 66

RETAILERS PLEASE DISPLAY
UNTIL NOVEMBER 30, 2002

\$6.99US \$7.99CAN



SYS-CON
MEDIA



FOCUS ON TOOLS AND AUTOMATION

Web Services Orchestration

The marriage of Web services and workflow



Ron Ben-Natan &
Doron Sherman **30**

The Tools Landscape

Web services technology matures

Neil O'Toole
38

FOCUS ON SECURITY

Secure Web Services

Using SSL as an alternative to VPNs



Jeff Browning
22

Web Services Security

The key is integrating existing and planned technology

Lakshmi Hanspal
42

XML: Designing Web Services with XML

Signatures A little extra effort maintains data integrity



Mark Young
10

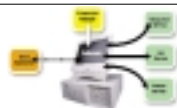
WSJ Feature: Monitoring & Performance

Management Optimizing design & runtime performance

Gunjan Samtani &
Dimple Sadhwani **30**

WSJ Feature: Web Services Infrastructure

Building transactional Web services with OASIS BTP



Jim Webber
50

BTP: Business Transaction Protocol: Transactions

for a New Age Bringing ACID models into the Web services world

Mark Little
56

IBM
www.ibm.com/websphere/integrate

WebServices JOURNAL

INTERNATIONAL ADVISORY BOARD

Jeremy Allaire, Andrew Astor, Steve Benfield, Graham Glass,
Tyler Jewell, Paul Lipton, Norbert Mikula, Frank Moss,
George Paolini, James Phillips, Simon Phipps

TECHNICAL ADVISORY BOARD

JP Morgenthal, Andy Roberts, Ajit Sagar,
Simeon Simeonov, Richard Soley

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

EXECUTIVE EDITOR

Gail Schultz gail@sys-con.com

MANAGING EDITOR

Cheryl Van Sise cheryl@sys-con.com

EDITORS

M'lou Pinkham mpinkham@sys-con.com

Nancy Valentine nancy@sys-con.com

ASSOCIATE EDITORS

Jamie Matusow jamie@sys-con.com

Jean Cassidy jean@sys-con.com

ASSISTANT EDITOR

Jennifer Stille jennifer@sys-con.com

PRODUCTION

VP, PRODUCTION & DESIGN

Jim Morgan jim@sys-con.com

LEAD DESIGNER

Richard Silverberg richards@sys-con.com

ART DIRECTOR

Alex Bolero alex@sys-con.com

ASSOCIATE ART DIRECTORS

Cathryn Burak cathyb@sys-con.com

Louis Cuffari louis@sys-con.com

Aarathi Venkataraman aarathi@sys-con.com

ASSISTANT ART DIRECTOR

Tami Beatty tami@sys-con.com

CONTRIBUTORS TO THIS ISSUE

Ron Ben-Natan, Bernhard Borges, Jeff Browning,
David Cameron, Kyle Gabhart, Phillip Hallam-Baker,
Lakshmi Hanspal, Prasad Joshi, Doron Sherman,
Mark Little, Tyler McDaniel, Joe Mitchko, Neil O'Toole,
Sean Rhody, Dimple Sadhwani, Gunjan Samtani,
Jim Webber, Mark Young

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 © 2002 by SYS-CON Publications, Inc. All rights reserved.
No part of this publication may be reproduced or transmitted in any
form or by any means, electronic or mechanical, including photocopy
or any information storage and retrieval system without written 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.

FROM THE EDITOR

Tools of the Trade

Written by
Sean Rhody



Author Bio:

Sean Rhody is the
editor-in-chief of Web
Services Journal. He
is a respected industry
expert and a consultant
with a leading Internet
service company.

SEAN@SYS-CON.COM

There's an old expression – "When all you have is a hammer, everything looks like a nail." There's a wealth of applicable comment in this expression. It's an admonition to see the bigger picture as well as a suggestion that to be a true craftsman, one must have the right tools.

This month our focus is on tools and automation for Web services. That seems very apropos for the idea of needing a variety of tools, each appropriate to the task at hand. As with craftsmanship, Web service development requires the developer to know what is the right thing to do, and the right tool to do it with.

Knowing what is the right approach is often a matter of trial and error, of learning what will work and what will not. In the world of Web services, one of the key areas for concern is the definition of the APIs that will be exposed as WSDL and UDDI entries. As we present a Web service interface to the world, we must frequently wrestle with the level of granularity that we expose as a result. Too little granularity and we run the risk of confusing our service consumers by not providing enough specificity for the service to be easily understandable (and therefore consumable). Too much granularity and we run the risk of shutting down the network with excessive communications.

Best practices garnered from other distributed computer paradigms such as CORBA, J2EE, and COM are of some assistance in this matter. As we examine the patterns and paradigms adopted in each of these platforms and note the successes and failures, we can draw analogies to our own Web services implementations, which are often implemented on one of these platforms. Just as in J2EE, we can make our services very granular, but we would expect to incur a significant performance penalty for doing so.

So best practices is one tool in our toolbox. Another is automation tools. No one should ever have to write WSDL. It just should not be a task to be done by hand. Like Web-page design, once you understand the underlying structure and concepts, there's no need to code HTML in Notepad when you can use something like Dreamweaver.

Similarly, tools are being developed that make it easier to create, run, and consume Web services, tools that take away the tedium of writing WSDL by hand and automate the process. In the Microsoft camp, this area is filled nicely by Visual Studio .NET and augmented by some serious .NET development at Borland. In the J2EE world, BEA and IBM are producing toolkits that enable you to write Web services with very few lines of code, using a wire-together approach rather than a pure coding paradigm. EAI vendors such as webMethods are also producing development platforms.

Additionally, the concept of orchestration and management is receiving due attention. As Web services move from trial or proof-of-concept approaches to full implementation, the need for management becomes ever clearer. Companies such as Actional are developing approaches to meet this need.

But even further, as we start to see a wealth of Web services we realize that coordination of processes and atomicity of transactions are also critical. The WS-I and OASIS, as well as vendors such as HP, are trying to define both transaction standards and a standard model for description of the orchestration process in the hope of standardizing the approach and qualifying the need. It's always difficult to prove the necessity of orchestration systems – you can typically hardwire the code in less time. But the ability to make on-the-fly changes and the opportunity to finally have the business achieve the definition rather than having the IT department code it has significant value. What's missing is a strong case for ROI on this technology, but it will arrive shortly.

So we can see that we need more than a hammer and nails – we need a whole Web services toolbox. This issue highlights some of those tools. Enjoy, and don't use a screwdriver to pound in a nail – it doesn't work well. ©



Sonic Software

www.sonicsoftware.com/websj



Phillip Hallam-Baker

Phillip Hallam-Baker is principal scientist and Web services architect for VeriSign, Inc., and is responsible for driving and delivering key security specifications and technologies through industry-recognized standards bodies and other organizations. Phillip is the coauthor of the XML Key Management specification, which marries XML and PKI technologies for higher levels of e-commerce security. He also coauthored the WS-Security specification with Microsoft and IBM. PBAKER@VERISIGN.COM

- **Data integrity:** Data exchanged by Web services must be safe while in transit.
- **Authentication:** Web services must positively identify the services with which they communicate.
- **Authorization:** Web services must intelligently restrict access to sensitive data and functions.

There are a number of standards and specifications floating about right now that attempt to address each of these specific areas. Most notably, VeriSign, Microsoft, and IBM recently co-authored a spec called WS-Security that attempts to add a layer of security to SOAP messages. WS-Security will serve as the foundation for a number of subsequent specifications the three companies hope to sponsor, including WS-Policy, WS-Trust, WS-Privacy, WS-Secure Conversation, WS-Federation, and WS-Authorization. Some of these names may change, but this roadmap does show a strate-

Enabling Trusted Web Services

Embed trust and security into the new infrastructure

Web services are demonstrating their value and exhibiting the potential to substantially enhance enterprise productivity and reduce operating costs. But they will never reach their full potential without two things: trust and security. That's because Web services are based on open, dynamic exchange of valuable data and services. But for everything to work the way it's intended, those deploying Web services must be able to ensure that the data or services being exchanged are kept confidential, secure, and reliable.

To deploy trusted Web services, you really need five things:

- **High availability:** The Web services must be easy to find using public or private directories.
- **Privacy:** Communications absolutely must be safe from eavesdroppers.

gic approach to building out the standards and technology for enabling trusted Web services.

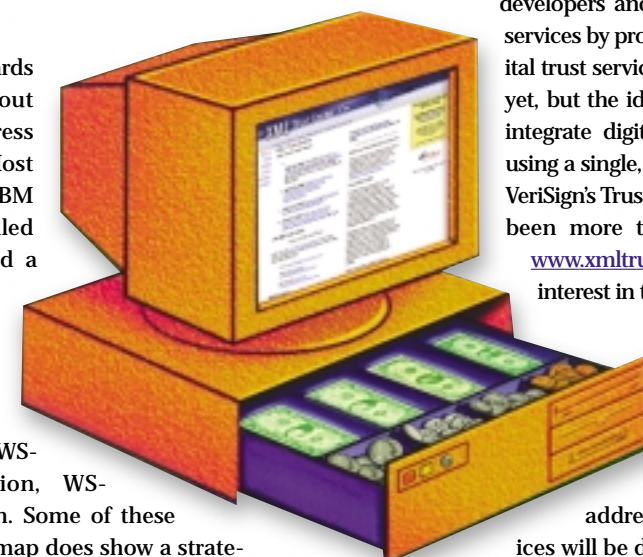
It will be critical, however, to keep the industry on track. No significant Web-based technology has taken off without addressing security issues in some way. During the past decade, VeriSign spurred the first wave of secure Internet commerce by embedding the VeriSign Trust Root in all the major Web browsers.

We must be just as diligent in trying to embed elements of trust and security into the fabric of Web services infrastructure. Loosely coupled applications must be able to make critical determinations at runtime, such as whether to entrust an inquiry, reveal strategic data or invoke contingent services. In addition application users who do not know one another must have access to a secure payment mechanism that allows them to pay for services that operate via the Web services platform. Finally, enterprises must provide a mechanism that allows applications to easily locate one another across the Internet and determine their suitability for interaction based on predefined criteria.

To meet these requirements, there must be an underlying trust infrastructure that is dynamic, reliable, and easily accessed by many applications. This infrastructure and the digital trust services that it provides must be integrated into Web services at both the network and application levels, enabling enterprises to securely utilize existing technology assets while participating as fully as possible in the emerging digital economy.

A number of industry players, including VeriSign, IBM, Microsoft, Sun, Oracle, and BEA, are currently cooperating to make it easier for developers and partners to create or resell trusted Web services by providing a single resource for integrating digital trust services into Web services architecture. It's early yet, but the idea is that developers will be able to easily integrate digital trust services into their Web services using a single, unified API, which is currently provided in VeriSign's Trust Services Integration Kit. So far, there have been more than 2,000 downloads of this kit from www.xmltrustcenter.org, indicating tremendous early interest in trusted Web services.

In any case, efforts to integrate digital trust services across all major Web services platforms will continue, and work on standards and technology will move forward. If it doesn't, and the industry doesn't adequately address issues of trust and security, Web services will be dead on arrival. ©



SpiritSoft

www.spiritsoft.com/climber

How Web Services Will Revolutionize CRM

An opportunity to reexamine a common methodology



Web services will revolutionize CRM (customer relationship management). This revolution will affect not only how sales, marketing, and service professionals interact with customers, but also how IT departments implement and support the technology. CRM is, in fact, suffering from precisely the constraints that Web services addresses architecturally. And several unique characteristics of CRM technology, data, and workflow exacerbate these constraints beyond other vertical applications.

The opportunities presented by this revolution will be balanced by equally powerful pitfalls. This revolution will demand a rethinking of the way many aspects of CRM technology are structured – in particular, data integration. As such, CRM is a unique case study in how to apply a new technology to an old problem

AUTHOR BIO:



David Cameron is vice president of product integration at AptSoft Corporation, an enterprise software solutions company pioneering the application of Web services to CRM integration at the event level.
DAVID.CAMERON@APTSOFT.COM

What Is CRM, and Why Is It So Hard?

Broadly speaking, CRM is the set of business processes that address customer acquisition, retention, and cross-sell/up-sell opportunities across sales, marketing, and service functions. Thus, CRM is big, and big can mean costly, unwieldy, and complex.

In fact, today most large organizations are CRM victims whose senior executives cringe whenever someone (particularly a vendor) mentions a certain three letters in a certain order. Several characteristics unique to CRM are behind the current challenges:

1. **CRM is dynamic.** Unlike back-office processes like logistics, customers are in control in CRM. Thus, organizations must constantly adapt to the way customers want to interact. Further, complex market and competitive forces are constantly affecting the way sales, service, and mar-

keting happens. All this translates to a constant stream of system changes.

2. **CRM is multichannel.** Customer interaction happens across multiple channels (phone, face-to-face, Web, e-mail, direct mail, third party, mass media). Activity in each of these channels affects how customers perceive activity in other channels. Thus, channel awareness is important.

3. **CRM cuts across organizational boundaries.** Ask who owns the customer and practically every department in the front office will raise their hand. Product organizations, channel organizations, and geographic organizations all interact to create a set of complex dependencies for customer-facing processes.

4. **CRM processes are complex.** Rarely do marketers ask the question, “How many customers do I have in California?” They do, however, ask, “How many customers do I have in California who have purchased at

least twice, have children, and whose last purchase was over \$200?” Then they start drilling down from there, resulting in perhaps dozens of variations on the same question.

5. **CRM data is typically everywhere, and metadata matters.** Unlike most other applications, CRM can involve information from all other applications. In this respect, CRM is bigger than you think. Invoice data? Sure, it has purchase history. Service call? Of course, we need to gauge level of satisfaction. Third-party demographics? You mean I don’t know if they have children by looking at my internal data? And the meaning of fields of information, even those with identical “tags” like “invoice_ amount”, can mean very different things in different databases (think taxes and discounts, net versus gross, etc.). Tiny differences in the meaning of data that are not

taken into account can cripple CRM analysis and implementation.

All of these factors combine to create very complex applications covering multiple touch points with proprietary data structures and self-contained integration.

The classic application suite approach that has evolved in response to these factors and all of the subsequent integration and maintenance headaches that have ensued provide the great opportunity for Web services. Simply put, organizations have over-invested in CRM and are not realizing the promise of their existing systems. In addition, these investments encompass multiple applications and databases. With today's strategic imperative to "grow revenue one customer at a time," CRM business processes are reaching maximum importance just when the enterprise has tired of big-footprint implementations. Rare is the company that will replace the functionality of one vendor with another just to standardize on a suite! The integration of such a coordinated, distributed architecture is ideal for Web services. Overcoming the integration challenge can provide the "killer app" all new technologies need.



The Integration Challenge

In order for CRM to work, organizations must be able to design, implement, and maintain business processes across applications and databases. One of the oldest problems in computing, application integration continues to impose constraints on the lines of business as they attempt to generate revenue through CRM business processes.

Very early solutions focused on writing custom code at the application layer, attempting to extend business logic across applications. The result, spaghetti code, became unworkable as scalability requirements increased. The current generation of solu-

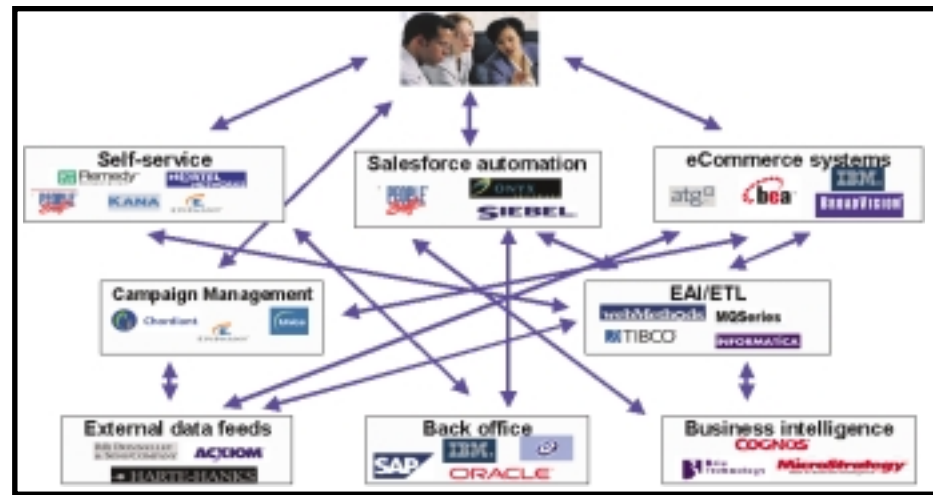


FIGURE 1 Disparate islands of technology: A tangled web of integration

tions focuses lower on the integration "stack," at the data level. These solutions, from custom data movement code to extract-transform-load (ETL) synchronization and enterprise application integration (EAI), have focused on the movement of data from one application to another, either in batch mode (ETL) or in near real time as "transactions" (EAI) that automate workflow by transporting data from one application to another.

Today, however, the problem of implementing and maintaining application integration has not only remained unsolved, but has gotten worse. The reason for this is simple: CRM applications function best with centralized data repositories (warehouses or marts), resulting in the creation of myriad data silos, and creating a Gordian knot of data movement processes within large organizations, locking valuable data definitions, business logic, and transformation logic into custom code or proprietary ETL and EAI applications.

The result is a tangled, confusing array of interdependent programs both within the enterprise, and connecting the enterprise to the outside world. Maintaining this complex environment, and assuring that the business processes inherent in the programs contribute to high-level corporate goals and objectives is, increasingly, a nearly impossible task.

Figure 1 depicts a computing infrastructure that's not at all atypical. The blue lines indicate the various point-to-point ETL, EAI, and custom tools used to link systems and applications together to form CRM

processes. Imagine the integration nightmare when a new department's Web site comes online, or when the data mart's schema needs to be updated!

The Web Services Revolution

So how does Web services architecture address this challenge? Web services promises to enable applications to communicate above the data integration level, and even above the application level. Web services enables a new generation of application integration applications that allow organizations to identify critical events and act on those events in real time, while maintaining all of the business logic already inherent in existing CRM investments within its applications.

Through lightweight adapters at each CRM application or database, events of interest to sales, service, and marketing that occur in one system, say a large withdrawal, or a complaint by a premium customer, can be readily and instantly identified, evaluated based on context and knowledge from around the enterprise via centrally stored business rules, and acted upon by sending a native transaction to another application that is also linked through a lightweight adapter. The adapter components of the architecture communicate with and extend the existing business logic in systems, dramatically reducing the application integration challenge (see Figure 2).

This approach, which I call "event coordination," provides the following advantages:

- Creates a "hub and spoke" messaging sys-

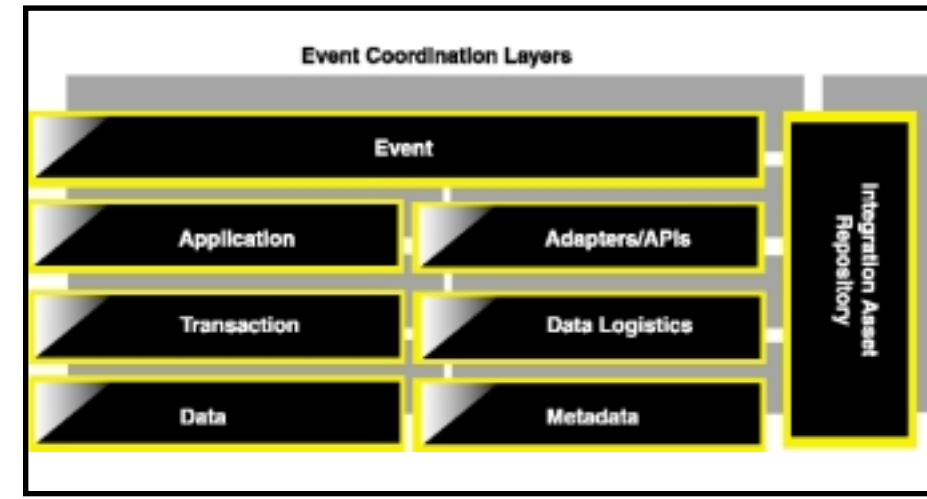


FIGURE 2 The Event Coordination stack: Integration above the application layer

- tem, replacing the point-to-point nightmare described previously
- Reduces reliance on expensive consultants writing custom code
- Enables more efficient use of resources such as hardware and bandwidth through distribution of functionality
- Extends the integration framework to multiple service providers and third parties outside the firewall
- Provides a balance between decentralized application infrastructure and centralized coordination of activity across applications, overcoming data and functionality "silos"
- Utilizes all of the business logic already contained in an organization's infrastructure without modifying or transferring it

“ This revolution will demand a rethinking of the way many aspects of CRM technology are structured ”

The approach plays to Web services' strengths and dovetails nicely with current trends in technology, specifically:

- Web services development is cheaper than proprietary approaches due to the amount of free basic infrastructure available.
- Web services standards are well developed, allowing the use of standards-based products for development and administration.
- The tremendous momentum of Web services has forced many EAI, ETL, and application suite vendors to reveal their business logic through SOAP APIs.
- An emphasis on Web services security enables distributed applications to extend beyond the firewall.

Data Provisioning: The Next Challenge

While facilitating application communication at the event level conforms nicely to the advantages of Web services architecture, provisioning data to decentralized functionality looms as a potential pitfall.

Application suites and other complex bundles of functionality evolved because of heavy reliance on standardized and reliable data in the form of centralized data warehouses and data marts. In fact, the decades-old trend toward data centralization is one of the main rationales for big-footprint applications. So, any Web services approach that seeks to decentralize functionality must address the data provisioning issue. Simply put, it is impossible to build a single database to provision all Web services, and it is equally impossible to implement custom code to provision data for each Web service.

Happily, one of the base enabling technologies in the Web services architecture promises to facilitate the data-provisioning challenge: XML. XML enables applications that can map decentralized data to straightforward business meanings. Thus, the term "customer" or "product" can be standardized for an audience and mapped to the various data elements in their native data stores. Further, business rules that describe how one definition of a product (say, an SKU code) can be standardized to conform to the business definition for that audience can be encapsulated in the XML. Thus, a lingua franca is born.

This common language can then serve as a reference point for disparate Web services to fetch data, transform it into the definitions expected by the audience, and then act accordingly. Otherwise known as a "canonical object" approach to metadata (where each term such as "customer," "product," or "campaign" is an object with specific attributes and meanings), this lingua franca overcomes one of the thorniest issues in CRM: data integration to support functionality. Most of the cost of the implementation of an application suite is the research, design, implementation, and maintenance of the processes that take data from around the organization, standardize it, load it, and update it over time. By leaving data in its original location, but making it accessible via mappings and business rules stored as XML, Web services architecture can dramatically reduce the cost of implementation and maintenance.

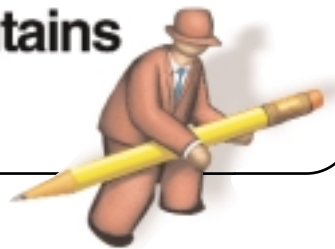
A side benefit to this approach is reusability. Definitions and business rules stored as XML can be provisioned to any application or process, not just the Event Coordination framework. And, all of the original research and design is no longer a one-off project: work that would have to be done anyway is accessible again and again.

Conclusion

Every new technology trend offers opportunities to rethink methodologies previously thought to be common knowledge. Web services offers this opportunity to CRM and, in doing so, promises to revolutionize an area of computing that is simultaneously critically important and desperately difficult. ©

Designing Web Services with XML Signatures

A little extra effort maintains data integrity



XML signatures apply digital signatures to XML documents. Digital signatures let parties that exchange data ensure the identity of the sender and the integrity of the data. This last item is a benefit that physical signatures can't provide.

Digital signatures don't have the legal status that physical signatures have, at least not yet. Over the last few years this has been changing, as the federal government and many state governments have moved to accept digital signatures as legally binding for some applications, where they identify the sender and provide nonrepudiation (the signer can't deny ever having signed).

Since Web services that use SOAP exchange XML documents, they can include XML signatures. Web service developers who produce Web services with XML signatures will be able to realize the benefits of those signatures, and may be able to use them in place of physical signatures where the legal community has agreed they are binding. Putting XML signatures into a Web service, however, is not trivial. In this article I'll provide some background on XML signatures, and then explore how you can incorporate them into your Web services. I'll develop an example Web service, written in Java for Apache Axis, that uses an XML signature, to illustrate how it can be done.

XML Signatures

XML signatures are the work of the W3C XML Signature Working Group (www.w3.org/Signature), which produced "W3C Recommendation on XML Signature Syntax and Processing" (February 12, 2002). The recommendation gives an XML schema for the signature syntax and also refers to recommendations on canonicalization, key type, transformation algorithm, digest algorithm, and identity certification by authorities. The recommendation specifies what may go into each part of an XML signature.

IBM and Microsoft have been leading an effort to develop Web service security, including the use of XML signatures in Web services. They produced the WS-Security specification (www-106.ibm.com/developerworks/library/ws-secure) along with VeriSign, Inc. They recently submitted this specification to OASIS, which created a technical committee dedicated to that proposal. This specification defines how a SOAP header entry can be used to carry an XML signature referring to information elsewhere in the same SOAP envelope (the basis for my code example). Placing the signature in a SOAP header makes it easier for software to locate.

XML signatures are the application of digital signatures to XML – you can digitally sign part or all of an XML document. To create a digital signature, you calculate a checksum (digest) of the XML to be signed,

and encrypt the digest using the private key of a public/private key pair (PKI). You send the encrypted digest along with the data to be signed, and also attach the public key and a certificate issued by an authority to identify the owner of the public key (the latter two items don't have to be passed if the receiver already has them). The receiver decrypts the digest and checksums it against the received information. If it matches, the data's integrity is assured and the identity of the sender is tied to the public key. By using the certificate, the receiver can be assured that the public key belongs to the desired sender, and not someone else who may have hijacked the digitally signed data and substituted their own keys, digest, and signature.

The XML signatures recommendation doesn't concern itself with the transport used to get the XML document to its destination. XML signatures could be applied to XML documents that are e-mailed or carried on a floppy disk. But, of course, sending information across the Web opens it up to attack, making security more important, and that is exactly the case for Web services.

There has been recent legislation on both the state and federal levels covering the use of digital signatures. (For a good survey of this legislation, see the McBride Baker & Coles Web site legislative tables [www.mbc.com/ecommerce/ecommerce.asp].) For example, the use of digital signatures has been addressed in federal legislation such as the "Electronic Signatures in Global and National Commerce Act" (E-SIGN), and has been used on a trial basis for IRS tax returns.

Using XML Signatures in Your Web Service

First, let's assume you are working on an application where you need the benefits of




Author Bio

Mark Young is vice president of Kamiak Corporation, the publisher of Omniopera WS, a WSDL and XML Schema editor. He is an expert in XML Schema and WSDL, and has written object-oriented libraries to model them. Mark has 20 years of software development experience, has been a software team leader for many years, and has developed software for applications from telephony to sales automation. MARK@KAMIAK.COM

Rational Software

www.rational.com/offer/javacd11



XML signatures. In my example, I'm developing an insurance claim system wherein agents can submit claims containing accident reports. Each agent is assigned a private/public key pair, which is installed on his or her computer in a keystore (a database of public and private keys and certificates). Note that you can use keytool (from Sun Microsystems) to create keystores for your own work. The agent signs each accident report using the private key (by telling the application where the keystore is and providing the proper identifier and passwords). The agent submits the claim containing the signed accident report(s) via a Web service to the central office.

At the office, each XML signature is verified, and if the identity of the submitter is accepted, the claim is processed. By verifying the XML signature, the central office has not only verified the identity of the sender, but also that the information in the accident report has not been altered after it was signed.

Now let's consider how to design and implement the Web service. For my example I will be using Apache Axis (available at <http://XML.apache.org/axis>). I'll be writing the code for my Web service in Java, and I'll develop a message-style Web service. Many Web service developers start with Java code and generate the portions of the Web service that talk XML from it automatically, shielding themselves from dealing with XML. For this insurance claim Web service though, where an XML signature needs to be applied to the XML that represents my data, I will produce my own XML.

I choose to design my service interface first so that I'll know what the XML should look like. I do this by authoring a WSDL file that describes my service. Its data definitions are done in an XML schema. Since the WSDL file should describe the messages that pass across the wire, my data definitions show the XML signature along with the data it signs. Listing 1 shows an excerpt from the WSDL file, the first portion from the XML schema that defines my data (the listings for this article can be found at www.sys-con.com/webservices/sourcecode.cfm).

The signature element is carried in the SOAP header, and I define it as a reference to the XML signature Signature element. The AccidentReport element is the data to be signed; it's in the SOAP body. The top-level element in the SOAP body, Claim, contains an AccidentReport element whose type is AccidentReportType.

The second portion of Listing 1 shows the WSDL message definition for the request message: the first part (name = "Claim") uses the Claim type, and goes into the SOAP body part of the SOAP envelope. The second part (name = "Signature") refers to the W3C Signature element, and goes into the SOAP header part of the SOAP envelope. When I define the WSDL describing my Web service operation, I use document/literal. I don't want SOAP encoding applied to the data being signed, and I don't plan to do any such encoding myself.

Now I can implement the Web service. I use the Apache XML security packages (available at <http://XML.apache.org/security>), which you can use from your Java classes, and which operate on DOM objects (which represent XML data). Therefore, I need to operate on the messages in my Web service at the DOM level.

To do this, I write a message-style Web service; I don't use Apache's ability to automatically serialize/deserialize Java classes. I use the Axis sample LogHandler.java (which you find in the Axis security samples), which intercepts incoming messages. It receives a MessageContext, from which it gets the Signature element out of the SOAP header and verifies it against the SOAP body. If the verification succeeds, the handler passes the message (without the signature) on to the Web service server itself. The server proceeds with processing the claim (in my example, I used a simple server that echoes back the elements it received). If the signature verification fails, the log handler raises an exception, which causes a SOAP fault to be returned.

On the client side, I also work at the DOM and message level. My code translates the data to be signed into DOM (along with the data that will not be signed), constructs

the XML signature, and puts both into their proper place in a SOAP envelope (the signature in the SOAP header, the data in the SOAP body). Finally, it sends the message (via the invoke method). Listing 2 shows the class that represents the data to be signed; the class includes a method asDOM() that returns a DOM representation of itself. Listing 3 shows the main body of the client, which populates the Claim to be submitted; creates a SOAP envelope; causes the correct parts of it to be signed; and calls call.invoke(), which sends the outgoing message. Listing 4 shows excerpts from the class that signs the AccidentReport portion of the SOAP envelope. For more details on how to do the signing, canonicalization, serializing, and deserializing, see the Apache Axis security samples.

Of course, my code could just as easily have signed the entire SOAP body, instead of just signing the AccidentReport. This would have a different effect – it would ensure that the entire message has been delivered correctly, and not necessarily that the correct party vouches for the AccidentReport.

Conclusion

As you can see from my example, it can be a bit of a struggle to use XML signatures in a Web service. Their relationship to the XML data they sign makes XML signatures hard to use in a Web service tool that seeks to hide XML from its user. It would take special mechanisms to express the relationship between signed data and signature at the programming language level, so that a Web service tool could handle the details for you. As my example shows, in Apache Axis it is easiest to get the XML signature out of the SOAP header of the message (in a log handler) before the main part of the Web service begins. So the developer must write some code that can work in a separate handler. This makes the developer's job a bit harder, since he or she must split the server logic into two parts, and possibly make them communicate (so that the handler can pass the service proper any signature information it needs). ©

Actional

www.actional.com/whitepapers/wsblues



It's critical that service-oriented architecture (SOA)-based Web services solutions provide high-performance, reliable, extensible, scalable, and open standards-based communication and integration for both internal and external applications that can be easily monitored. To meet these goals, it's imperative that companies design their Web services correctly, check service quality, conduct active performance management, and continuously monitor the end-point integrity of Web services. This article discusses monitoring and optimizing the design and runtime performance of Web services.

Written by Gunjan Samtani and Dimple Sadhwani

Essential Traits of Web Services

Web services is just another distributed-computing technology for revealing the business services of applications on the Internet or intranet using open and standards-based XML protocols and formats, but

some essential traits make performance management and monitoring of Web services both different and necessary.

Simple Object Access Protocol (SOAP) is the wire protocol for Web services. SOAP HTTP calls cost in terms of network traffic and latency, CPU cycles at the SOAP server, and possibly SQL latency at the database server. SOAP overhead includes extracting the SOAP envelope and parsing the contained XML information. Further, XML data cannot be optimized much. Other essential traits include:

- The overhead of encrypting and decrypting data
- The overhead of using Universal Description, Discovery, and Integration (UDDI) for dynamic binding
- Dependence on the quality and service-

level agreements (SLAs) of external Web services

- Latency of dynamic binding
- Monitoring of end-point integrity

The core intent of Web services is to loosely couple disparate applications and platforms using standard interfaces and protocols. This is also its weakest link as far as predictability and runtime behavior are concerned and makes monitoring and performance management even more important.

The Current State of Monitoring

Before we look at Web service monitoring, it's important to understand the current state of application monitoring methodology and know the tools used within companies. The current state can be summarized as follows:

1. End-to-end monitoring and performance-management tools and technology are still evolving.
2. A proactive strategy is desired, as opposed to reactive monitoring and performance management.
3. Usage and existence of "island or silo management" and summing up, i.e., monitoring the individual components of an application and summing them up for the end result.

Typical reactions to production problems vary from "it's their system/API/Web service... not ours!" to "the database is extremely slow" to "the network is congested" to "everything is fine, just reboot your machine." Bottom line: in the absence of an end-to-end monitoring tool, no group is willing to take responsibility.

Monitoring Web Services

Both network and systems management monitoring are critical for Web services operations. In lieu of this, testing can be used to monitor Web services – some of the tests mentioned below have to be performed in all three application environments (development, quality assurance, and production):

- **Availability test:** Checks whether a Web service is present and ready for immediate use.
- **Performance test:** Measures the throughput (how many requests can be served in a given time frame) and latency (how much time elapses between the request and the response). This test is useful in evaluating performance in terms of how long it takes to service clients' requests.
- **Stress test:** Determines and monitors the breaking point of a Web service based on

AUTHOR BIOS:



Gunjan Samtani is divisional vice president, information technology, at UBS PaineWebber, one of the world's leading financial services firms. He has several years of experience in the management, design, architecture, and implementation of large-scale EAI and B2B integration projects. Gunjan is the primary author of *B2B Integration: A Practical Guide to Collaborative E-Commerce* (Imperial College Press) and *Web Services Architectures and Business Strategies* (Wrox Press).
GSAMTANI@UBSPW.COM



Dimple Sadhwani is a senior software engineer at Island ECN based in New York. She has many years of experience working for financial and telecommunication companies on large-scale trading systems, CRM applications, Internet/intranet portals, and client/server applications. Dimple is the coauthor of *B2B Integration: A Practical Guide to Collaborative E-Commerce* (Imperial College Press) and *Web Services Architectures and Business Strategies* (Wrox Press).
DSADHWANI@ISLAND.COM

load and permits only that level of load.

- **Reliability test:** Determines how capable a Web service is of maintaining the SLA and service quality.
- **Scalability test:** Checks whether a Web service is vertically and horizontally scalable.
- **Failover or high-availability test:** Ensures that the failover solution for a Web service is working correctly and that Web services clients aren't affected by the failover.
- **Integrity test:** Determines the correctness in execution of Web service transactions.
- **Corrective measures monitoring:** Checks the health of the Web services provider application for features such as deadlock, process being hung, and time-out periods. If the provider application is supposed to take corrective measures, its behavior should be monitored.

There's no doubt that Web services will be only one of several technologies to play a role in both enterprise application integration (EAI) and business-to-business (B2B) integration. For end-to-end application monitoring, which may be supported by one or many Web services, all the supporting application and platform components (such as messaging and databases) must be monitored as a single unit.

Performance Management of Web Services

Performance management of Web services should be discussed and tackled from two different perspectives: design and runtime. They're equally critical. Let's start with good design considerations that will impact the performance of a Web service.

- Web services invocation operations are expensive. It's preferable to use them in macro rather than micro cases, i.e., where a lot of work is to be done and a lot of information has to be returned.
- Depending on the nature of the Web service, asynchronous messaging can improve throughput. This, however, will come at the cost of latency.
- The design of a Web service should be such that there is very loose coupling between the presentation-tier clients and business services.
- "Business delegate" design patterns should be used whenever possible. This will hide implementation, lookup, invocation, and caching details from the client application. The business delegate will convert all infrastructure-related exceptions into business exceptions, shielding Web services clients from knowledge of the underlying implementation specifics.
- Whenever possible, implement a caching mechanism for business service information. This will speed up the application and reduce network traffic between the client and the provider.
- Avoid aggregating the data from data-supplying Web services in real time on demand. Instead, create a data-aggregating Web service that aggregates the data in a batch mode.
- Several XML parsers perform expensive operations (type checking and conversion, checking for the correct format) that make performance slower than optimal. Consider using a stripped-down XML parser that performs only essential parsing.

Problem Resolution

To achieve runtime performance management for Web services, the monitoring tool must be able to collect specified, defined measurement data; analyze it; and check for exception conditions. If any

exception condition has been met, as defined in the configuration, the monitoring tool should alarm the operators and/or take corrective measures. Exception conditions can be based on an individual parameter or a combination of parameters and can be defined using both thresholds and duration. This will allow for proactive detection of problems. For example, to deliver good SLAs, exception conditions can be set on the latency of Web services calls and throughputs, which will be monitored in real time by the monitoring tools.

"The monitoring tool must be able to collect specified, defined measurement data; analyze it; and check for exception conditions"

To achieve sophisticated runtime performance management of Web services, the tool should not only detect exceptions and take corrective measures, it should also be able to re-create the exceptions. Re-creation of exceptions will help eliminate performance bottlenecks by diagnosing the cause. Further, end-to-end reporting of all activities involved in a Web service operation will help the application developers and managers discover what's going on in operational Web services-based applications and, if required, take corrective measures. Finally, we can't stress enough that runtime performance management may require considerable storage and even processing overhead. Application-specific policies concerning the extent of Web services monitoring and

the length of data-retention periods may need to be developed. If the duration of data retention is long, you may need to develop data compression techniques. But the monitoring tools should provide all these features.

Web Services Monitoring and Performance Management

Here are the key factors for monitoring Web services, based on our experience:

- Use design patterns to alleviate performance bottlenecks.
- Monitor the whole transaction as a single unit.
- Make SLAs part of your application review process.
- Monitor the SLAs and take corrective measures.
- Ensure that Web services supporting mission-critical applications are monitored in real time with very little latency.
- Make sure that there's a single interface for monitoring, analyzing, and forecasting resource utilization for Web services.
- Define the exception conditions for performance management of both the threshold and the duration for proactive problem detection and resolution.
- Last, avoid chaining of Web services if possible, as it will make their monitoring even more complex. (Note: a Web service can be implemented as its own business process, or a business process can be composed of any number of Web services, which themselves can aggregate other Web services to provide a higher-level set of features. This results in chaining of Web services.)

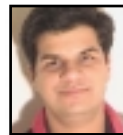
Internal First, External Later

Monitoring and performance management of external Web services is much harder than that of internal Web services, since the user may have no control over the quality of the Web service provided. In order for companies to start embracing Web services technology, it's crucial to start using Web services for private processes or internal applications before using them for public processes or external applications. Public processes and external applications pose greater risks as far as security, transaction management, auditing, and persistence are concerned – and monitoring them is even harder. ©

Sitraka
www.sitraka.com/jclassSS/ws



Reviewed by Prasad Joshi



About the Author:

Prasad Joshi works as a senior developer for Syntel, Inc. He has four years of experience in designing and implementing Web-based applications using Java and XML technologies.

Previously he worked on Microsoft technologies such as ASP and Visual Basic.

PRASADVJOSHI@HOTMAIL.Com

Parasoft SOAPtest

Verify functionality and scalability

During the past few years we've seen an exponential rise in Web-based applications, and we're currently beginning a transition to a service-oriented Web. Web services technology enables several functional elements (services) to communicate with each other. The key to the success of this technology is the ability to exchange messages between services implemented on disparate platforms.

We've embraced SOAP as the protocol for message exchange, and fortunately, it has received a good response from the key players in this technology. Loose coupling is a great feature of Web services technology, but at the cost of stringent constraints on the validity of the messages exchanged.

With Web services, the underlying idea of serving client requests isn't very different from the technologies we've grown accustomed to in the past few years. Based on experience with Web-based applications, simulated clients and servers are ideal for testing client and server behavior. As the technology receives the anticipated acceptance, performance testing under varying loads will be another requirement. Parasoft has addressed these specific needs with SOAPtest, a collection of tools that can be assembled to test certain scenarios and record the results.

Installation

The installation went very smoothly. It's

available with and without the JRE; I used the one with the JRE.

Features

After installation, I browsed through the documentation provided with the product and was overwhelmed by the variety of functionality. The documentation highlights specific functionality while describing the general features.

SOAP Client and SOAP Server

The core feature of SOAPtest is the ability to simulate SOAP-client and SOAP-server functionality. In Web application environments, the Web browser (client) tested the server-side applications. With Web services technologies, we'll have to rely on custom-built client applications. In most cases these client applications will have to be recompiled to test different scenarios. A tool that can simulate client behavior will be very valuable.

The SOAP Client tool makes simulating the SOAP client functionality easy. The tool provides many options for testing a variety of scenarios. It offers a choice between "RPC" and "Document" for the body type of the SOAP message. The message can be created in three different ways: Literal XML, Scripted XML, or Form Input. If WSDL can be provided, most of the options are prefilled.

I used this tool to create a test request to the famous "Stock Quote Web Service" deployed on my machine. I used the Form Input method to create the request message (see Figure 1).

Once all the required fields on the screen were set, I could run the simulated client by just clicking a button. I could send different requests by changing the options/data on the screen. At



TESTING ENVIRONMENT

OS: Windows 2000 Professional

Processor: Intel Pentium (550MHz)

Memory: 512MB

CORPORATE INFO

Parasoft

2031 S. Myrtle Avenue.

Monrovia, CA 91016

Phone: 888 305-0041

E-mail: info@parasoft.com

Web: www.parasoft.com

LICENSING INFO

Please contact Parasoft for licensing costs.

EVALUATION DOWNLOAD

A fully functional evaluation download (two weeks) is available at the Parasoft Web site (www.parasoft.com).



the same time, I could test whether the server could handle incorrect input data, incorrect method names, and so on. The response message, with the fault element, could be examined in case of an error.

The ability to chain the output to certain other tools is a really cool feature. The SOAP Client tool allows examination of the following types of data: the XML request data, deserialized response, and HTTP traffic data. Later we'll look at the feature used for chaining tools to view the data generated during a test.

SOAPtest lets us simulate the SOAP server; the documentation explains this feature in the "Deploying Web Services" section. Also, a tool is provided to automatically create the server stubs from

Altova

www.xmlspy.com

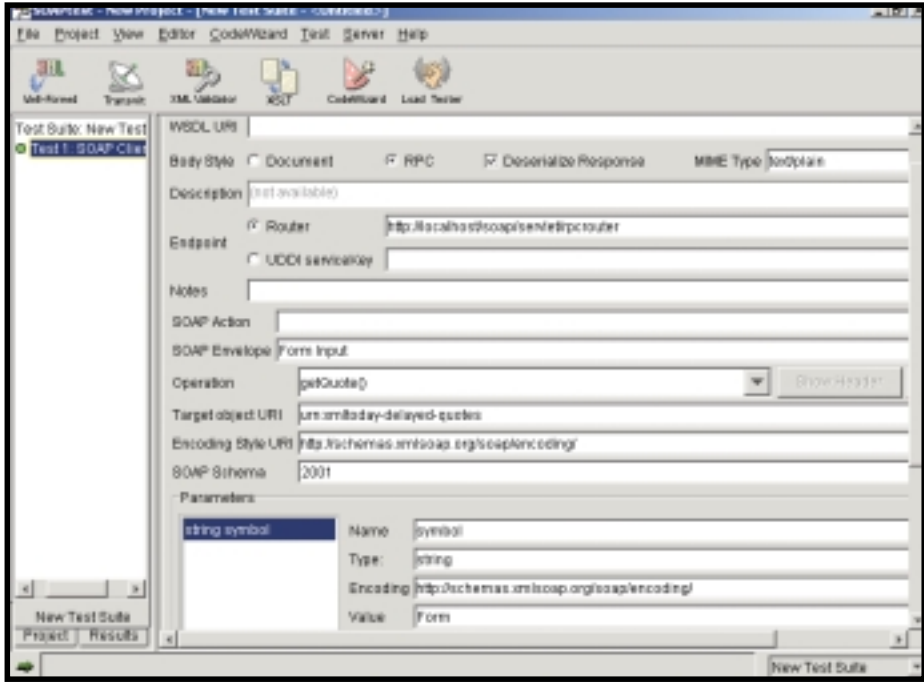


FIGURE 1 | SOAP Client input screen

the WSDL file; this tool can be used to emulate critical functionality that cannot be made available for testing.

In addition to the basic functionality of SOAP client and SOAP server emulation, SOAPtest has a myriad of other tools that can be of great help in performing the tests. As mentioned earlier, the available tools can be chained so that output from one tool is piped to one or more other tools.

Browser

The Browser tool can be used to display the data exchanged (e.g., XML request message, deserialized response, etc.) in a new browser window. I tried using the Browser tool to display the XML request message for the “Stock Quote Service” test. I just added the Browser tool as an output for the XML request of the SOAP Client tool and it allowed me to select from available browsers or add my own.

File Writer

The File Writer tool can be used to write out data. I also used it to write out all the HTTP traffic in my earlier test. (Writing out the HTTP data is another neat feature of the SOAP Client tool. This

is helpful because you can examine the request and response data from the transport layer.)

Check Well-Formedness/XML Validator

The Check Well-Formedness tool and XML Validator tool can be used to validate the XML message received as a response from a server.

Rule Enforcer

The Rule Enforcer tool allows you to check for certain patterns in the response message by creating custom rules. This tool has a companion tool, the Code-Wizard, which lets you create the custom rules graphically. There is separate documentation titled “RuleWizard Documentation” for using this feature. At first look, it seems a bit involved, but I think this tool could be very useful to test occurrences of certain patterns in the response message. Besides, all the tests can be saved and then later executed when there’s a change/update in the functionality of the Web service.

Diff

The Diff tool lets you compare the outputs. The output or response from a serv-

er can be compared to preset data in a file or a certain specific text data. Both the text data and the file name can be supplied to the tool. The output data, which is used in creating regression controls, can be compared in three modes: Text, Binary, and XML. Using these we can easily determine if the functionality of the server changes. I used this to verify the address information returned by a sample “Get-Address” Web service. If the XML mode of comparison is used, the results indicate the XPath location of the element that was modified.

Method

Another interesting feature is the Method tool, which lets you create your own application-specific tests – for example, checking database updates. The product supports coding in Java, JavaScript, and Python. The documentation provides instructions for configuring the Method tool.

Additional Functionality

SOAPtest is designed with extensibility in mind and has features to use external tools within the test scenario. The documentation says that any third-party tool can be integrated into a test scenario. There are many other support tools, such as the XML Cleanup, XSLT, etc. These can all be chained to process the test results or to save them.

Another important feature is load testing, which lets you create a profile with the desired tests to be performed. The profile can then be tested in different scenarios with simultaneous/staggered users. You can easily run a simple load test that shows the results in a tabular report giving the option to select/deselect the columns in the report.

Conclusion

Parasoft seems to have given serious consideration to the experience of testing Web-based applications. SOAPtest provides almost all the features found on the wish lists of programmers and QA professionals. I recommend this product to software developers and QA professionals who want to more easily and efficiently verify the functionality and scalability of their Web services throughout the development life cycle. ©

Macromedia

www.macromedia.com/go/CFMXlight

Secure Web Services

Using SSL as an alternative to VPNs

Businesses need to provide their users with a method for securely connecting to their networks while minimizing the costs associated with providing this service – and also providing end users with as much convenience as possible.

As businesses embrace Web services as the method for delivering their applications, they are struggling with security issues. Network World recently reported that the top worry for IT executives deploying Web services is security. SSL (Secure Sockets Layer) can provide a viable alternative to Virtual Private Networking (VPN) companies for securing Web services.

Remote Access

Originally, remote end users connected to their corporate networks using dial-up modem services over POTS (plain old telephone service). This essentially provided businesses with a private connection to an end user, albeit temporary in nature. The primary security concern was one of authentication of the end user – guaranteeing that the business was letting the right people access the network through its modem pool.

As the Internet became ubiquitous, businesses longed for a way to eliminate the long-distance charges generated by their dial-up remote access services. End users were dialing into their ISPs locally to access the Internet with no long distance charges – why not just let them access the corporate network via the Internet? The simple answer was security. VPN companies came to the rescue.

VPNs

VPNs typically put specialized software on the client machines as well as a machine acting as the gateway to the cor-

porate network. These pieces collaborate to encrypt traffic between the end points, guarantee the identity of the remote users accessing the corporate network, and guarantee that end users connect to the right place. Businesses can enjoy the savings of eliminating long-distance charges while maintaining the security of a private connection.

There is, however, a downside. The specialized software that has to go on the client machine costs both time and money. The client software itself must be purchased and installed on every client machine that will be enabled to access the corporate network. Anyone who's been involved in these rollouts knows that words like "incompatible," "conflicting programs," and "pilot error" make the cost of deploying this crucial service much higher than simply the price of the software at the end points – especially when dealing with thousands of remote users.

SSL

The Internet, of course, was (and is)

growing by leaps and bounds. But consumers at large were wary of sending their credit card information over the Internet and being defrauded. SSL was popularized as a method to eliminate this concern. As long as the little lock or key icon popped up in the end user's browser, he or she felt more at ease and willing to conduct transactions over the Internet.

Originally, SSL delivered two basic functions:

- It allowed the browser to be certain that the site being connected to was genuinely the one requested (by using a form of authentication).
- It secured the data that was in transit between a browser and a Web server by using encryption.

SSL allows end users to guarantee the identity of the server to which they're connecting. Certificate authority (CA) companies such as VeriSign sell certificates that are installed on the SSL server. The CA acts as an objective third party that forces the business requesting the



AUTHOR BIO:



Jeff Browning is a product manager at F5 Networks, Seattle. F5 is a leading provider of integrated products and services that manage, control, and optimize Internet traffic delivery. J.BROWNING@F5.COM

certificate to prove its identity prior to being granted a certificate. End users or browsers can verify that the SSL server to which they're connecting has a valid certificate issued by a CA and actually belongs to the business to which they're attempting to connect using SSL. Finally, certificates are the mechanism used to associate a unique key used for encryption with a particular SSL server.

The browser and server exchange keys in order to be able to negotiate an encrypted session. SSL then encrypts data while it's flowing between the end user and the SSL server to secure the traffic while it's in transit.

These functions have been crucial to the success of online business. Without them, end users wouldn't have the peace of mind needed to share information required for completing business transactions over the Internet.

So Why Not Just Use SSL Instead of VPNs?

Every system that's used to connect to the Internet has the client software installed, by default, as SSL into every browser. End users are familiar with it. SSL is a widely adopted standard. There's no cost for the client software. There are no integration issues on either the client or corporate network side. So what's missing?

The majority of the SSL benefits discussed have been end user-centric. In order for SSL to be successfully used as a viable alternative to VPNs, another element is necessary – essentially, a method to control which clients are allowed access to the corporate network. The SSL-based solution must be able to guarantee the identity of the end user attempting to access the corporate network and decide whether he or she is allowed access. This can be accomplished using client certificates. The company can simply act as its own CA and have end users download certificates. This allows coverage of the basic security tenets: "who you are" (typically a user ID), "what you have" (in this case a valid company-issued SSL certificate), and "what you know" (a password).

This method allows the company to guarantee that only end users with valid certificates are able to access the network. The authentication must occur at a gateway point prior to the remote user's actual-

ly gaining access to the network. The key is having a gateway solution that allows a business to enforce these policies easily. With that in place, we have the security issues addressed – encrypted traffic between the end points, guaranteed identity of the remote users accessing the corporate network, and a guarantee that end users are connecting to the right place – all without the cost or administration problems associated with VPN solutions.

Coexistence

While the SSL solution works perfectly to secure Web services or Web-enabled applications and address the concerns expressed by IT executives, VPN technologies provide a few things that SSL can't – dictating that the technologies coexist. VPNs provide a solution for applications that aren't Web enabled, such as client/server-based applications, print services, and general file sharing. While SSL can certainly address downloading files through a browser, there isn't a solid solution for the other two applications at this time.

Businesses now have the opportunity to supply an SSL-based solution to the 80% of their user population that likely uses only 20% of the applications available (VPN services will continue to be required for the other 20% of the population). This shift will result in tremendous savings for businesses in terms of both time and money through:

- Elimination of costs associated with specialized client software
- Reduced help desk calls
- Lowered demands on IT
- Increased ease of use for remote users

Conclusion

Not exactly "adios VPNs, hello SSL." But as businesses embrace Web services through such efforts as Microsoft's .NET strategy (and J2EE-based platforms for Web services) and the Web enabling of most major business applications available now or within the near future, IT executives will be able to say "adios" to VPNs for a greater percentage of their end users and enjoy the bottom-line benefits as a result.

Reference

- Fontana, John. (2002). "Top Web services worry: Security." www.nwfusion.com/news/2002/01_21webservices.html?docid=7747 ©

PUBLISHER, PRESIDENT, AND CEO

Fuat A. Kircaali fuat@sys-con.com

COO/CFO

Mark Harabedian mark@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

ADVERTISING ACCOUNT MANAGER

Megan Ring-Mussa megan@sys-con.com

ASSOCIATE SALES MANAGERS

Carrie Gebert carrie@sys-con.com

Alisa Catalano alisa@sys-con.com

Kristin Kuhnle kristin@sys-con.com

Leah Hiltman leah@sys-con.com

SYS-CON EVENTS

VP, EVENTS

Cathy Walters cathyw@sys-con.com

REGIONAL SALES MANAGERS, EXHIBITS

Michael Pesick michael@sys-con.com

Richard Anderson richard@sys-con.com

CONFERENCE MANAGER

Michael Lynch mike@sys-con.com

CUSTOMER RELATIONS/JDJ STORE

MANAGER, CUSTOMER RELATIONS

Anthony D. Spitzer lony@sys-con.com

CUSTOMER SERVICE REPRESENTATIVE

Margie Downs margie@sys-con.com

MANAGER, JDJSTORE

Rachel McGouran rachel@sys-con.com

WEB SERVICES

WEBMASTER

Robert Diamond robert@sys-con.com

WEB DESIGNERS

Stephen Kilmurray stephen@sys-con.com

Christopher Croce chris@sys-con.com

Catalin Stancescu catalin@sys-con.com

ONLINE EDITOR

Lin Goetz lin@sys-con.com

ACCOUNTING

ASSISTANT CONTROLLER

Judith Calnan judith@sys-con.com

ACCOUNTS RECEIVABLE/COLLECTIONS SUPERVISOR

Kerri Von Achen kerri@sys-con.com

ACCOUNTS PAYABLE

Joan LaRose joan@sys-con.com

ACCOUNTING CLERK

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: \$99.99/YR

ALL OTHER COUNTRIES: \$129.99/YR

(U.S. BANKS OR MONEY ORDERS)



Written by Joseph A. Mitchko



About the Author:

Joe Mitchko is the product review editor
for Web Services Journal.
JOE@SYS-CON.COM

AltioLive 3.0 from Altio, Inc.

Rich client presentation for Web services

Altio, Inc., is about to announce the preview edition of AltioLive 3.0, a presentation server platform for “rich client” Web applications. In addition to its current architecture, this release provides a Web service adapter capable of providing real-time synchronization, as well as an improved IDE, to an intelligent front-end client.

In what looks like an X-Windows server in a browser, the AltioLive platform offers a desktop application environment complete with most of the GUI elements missing from browser-based Web applications, including drag-and-drop and other desktop controls. It's designed for those applications that demand a more complex and interactive interface, including real-time updates of data-oriented controls. In fact, the Altio Presentation Server (APS) engine and associated architecture are a precursor to the next generation of Web applications, providing a higher degree of user interactivity and ease of use while retaining the deployment costs typical of browser-based applications. The platform consists of the APS; a set of client modules (for Windows GUI, WAP, and other clients); and the AltioLive Development Edition (ADE) integrated development environment.

The client side of the platform is data driven and resides on the browser as a 200K Java applet. The AltioLive applet runs out of the box across any browser and any Internet-enabled device, including cell phones, TVs, and PDAs. The GUI environment was very responsive and using it was similar to the experience of navigating through the X-Windows desktop environment on a Linux or Solaris workstation. I had to keep reminding myself that all of this was running inside a Web browser.

The APS is servlet based and runs in most of the leading application servers on the market,

including WebLogic, WebSphere, and Sun ONE. It also works standalone using a Tomcat application server provided with the installation package.

For those of us nostalgic about rapid client/server application development using tools such as PowerBuilder, AltioLive has a surprise. The ADE console contains a window painter, scripting utility, and test environment akin to PowerBuilder or Visual Basic IDE software. Surprisingly, the ADE itself is an AltioLive application and, therefore, executes within a Web browser.

With this release, you can rapidly create application views (GUI) for a specific Web service given nothing more than the URL of the Web service WSDL. You can easily integrate Web service operations into an application with only a few event property settings. The APS provides real-time data synchronization to open client sessions and will monitor or poll data sources for any updates. It will then make the appropriate updates to the client application. All communication and programming of the client is XML data driven and fits within Web service-related architectures.

The client part of the platform is compati-

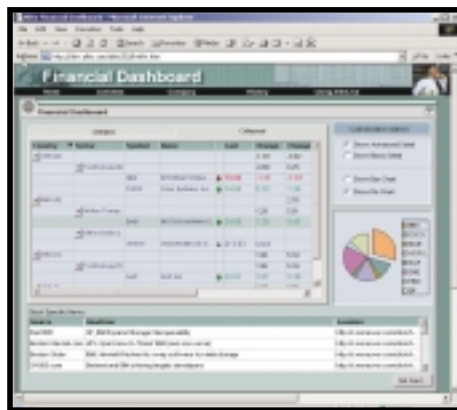


FIGURE 1 Using the AltioLive WSDL Wizard to connect to a Web service



COMPANY INFO

Altio, Inc. 529 Main Street Suite 208 Boston, MA 02129 USA	Phone: 617 886-9570 Fax: 617 886-9580 e-mail: info@altio.com Web site: www.altio.com
--	--

ble with most browsers capable of running Java applets and, since it is not sensitive to HTML browser incompatibility problems, it can run on some of the older Netscape and Microsoft browser releases.

Although having a full “desktop-like” environment is not a replacement for today's HTML-based applications, it addresses the needs of applications that require a more sophisticated and interactive user interface. One example would be a stock-trading application that packs a lot of real-time updated information on to a trading desk screen (see Figure 1).

AltioLive is a hybrid of sorts, combining the best aspects of the Web (HTTP SOAP) with the strength and flexibility of a “Windows-like” fat and intelligent client. It's a change from the current Web architecture, in which the browser spends a good percentage of its time rendering HTML while the server makes the business flow decisions. In fact, some analysts see a trend toward more processing performed on the client side as the next wave in Internet technology. This presentation technology provides interesting possibilities when coupled with emerging Web service standards.

Licensing

Entry-level licensing starts at under \$1,000 and a special developer license is available. The preview edition with a limited trial license can be downloaded from the Altio Web site at www.altio.com. General availability of AltioLive 3.0 is slated for 4Q 2002. ©

Richard Hale Shaw Group

www.richardhaleshawgroup.com

Eclipsing .NET

A Free .NET IDE



You've heard the hype about .NET. You've read a couple of vague articles about dynamic discovery and invocation, service-oriented architecture, and how SOAP and a handful of other XML standards are forever changing the software industry. These ideas have intrigued you and you're interested in learning more – or at the very least, you recognize the importance of being able to add these acronyms to your resumé. In either case, you want to explore the world of .NET, but are unable or unwilling to fork over a thousand bucks for Microsoft's Visual Studio .NET product. This article is for you.

TABLE 1: System requirements

System Requirement	Minimum Value	Recommended Value
Processor speed	P 133MHZ	PII 266MHZ
RAM	128MB	256MB
Hard drive space	400MB	500MB (with 10% free disk space)

The Eclipse Project

Visual Studio .NET is not the only option for exploring .NET in a commercial-quality integrated development environment (IDE); it's just the most expensive one. On November 5, 2001, IBM released the Eclipse project (with an estimated value of \$40 million worth of software) to the open-source community. The project is now maintained by the Eclipse organization (www.eclipse.org). This open-source software provides a core, extensible infrastructure for building software development environments. It also provides a basic user interface and an open API that supports the extension of the product through its plug-in mechanism. Eclipse provides a framework for producing customized software development environments that can either be used internally by an organization or repackaged and sold commercially. In fact, the Eclipse framework is the foundation for IBM's new suite of WebSphere Workbench tools.

What does all of this have to do with .NET development? When you combine Eclipse (an open-source IDE) with Microsoft's .NET SDK (a free download) and the C# plug-in for Eclipse provided by Improve Technologies (also open-source), you have a comprehensive .NET application development platform that costs no money, and uses less than 400MB of hard disk space!

This article will walk you through the process of setting up a free C# .NET development environment on a Windows 2000 machine (the recommended platform) and deploying a simple C# application. This combination of tools will also work on Windows 95/98, XP, and NT 4.0 (the .NET SDK currently limits you to a Windows OS).

Environment Setup

Before starting, check Table 1 to be sure your system has the appropriate resources for this configuration.

Although the minimum and recommended values listed in Table 1 are accurate, you'll be much happier if you have more horsepower available. I'm running this configuration on a P4 1.6GHZ Dell Inspiron with 512MB and plenty of free hard drive space.

Assuming your system has the appropriate resources, you will need to download two components:

1. **The Eclipse 2.0 IDE** (www.eclipse.org/downloads/index.php): Windows OS.
2. **The .NET Framework SDK** (<http://msdn.microsoft.com/netframework/downloads/howtoget.asp>): Be sure to get the SDK, not just the runtime framework. You can download the SDK as a single 130MB file (setup.exe), or ten 13MB files and a batch file (setup.bat).

Once you've downloaded Eclipse and the .NET SDK, you can install the software. To install Eclipse, simply extract the Zip file in the parent directory where you would like to store



Author Bio
Kyle Gabhart is a senior mentor for LearningPatterns, a dynamic knowledge company providing world-class mentoring, training, and consulting to clients all over the world. Kyle is a popular national speaker, and a prolific writer, with more than a dozen technical articles and books to his name. You can find Kyle on the Web at www.gabhart.com.
KYLE@GABHART.COM

Eclipse (it will create a new directory called "Eclipse"). To install the .NET SDK, run the setup file and follow the prompts to install and register the appropriate components. (You may be prompted to first install Data Access Components 2.7 and Microsoft Internet Information Services, but these components are only needed for ASP.NET development.)

Eclipse Primer

The Eclipse IDE (shown in Figure 1) is divided into two basic levels: the Workspace and the Workbench:

- **Workspace:** Files, packages, projects, source control connections
- **Workbench:** Editors, views, and perspectives

This division isn't readily apparent, and isn't labeled within the IDE, but it is an important distinction to understand when working with Eclipse or an Eclipse-based tool.

The Workspace is very team-centric, focusing on projects and file resources as well as providing integrated source-control capabilities for these resources, such as CVS or Rational's ClearCase. The Workspace is also somewhat transient in that it reflects the current state of the local projects, source, and compiled files that are in active memory. The Workspace also consists of currently active Workbench elements (explained below). When you close Eclipse, it saves the current state of the local Workspace, allowing developers to restart Eclipse and pick up right where they left off.

The Workbench consists of the visual artifacts that sit atop the Workspace resources and provide distinct views and role-based perspectives of the underlying projects and resources. The user interface paradigm is based on editors, views, and perspectives:

- **Editor:** An editor is a component that allows a developer to interact with and modify the contents of a file (source code, XML file, properties file, etc.). This is the standard text area where application source code is modified within an IDE. Figure 2 is a screen shot of the C# editor, which we will install later.
- **View:** Views provide metadata about the currently selected resource. This data is pertinent to, but not directly related to, the actual contents of the resource (the contents would be accessed via an editor). The outline of a

resource's contents; the status of compilation, resource, and element properties; and the organization of a resource within a package or project are all examples of views within Eclipse.

- **Perspective:** A perspective represents a configuration of related editors and views, as well as customized menu options and compile settings. Eclipse comes with three standard perspectives (CVS, Install/Update, and Resource), and the ability to plug in additional ones as needed. Switching between perspectives brings up different editors, views, and menu options. Perspectives are either tailored to a particular task, an application development role, or both.

Installing the C# Plugin

To install the Improve Technologies C# Plug-in, we will use the built-in update feature, the Eclipse update manager. To do so, you will need Internet connectivity, and to follow these steps:

1. Open Eclipse by running `eclipse.exe` in the base of the `eclipse` directory.
2. Open the Update Manager perspective (Help —> Software Updates —> Update Manager).
3. In the "Feature Updates" view (bottom left), right-click in the view window and select: New —> Site Bookmark...
4. Fill the Name field with "Improve's Eclipse Plugin Site", and the URL with "www.improve-technologies.com/alpha/updates/site.xml". This XML file describes the plug-ins available for download from Improve Technologies, and will allow you access to those plug-ins now, and in the future.
5. A new bookmark, "Improve's Eclipse Plug-in Site," has been added. Expand it until you find the newest Improve C# plugin feature ("C Sharp Feature x.x.x"). If you select the feature, its description will be displayed in the "Preview" view. You can read the license agreement by clicking the link.
6. In order to install the feature, click the "Install" button in the "Preview" view: click "Next" for each page; and then "Finish". On the last page ("Feature Verification"), click on the "Install" button. Eclipse will automatically install the feature, and you will be asked whether you want Eclipse to reboot or not. Say yes.
7. Once Eclipse has rebooted, you can check

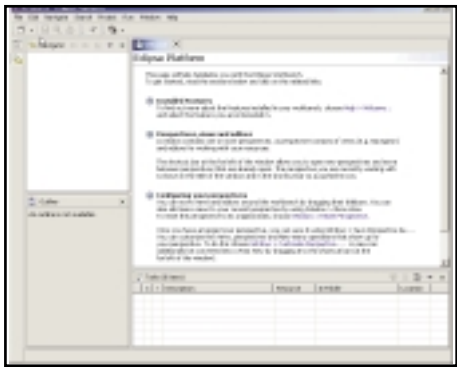


FIGURE 1 | The Eclipse IDE

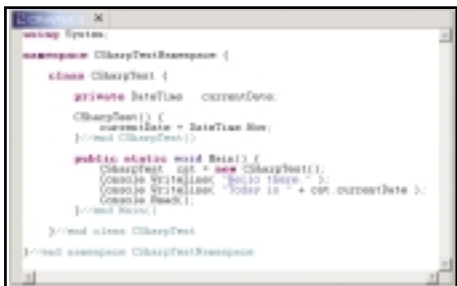


FIGURE 2 | The C# editor

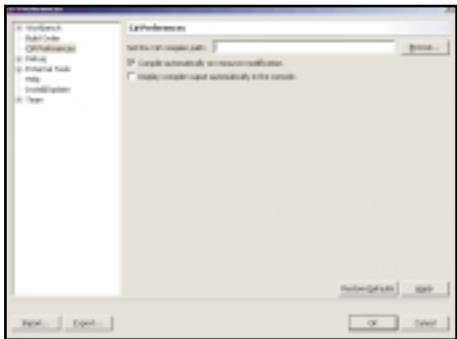


FIGURE 3 | C# Preferences

that the feature has been installed: expand the item "Current Configuration" (in the "Install Configuration" view), and you will find the feature in the list of the installed features. If you expand "Configuration History", you will see all the previous configurations. You can save a specific configuration so you can restore it later: right-click on a configuration and click "save".

8. The last step is to configure the new Eclipse C# perspective to use the .NET C# compiler that was installed with the .NET SDK earlier. To configure the C# compiler, navigate to the EclipseWorkbench preferences (Window —>

Preferences) and select the C# Preferences item. This preferences window should now look similar to Figure 3. To specify the C# compiler (csc.exe), click the "Browse" button and navigate to the Microsoft .NET/Framework directory. On Windows 2000, that directory is C:\WINNT\Microsoft.NET\Framework\v1.0.3705\csc.exe. (The name of the directory under the Framework directory may vary depending upon what version of the .NET SDK you have installed.)

Sample C# .NET Application

Now we'll walk through the creation of an Eclipse project containing a single C# program. We'll verify our success by executing the completed program, and finally, we'll touch on a few of the available application development tools (keyword content assistance, task view usage, and file-level compiler arguments) that come with Eclipse and the C# plug-in.

To begin, we'll open a new project. As with any good tool, there are several ways to do this. You can click the "New Wizard" button directly (leftmost button below the menu bar), select the New Wizard drop-down (the drop-down arrow on right-hand side of New Wizard button), or use the menu bar (FileNewProject). The New Project window should now be open. Select "Simple" on the left and "Project" on the right. Click the "Next" button, assign a unique name for the project, and then click the "Finish" button.

Next, use the File menu or the New Wizard to create a new C# file. If you use the New Wizard drop-down or File menu, select the

"Other" option (this is the equivalent of clicking the New Wizard button directly). Whichever path you take, the New window will provide two categories: C# and Simple. Select the "C#" category on the left and "C# File" on the right. Click the "Next" button and provide a name for the C# file. We'll use "CSharpTest.cs" for this example ("new_file.cs" is provided by default). Also, be sure that the correct project name is provided in the Container field. Finally, click the "Finish" button.

Now it's time to actually write the program. Enter the code displayed in Listing 1. When run, this code constructs a simple object with a single instance field of type DateTime containing a value corresponding to the current system time and date. The program output consists of a brief text greeting followed by the current time and date.

To save the file, click the disk icon at the top, use the menu system (FileSave), or hit Ctrl-S. When you save, Eclipse will also attempt to automatically compile the program (unless you turn this option off in the C# Preferences window). If there are any compilation problems, they will be displayed in the task view below the C# editor. Once you have resolved those issues, you are ready to execute the application. To do this, simply double-click on the CSharpTest.exe file that appears under the project you created. The result should be two lines of text. The first line displays a simple greeting and the second displays the time and date. To end the program, simply hit the return key.

Finally, there are a couple of features in Eclipse that you may find useful:

- **Compilation settings:** Eclipse stores file-level compiler settings, including compiler arguments, output file type, and the specification of required .NET modules.
- **Content assistance:** Hitting Ctrl-Space activates the C# plugin's content assistance dictionary. This dictionary is currently limited to keywords only (version 2.0.1).
- **Task view:** In addition to displaying compiler problems, the task view can be manually manipulated to serve as a way of tracking progress, and recording project notes. You can create tasks for yourself, assign priorities to your tasks, check them off when complete, and filter them to display only certain kinds of tasks. Think of it as a virtual sticky pad that is associated with your Eclipse Workspace.

Conclusion

Web services are hot! (That's why you bought this magazine.) The .NET platform offers some intriguing possibilities for the development of XML Web services, and it's a good idea to explore it now and become familiar with the basic constructs. Unfortunately, the price tag on Visual Studio .NET is high, and setting up Visual Studio is a pain in the neck. The Eclipse Workbench, together with Microsoft's .NET SDK and Improve Technologies' C# Plugin, provides a powerful combination of tools that allows you to experiment with C# .NET application development, using a full-powered IDE – without paying a penny. Enjoy! ☺

Listing 1

```
using System;

namespace CSharpTestNamespace {

    class CSharpTest {

        private DateTime currentDate;

        CSharpTest() {
            currentDate = DateTime.Now;
        } //end CSharpTest()
    }
}
```

```
public static void Main() {
    CSharpTest cst = new CSharpTest();
    Console.WriteLine( "Hello there." );
    Console.WriteLine( "Today is " + cst.currentDate );
    Console.Read();
} //end Main()

} //end class CSharpTest

} //end namespace CSharpTestNamespace
```

Download the code at
sys-con.com/webservices

Macromedia

www.macromedia.com/go/CFMXlight

Web Services Orchestration

The marriage of Web services and workflow

AUTHOR BIOS:



Ron Ben-Natan, CTO of ViryaNet, Inc., holds a PhD in computer science in the field of distributed computing and has been architecting and developing distributed applications for more than 15

years. Ron's hobby is writing about how technology is used to solve real problems; he has authored numerous books, including IBM WebSphere Application Server: The Complete Reference (Osborne/McGraw-Hill).

RBENNATA@HOTMAIL.COM



Doron Sherman, CTO of Collaxa, holds a BS in math and physics, and an MS in computer science from the Hebrew University in Jerusalem, Israel. Before joining Collaxa, Doron was a chief

scientist and cofounder of NetDynamics, the startup that pioneered the Java application server in 1995 and led its market until its merger with Sun Microsystems in August 1998.

DORON@COLLAXA.COM

With the advent of Web services, workflow vendors and enterprise application integration (EAI) vendors are aligning themselves and often reinventing themselves to make full use of Web services and the inherent strengths of the asynchronous, loosely coupled software model. While Web services are powerful in and of themselves, the combination of Web services with a process-based approach is even stronger. This marriage of workflow with Web services is often termed *Web services orchestration*.

Orchestration is a relatively new term, but it's already being used in differing contexts. While Web service orchestration is usually used consistently at the 30,000-ft level, when you look at the implementation details, you find that there are meaningful differences in terms of product capabilities and even in philosophy. At this point, there's plenty of confusion surrounding the terminology and we're seeing the emergence of an acronym soup. This is demonstrated by commonly raised questions, such as, "Is EAI the same as workflow?"

This article provides a broad market sur-

vey and introduction to the major categories being described as Web service orchestration systems. The taxonomy we present will range from categories that are loosely related to Web services to those that depend heavily on Web services.

Workflow Systems

Many systems are based on process/workflow engines. It is readily accepted that workflow management is an important ingredient in flexible integration. The strength of workflow systems is that they abstract the essence of the application flows into a set of processes that can be easily modified to account for differences in business processes. Because these flows usually touch many systems, the management of the workflow has always been closely tied to the integration of the systems involved in the business process (see "XML Glue," *XML-Journal*, Vol. 3, issue 3).

Workflow is not a new concept. It has been around for over 20 years and has been well formalized by the Workflow Management Coalition (WfMC). The following definitions appear in the WfMC glossary:

- **Business process:** A set of one or more linked procedures or activities that collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships
- **Workflow:** The automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action according to a set of procedural rules
- **Workflow management system:** A system that defines, creates, and manages the execution of workflows through the use of software running on one or more workflow engines, and that is able to interpret the process definition, interact with workflow participants, and, where required, invoke appropriate IT tools and applications

A workflow management system needs to support three types of features:

- **Build:** A workflow management system needs to provide tools with which a business analyst can design and assemble a

Gartner

www.gartner.com/us/eai

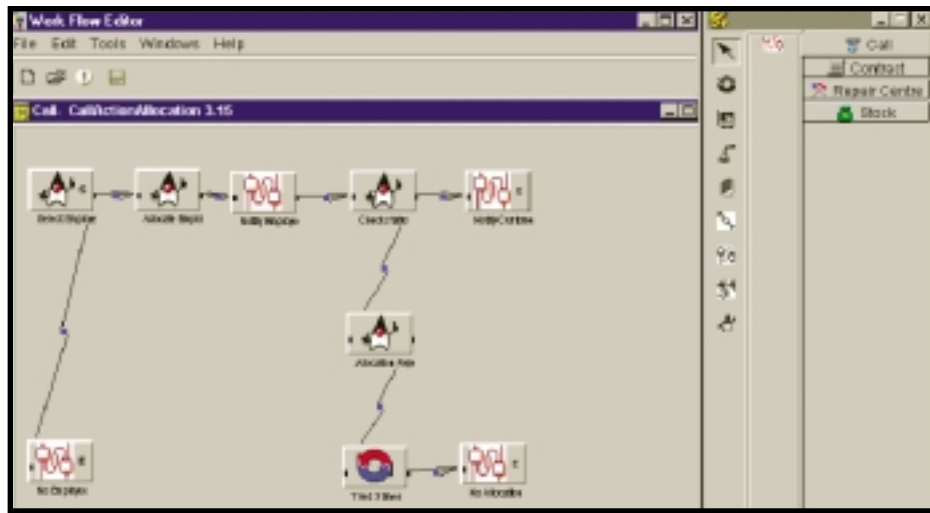


FIGURE 1 | ViryaNet process editor

process definition comprising the activities that need to be performed, the roles that participate, and the flow between the activities based on a set of business rules. Figure 1 shows the process editor in use when deploying systems built upon ViryaNet's Workflow Engine.

- **Process engine:** A workflow management system must include a runtime engine that manages the processes and walks each process through the set of activities specified by the process definition. The engine needs to perform the actions and evaluate the business rules to determine how the flow proceeds.
- **Runtime user interface:** A workflow management system must include user interfaces through which people interact with the system. The workflows modeling the business processes almost always include a set of activities that require human intervention; the system must therefore manage a set of screens to allow this. In addition, the process engine should have a user interface component to allow users to monitor and control the engine and its processes.

While workflow systems are not directly related to Web services, they are receiving a facelift due to the promise of Web services. The marriage between workflow engines and Web services is made by introducing a new type of step into a process (a step in which a Web service is called), and by allowing each process to be created or updated through a Web service. The orchestration capability is inherently provided by the workflow engine that manages the transitions within the state machine defined by a process.

Enterprise Application Integration (EAI)

EAI has developed from an evolution of approaches over the past two decades to deliver on the promise of enabling data sharing across applications and processes. Initially, database vendors pushed the enterprise data model and federated databases as a way to pull together data from diverse applications. Later on, ERP vendors focused on using a monolithic homogeneous system for integrating back-office functionality, such as manufacturing processes, human resources, and financial processes.

Recently, message-oriented middleware (MOM) has emerged as a means to route messages among diverse systems. MOM has now evolved into integration broker product suites (AKA EAI) to unify heterogeneous applications. Gartner estimates that EAI can reduce integration costs associated with implementing a packaged application by one third, and the cost of maintaining and making changes to such applications by two thirds.

EAI software suites are still evolving to address the complex requirements of application integration. These software suites include:

- **An integration broker:** A set of services for message transformation and smart (e.g., content-based) routing
- **Application adapters:** Off-the-shelf adapters for common applications (e.g., PeopleSoft or SAP R/3)
- **Development tools:** A set of tools for specifying the transformation and routing rules to be executed by the broker
- **Adapter tools:** A set of tools for building adapters into legacy applications

- **Administration tools:** A set of tools for monitoring, security, etc.

Most EAI product suites are vertically integrated on top of MOM infrastructures and in recent years have added products to their suites, which focus on higher value-add functionality. These products include business process management (BPM), enterprise portals, B2B connectivity, and even e-commerce solutions (e.g., management of trading partners).

The advent of Web services hasn't been lost on EAI vendors, who see Web services as an opportunity to increase interoperability with other EAI solutions and streamline the complex and expensive task of creating custom connectors to legacy systems. Increasingly, these vendors are adding Web services support as well as business activity monitoring capabilities.

Web services play a particularly important role in extending the integration capabilities of the BPM tool within the EAI suite. They enable the EAI solution to expose business processes as Web services and to create new business processes from Web services. The BPM tool has the added capability of orchestrating business processes that incorporate Web services and coordinating Web services along with legacy systems and human-based workflow. In that view, orchestrating Web services within the context of a BPM tool as part of an EAI suite is not materially different from such orchestration within the context of workflow systems.

BizTalk Orchestration and XLANG

While BizTalk orchestration may be thought to fall into the same category as EAI-centric orchestration, Microsoft has made BizTalk orchestration, as well as Web services, so central to their architecture that this solution warrants its own category. BizTalk provides two core functions: EAI and orchestration services. Orchestration is not viewed as a subelement within the EAI facilities, and it supports advanced workflow features. In fact, we believe BizTalk orchestration is one of the best workflow engines out there today.

A BizTalk Server orchestration is a process created in the BizTalk Orchestration Designer, which is based on Visio, as shown in Figure 2. The process is serialized into an XLANG document – XLANG being an XML language that describes the process flow. Implementation services orchestrated by such a process can

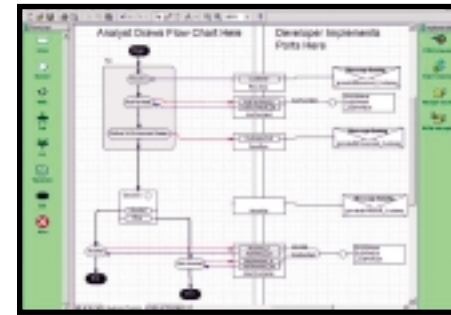


FIGURE 2 | BizTalk Orchestration Designer

be any .NET component. Because the use of Web services is so prominent in VS.NET, ASP.NET, and in the .NET Framework, the use of BizTalk orchestration to compose Web service calls is becoming a dominant theme within the BizTalk community. Becoming – but not quite there yet. BizTalk orchestration provides a number of implementation shapes that are composed into a process, including COM components, script components, message queuing, and BizTalk messaging. While Web service calls may be wrapped within one of these components, there is no native Web services shape in BizTalk orchestration yet.

The next version of XLANG, called XLANG+, will have Web services as a native shape. Until then, the simplest alternative is to implement a COM or .NET component that invokes a Web service using SOAP, something that is very easy to do within the Microsoft tools.

IBM Web Services Flow Language

The Web Services Flow Language (WSFL) is an XML-based language used for describing the composition of Web services. A flow composition (also called orchestration or choreography of Web services) defines the execution sequence of the functionality provided by the composed Web services. WSFL fits neatly into the Web services stack above the Web Services Description Language (WSDL) and supports a recursive model by which orchestrations defined using WSFL can themselves be externalized as new Web services to support business processes hierarchies.

WSFL is an IBM specification, in many ways very similar to Microsoft's XLANG. In fact, a Gartner research note in May 2001 predicted that IBM and Microsoft would jointly submit a proposal to the W3C to combine XLANG and WSFL by year-end 2001. While this has not yet come to pass, WSFL is receiv-

ing a lot of industry backing. However, it's not getting the same backing from IBM that XLANG is getting from Microsoft. As an example, IBM offers the Web Services Process Management Toolkit at www.alphaworks.ibm.com/tech/wspmt. This is a toolkit for composing Web services, including them in a business process, and implementing them as business processes. It is based on MQSeries Workflow and uses the Flow Definition Language (FDL) instead.

Web Service Orchestration Server

Web services provide a number of benefits and additional capabilities to EAI tools that address the high-end segment of the business process integration problem. However, the adoption of Web services carries a bigger promise, which is to fundamentally change the economics of integration. The promise of Web services is that companies will be able to implement business processes cheaper and faster, align their integration efforts with internal application development – thereby utilizing development skills more effectively, and avoid duplication and operational overhead of their computing infrastructure.

The clean-slate approach to Web services is based on the realization that making Web services work is a two-step process. First you publish them, i.e., make the services available, and then you orchestrate them, i.e., coordinate the execution of multiple Web services into business processes. Orchestrating synchronous and asynchronous Web services into a collaborative or transactional business process is complex. The emergence of services as building blocks for delivering process-centric applications introduces new challenges throughout the development life cycle.

In particular, the synchronous request-reply programming model using fine-grained components is now giving way to a more flexible and reliable model called orchestration, based on asynchronous and nonlinear multistep interactions across loosely coupled service components. Orchestration entails a consistent set of infrastructure-level requirements that can be grouped into three major categories:

- **Coordination:** Asynchronous conversations, parallel processing, event handling, transaction management, clustering, and scalability

- **Management:** Administration, cancellation and change management, exception and timeout handling, version control
- **Activity monitoring:** Business reporting, audit trailing, nonrepudiation

A Web Service Orchestration Server (WSOS) is a built-from-scratch piece of infrastructure software that reduces complexity by encapsulating the facilities needed for executing the orchestration's technical requirements.

Orchestration within the context of a WSOS differs from that within the context of EAI. WSOS is built from the ground up to extend the computing architecture defined by J2EE and XML Web services standards, while EAI takes the tools approach.

Put together, these innovations change the economics of integration by dramatically reducing the complexity, skill requirements, and total cost of integration relative to traditional EAI solutions. Similar to how J2EE and application servers emerged to deliver and manage fine-grained, tightly coupled Web applications, Web services and orchestration servers are now emerging to address the challenge of delivering and managing loosely coupled service-oriented business processes.

Summary

There is evidence that Web services impact the various approaches to integration and promote a continuing shift toward delivery of process-centric applications built from loosely coupled service components. This new paradigm induces a shift toward a service-oriented architecture in which orchestration becomes the primary concept for coordinating the execution of services for delivery of collaborative and transactional business processes. Web services promises to fundamentally change the economics of integration and reduce its perplexing complexity. The extent to which orchestration systems succeed in delivering on that promise and grab the mind share of the developer community, will ultimately determine the leaders in the rapidly evolving integration space.

Resources

Gartner. (2001). "Microsoft Continues Web Service Leadership With New XML Specs." www3.gartner.com/resources/98300/98366/98366.pdf ©

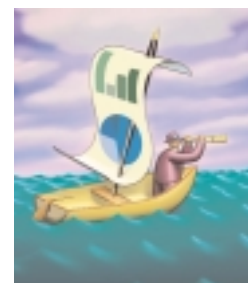
Chart Your Course to Success in IT...

...order your copy of

Java Trends: 2003

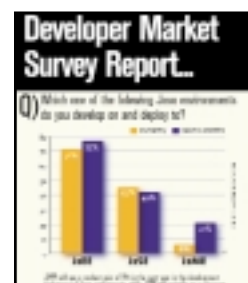


Don't go astray. In the vast sea of Internet technology, market conditions change constantly. Will Java remain the hot platform it is today? Will C# rapidly gain ground? What are the world's foremost Java developers aiming their sights toward? Which companies come to mind when planning their next project? How can their thinking direct your efforts?



EnginData Research has studied the IT industry extensively, spotting many key trends and alerting industry leaders along the way. In the first quarter of 2002, they embarked on their most challenging mission ever: the most in-depth study ever done of the Java development marketplace.

After months of analyzing and cross-referencing more than 10,500 comprehensive questionnaires, the results are now available.



Here's just a sample of the critical data points the Java report delivers...

- ✓ Current and projected usage of languages and platforms
- ✓ Multiple rankings of hundreds of software vendors and tools
- ✓ Types of applications being developed
- ✓ Databases being used
- ✓ Purchasing patterns
- ✓ Company size
- ✓ Development and purchasing timelines
- ✓ Perceptions and usage of Web services
- ✓ Preferred Java IDE
- ✓ J2EE, J2ME, J2SE usage comparisons

As an IT specialist, **EnginData Research** focuses exclusively on three leading drivers in IT today – Java, Web services, and wireless development. Coming soon, our Web services survey will deliver such critical knowledge as:

- ✓ Time frame for developing Web services – enabled apps
- ✓ Percentage of apps with Web services today
- ✓ Sourcing and hosting Web services
- ✓ Perceived leaders in Web services tools and solutions

Navigate your company through the challenging IT marketplace with the trusted knowledge and intelligence of **EnginData Research**. Order your copy of the 2002–2003 Java market study by calling Margie Downs at 201-802-3082, or visiting our Web site.



www.engindata.com

EnginData's research is invaluable to leading IT vendors, integrators, Fortune 500 companies, trade-show organizers, publishers, and e-business organizations worldwide.



Navigating Web Services

A checklist for success



Web services promises to finally provide a universal mechanism for connecting loosely coupled systems. If the dream bears fruit, it will represent a huge advance in enterprise computing, facilitating distributed, transaction-centric collaboration in an inexpensive, quick, and reliable manner.

As with every “next big thing,” the combination of analyst buzz, Silicon Valley expectations, and incessant media attention has led to a manic state of frenzy. A heady mix of confusion is rarely a good thing and can derail businesses and get CIOs fired. Fast.

To date, we face several issues that may lead to the implementation failure or even premature demise of Web services. As all of this unfolds, technology professionals will need a game plan to separate the hype from reality.

The Premise and Promise of Web Services

Web services is neither a single technology nor a new concept. Instead, it is a portfolio of methodologies and standards that extend and build on Internet technologies. Historically, loosely cou-

pled, highly distributed systems have enjoyed marginal success due to the very complexity required to design such systems, as well as the sheer cost of supporting private networks and proprietary interfaces. The premise – and the promise – of Web services is that the use of open and shared interfaces, e.g., WSDL, atop the Internet will make it easier and cheaper to connect, integrate, and collaborate.

While the Internet is a mature networking platform, higher-level protocols, such as XML, are still relatively new. Moreover, the standards that use XML to specify Web services interfaces are at the very first stages of their life cycle. The ensuing state of flux results in a number of issues that an aspiring adopter must consider.

Competing Platforms

Two major platforms compete for your Web services business: .NET and J2EE. Platform choices and the closely associated tool(s) selection are significant decisions that go far beyond the scope of this article. It should be noted, however, that Web services implementations are eventually constrained by the underlying platform, which is of particular concern in high-volume transaction environments.

Maturity

Several important functional components of the XML substructure for Web services are either just under development or in their early stages of release (see www.w3.org/2000/03/29-XML-protocol-matrix). Moreover, fundamental Web services specifications and interfaces are also immature and fall short in terms of functional completeness, e.g., robust security and authentication mechanisms.

The lack of maturity in these core standards currently forces adopters to either do without certain functionalities or pursue proprietary extensions. While the former can compromise the completeness of a solution, the latter generally impedes interoperability and extensibility.

Right of Ownership

Although the majority of Web services protocols are open, they are not unencumbered intellectual property. Some major computer companies today hold rights to various pieces of SOAP, WSDL, and UDDI, and seem intent on capitalizing on their IP through the “reasonable and nondiscriminatory” (RAND) licensing framework (see www.w3.org). This may have costly implications to users and fundamentally undermine the adoption of Web services.

Tackling Web Services

So what can a CIO and IT staff do while these issues evolve? Following are several practical tips on how IT professionals can educate themselves and test the premises and promises of Web services – a do’s and don’ts guide:

- **Evaluate:** Don’t adopt for adoption’s sake and prematurely lock into a platform. The future of Web services is just unfolding and

interoperability between different platforms is a possibility.

- **Play:** Be an early adopter and encourage staff to explore Web services to identify both promises and pitfalls. In some contexts, Web services have been successfully and cost-effectively used to solve integration problems.
- **Assess:** Different requirements from both an IT and a business (process) perspective may favor different architectures and platforms. Demand citations and reference implementations and question slideware.
- **Discuss:** Put technology partners to task. Begin discussing the long-term requirements of Web services with your IT providers and consultants to develop a strategic roadmap. Talk to your (prospective) business partners to assess needs and requirements.
- **Scope:** Scope the impact of Web services. The component-based, granular nature of Web services design and deployment will undoubtedly require a close look at business process design and best practices. Be aware, proactive, and careful.
- **Participate:** Companies should at the very least follow standards developments very closely or, even better, participate in the formulation and fine-tuning of standards.
- **Relax:** Withstand the hype. Even if Web services eventually leads to nirvana, it’s going to take time. Widespread interoperability and full-fledged UDDI instances precipitating the transformation of business are years away.

Only time will tell if Web services will live up to the hype and forever change the way business is conducted – creating new internal efficiencies, business opportunities, and better ways to serve customers. Platform maturity and other issues will undoubtedly significantly impact the general adoption of Web services as well as enterprise-specific implementations. Keeping these practical guidelines at the top of your thinking will help you better navigate the preliminary stages of the Web services maze. ☺



Author Bio
Bernhard Borges is a managing director at PwC Consulting and leads the Advanced Technology Group, a part of the global Information Technology Solutions group.
BERNHARD.BORGES@US.PWCGLOBAL.COM

THE WORLD'S LEADING INDEPENDENT WEBLOGIC DEVELOPER RESOURCE

Helping you enable intercompany collaboration on a global scale

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

SPECIAL OFFER!
SAVE \$31*
OFFER EXPIRES NOV 30, 2002

Now in more than 5,000 bookstores worldwide – Subscribe **NOW!**

Go Online & Subscribe Today!

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



SYS-CON MEDIA

WebLogicDevelopersJournal.com

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

The Tools Landscape

Web services technology matures



Web services products have matured rapidly over the past 12 months, to the point where it's become acceptable to utilize the technology in major projects. However, though improving, the currently available toolkits can't be considered complete. This article surveys the rapidly evolving development tools landscape and addresses what tools a fully featured environment should provide to the Web-services developer.

AUTHOR BIO:



Neil O'Toole is a technology evangelist at Cape Clear, where he is responsible for promoting the adoption and effective use of Web services by developers. He manages the CapeScience developer

network (www.capescience.com), moderates the Cape Clear newsgroup, and works with the Web services development community to help define Cape Clear product direction.

NEILOTOOLE@YAHOO.COM

IDE Integration

Historically, developers have toiled with an array of utilities, editors, and command-line tools. The ability to effectively navigate this maze of programs is a source of pride to many of them; however, this maze introduces a steep learning curve and, in the long run, lowers productivity. Some of the costs associated with such a nonintegrated approach are time lost switching back and forth between programs, time required to learn a new user interface, and cost of owning and managing multiple applications. It's also been costly for the tools vendors; each has written many thousands of lines of code that could have been shared between implementations. This includes code for the user interface, file and project management, version control, and many other house-keeping tasks.

The past year has seen major changes in the development landscape. Proprietary environments have fast lost ground to the concept of a single development framework into which additional tools can be deployed

via a plug-in architecture. Products like JBuilder and NetBeans have provided this type of framework for some time, but it's fair to say that an industry-wide paradigm shift has coincided with the arrival of two uber-frameworks: Microsoft's Visual Studio .NET and the open-source Eclipse project.

IBM's reasons for donating Eclipse to the open-source community are no mystery. If the non-Microsoft community doesn't close ranks around a single framework, it will not be economically feasible to compete with the de facto IDE monopoly for Windows development: Visual Studio. Developing an IDE is an expensive business – IBM spent \$40 million on Eclipse. This may seem a costly bit of software to give away, but having the developer community adopt its technology gives IBM a significant head start since IBM's tools already run in Eclipse, while other vendors must go to considerable lengths to move to a plug-in architecture. Tools vendors may not wish to swallow this bait, but the drive toward a single IDE platform is given unassailable momentum

by two converging forces: the developer-side desire for an integrated experience, and the economics of proprietary development.

One of the consequences of IDE integration is that products built on the same framework will largely look the same. The mainstream toolsets will also have significant overlap in functionality, though vendors will try hard to differentiate their offerings. This commoditization of the tools market is bad news for smaller vendors. In a battle of commodities, the larger, better-resourced players tend to win. It's not all gloom though. As vendors strive to outshine the competition, we can expect to see higher quality and more innovative software. Another plus: open-source groups and creators of niche or boutique software can take advantage of the IDE frameworks to deliver their software in a first-rate package. Although developers will initially lose some choice due to the inevitable process of consolidation in the tools market, this will be more than offset by improved product sets from surviving vendors and a more diverse



Don't Delay!
Subscribe
for **FREE!**

at www.sys-con.com

THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE **FREE** E-Newsletters SIGN UP TODAY!

Go to www.SYS-CON.com

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.



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

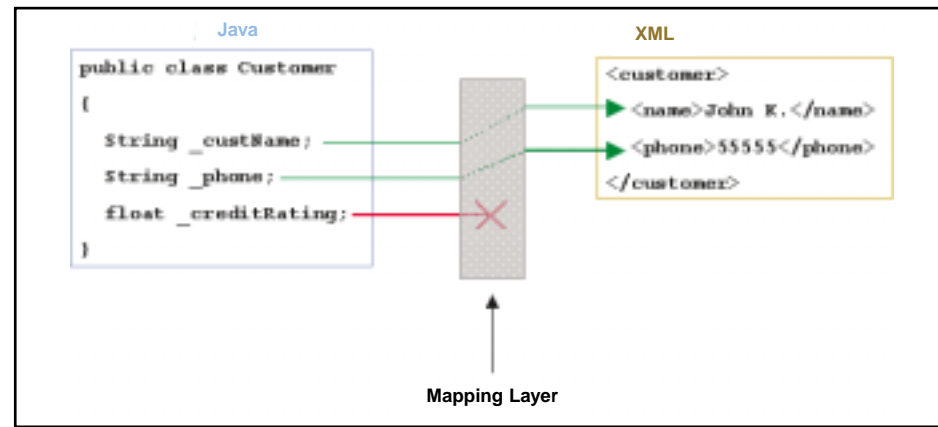


FIGURE 1 | WSDL mapping layer

range of software from smaller software houses. Web-services developers in particular stand to gain, as there are a great number of companies targeting this space.

Basic Tools

Before we survey some of the more interesting tools that are becoming available, let's establish the basic functionality required to develop Web services. From within the IDE, the developer should be able to:

- **Generate Web Services Description Language (WSDL) from a component:** This should be as simple as right-clicking com.acme.MyClass and selecting "Generate WSDL...". This command should bring up a dialog that allows the developer to specify which methods to expose, what to name the Web service, and other basic options.
- **Generate stubs and skeletons from WSDL:** It should be possible to generate both client-side stubs and server-side skeletons, preferably in a variety of languages. For Java, the developer should have the option of generating EJB or regular Java skeletons.
- **Deploy a Web service:** Deployment can be an intricate process, so the IDE should provide a wizard to help assemble the classes, documents, and configuration data necessary to successfully deploy. Several products have introduced the concept of a Web Services Archive, similar to a JAR or WAR file. This can make it considerably easier to manage deployment, especially across multiple machines.

Note that the degree to which an IDE exposes WSDL can vary. In Visual Studio, the developer adds WSDL to a project by invoking the "Add Web Reference..." command. This automatically generates client proxy code and adds it to the project. The WSDL is effectively hidden from

the developer. Hiding the complexity of WSDL is fine if your application is a consumer of Web services, but server-side development requires more control. This is particularly the case with Java because Web services haven't been tightly integrated into the JVM as Microsoft has done with the .NET Common Language Runtime (CLR). We might expect the toolset to grant the developer considerable control over the WSDL. In particular, it's important to be able to customize the mapping between WSDL and the native language.

WSDL Mapping

When WSDL is generated, the created schema types are typically named after classes, with elements named after the members of the class. Frequently, the developer wants to rename or omit a member, or otherwise change how the class is serialized. Why? Perhaps the Web service has to conform to a specific WSDL interface defined by a standards organization or commercial partner. Or it might be for more cosmetic reasons, such as changing unhelpful element names; or for business reasons, such as not exposing sensitive information.

To serialize an object, a Web services platform uses a mapping layer that defines how the native representation (e.g., a Java class) is mapped to XML (see Figure 1). The method and degree of control over this layer varies by product, but there are three common ways to provide customization:

- **Configuration file:** This will usually allow fairly limited customization, such as changing member names or omitting elements.
- **XSLT:** The native object is first serialized per the default rules, and then an XSLT transform is performed, allowing a

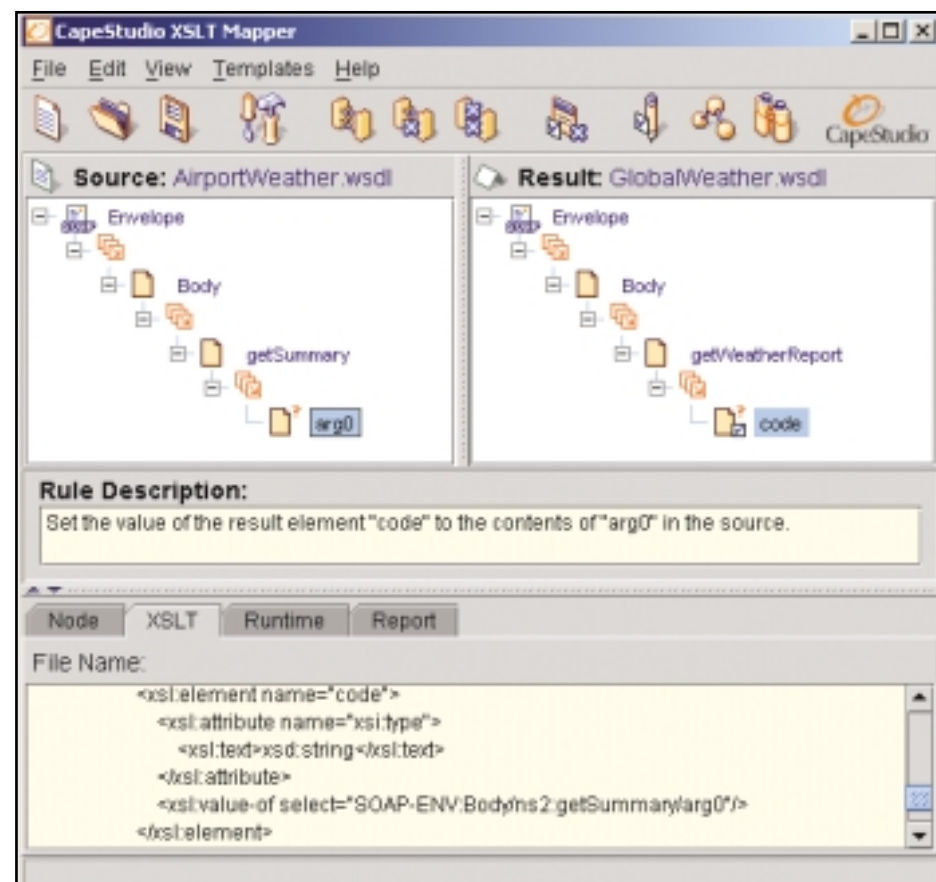


FIGURE 2 | Graphical XSLT Mapper

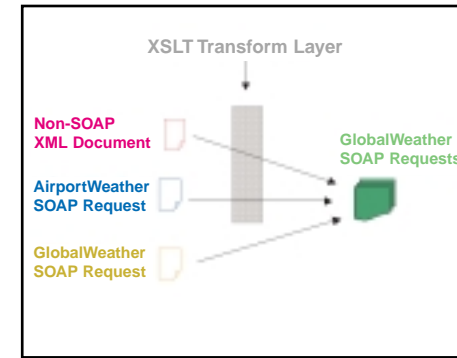


FIGURE 3 | Using XSLT with SOAP

greater control. However, developers can find XSLT difficult to work with, and the transform is processor intensive.

- **Custom serialization class:** Instead of using reflection to inspect and serialize the object, the serialization process can be delegated to a helper class created by the developer. This mechanism provides almost unlimited control, although modifying and managing the additional classes is cumbersome.

Most platforms support the use of at least one method of directly editing the mapping layer. However, tools support is still weak. In the near future, developers can expect to see software that provides full round-trip support for editing WSDL. Such an editor will allow developers to modify elements in the WSDL and will automatically propagate these changes to the mapping layer. Note that it should be possible to perform the converse operation: modify the source component, and the changes are propagated toward the WSDL. Mapping is one concept that developers will encounter frequently with Web services. This is because of an underappreciated enabling technology: XSLT.

XSLT and Graphical Mapping

XSLT is a technology that most developers haven't had significant contact with. It's often thought of in the context of generating HTML pages, but its utility extends far beyond that. As mentioned earlier, XSLT can be used to control the mapping between WSDL and a native language. Some other uses are:

- Map requests from an older version of a Web service into the new version's format
- Map non-SOAP XML documents into SOAP requests

XSLT has a steep learning curve – for

many developers the “side effect-free” programming model is less than intuitive to grasp, so good tools are essential. Thankfully XSLT lends itself quite well to graphical editing, where two schemas are laid out side by side and related elements are linked by dragging one to the other (see Figure 2).

Let's look at an example. AirportWeather is a publicly available Web service that reports on conditions at airport weather stations around the world. There's also a recently created successor Web service, GlobalWeather, which is faster and has a richer data model. Both of the services have an operation that takes one parameter, the weather station code (e.g., JFK or SFO), and returns a weather report for that location.

We can use XSLT to transform a SOAP request for AirportWeather into a SOAP request for GlobalWeather. Instead of creating the XSLT by hand, we load the source schema (AirportWeather.wsdl) and the target schema (GlobalWeather.wsdl) side by side. Then we graphically link the two parameters – AirportWeather's nonintuitively named “arg0” and GlobalWeather's “code” – and the mapper will generate the necessary XSLT. In a similar manner, we can use XSLT to transform a non-SOAP XML document into a GlobalWeather request (see Figure 3).

XSLT is an underused technology with great potential, in particular for integrating Web services and legacy XML applications. A good development environment will support the graphical creation of XSLT, and it should allow the easy deployment of XSLT documents into your Web-services platform.

Testing

Once a Web service is built and deployed, the next step is to test it. Testing can range from merely verifying that a service is operational to a full automated test suite. The most common testing approach is to generate a client proxy from the WSDL and write a test class to invoke the proxy. But there's a better way: the IDE can generate a test class which invokes the Web service's operations using default parameter values. For Java, the test cases can make use of the JUnit test framework and the Ant build system to make it easy to integrate Web service testing into existing automated test suites.

Another approach is to generate a SOAP document from the schema information in the WSDL and send it directly to the Web service endpoint using HTTP. This isn't

amenable to automated testing but is useful if the developer is interested in the lower-level details of the SOAP messages. However, the friendliest approach is to generate an HTML Web client that's deployed on the Web server alongside the Web service. This Web client is typically based on JSP or ASP and consists of one page for each operation. The form fields correspond to the operation parameters, and submission of the form results in the Web service being invoked. As well as being a fast means of verifying that a service is alive, this is also an effective way of demonstrating a developing Web service.

Debugging

Debugging a Web service is essentially the same as debugging any other server-side software, such as an EJB. The IDE's debugger attaches to the service process using the standard mechanisms for the platform, which usually involves starting the server in debug mode. But unlike an EJB, the raw messages between a client and a Web service are intelligible and of interest to the developer.

There are two quite similar ways of monitoring SOAP messages. A TCP tunnel is a program that listens on a port and directs all traffic to a specified host and port (e.g., tunnel from localhost:7000 to server.acme.com:8000). The tunnel program can then display the messages in a GUI. To enable tunneling, the SOAP endpoint URL used by the client must be modified to point at the tunnel's listening port.

Like a TCP tunnel, an HTTP proxy listens on a port, but it has specific knowledge of the HTTP protocol. The proxy reads the “Host” field in the HTTP header and directs traffic to the indicated location. A proxy is more transparent than a tunnel in that proxy settings can generally be set on a process-wide basis and thus modification of individual endpoints is not required.

Conclusion

Monitoring tools were an indispensable item in the early days of Web services, when developers spent a lot of time working with raw SOAP messages. Although a development environment should still provide the ability to work at this level, many developers will never need to do so. It's a sign of the maturity of the technology that this is the case. Web services tools are entering the mainstream. ☺

Web Services Security

The key is unification



Web services is a promising technology with the potential to greatly simplify B2B enterprise application integration. This is good news for any organization trying to provide seamless access to their business applications for their customers and partners.

Along with these increased integration opportunities, however, arises a new—and daunting—set of security challenges.

AUTHOR BIO:



Lakshmi Hanspal is a security architect with Quadrasis, a business unit of Hitachi Computer Products (America). She has several years of extensive experience in the architecture, design, and development of security and directory services, including a patent on directory service schema.

Lakshmi works with customers to provide enterprise security solutions. She also contributes to technical committees at JCP and OASIS. Lakshmi is a Sun Certified Java programmer and J2EE Architect.
SOLUTIONS@QUADRASIS.COM

What is a Web Service?

From a business perspective, a Web service is an innovative, new approach to integration. Existing applications can be reused, or new Web service applications can be deployed to promote integration – within an organization's intranet, over extended (partner) virtual private networks, or over the Internet.

From a technical perspective, a Web service is a collection of one or more related operations described by service descriptions (XML) and network accessible through standardized XML messaging. Thus, a Web service can be described, published, discovered, invoked, and integrated (with other services).

Security Challenges and Solutions

While Web services offer many integration advantages, they do present security challenges. Indeed, the absence of security seems to be the single largest obstacle to the general acceptance of Web services. Additionally, some misconceptions about what constitutes “adequate” security have

further muddled the understanding of security needs with Web services. The following are examples of common misconceptions:

- **Transport and/or message security are good enough for Web service applications.**

Security is a system-wide concern. While transport security (e.g., SSL) or message security (e.g., SOAP security, including digital signature and encryption) can play a part in providing one layer of security, protection against unauthorized access to Web services themselves, the Web services registry, and enterprise applications and their data, and support for secure single sign-on (SSO) for accessing Web services within and across business domains must also be addressed.

- **Analysts predict that until security considerations for Web services are resolved, Web services are likely to first be deployed in an enterprise's intranet.**

Though this notion is reasonable, cyber threats come from many quarters. Even employees with valid system access can abuse their privileges. According to a survey of com-

panies and government agencies commissioned in 2001 by the FBI and the Computer Security Institute, while 70% of computer attacks came from outside via the Internet, attacks from within accounted for most of the financial losses. This requires that security be addressed with equal importance in all enterprise tiers.

Let's examine some of the security challenges that plague the world of Web services, and discuss solutions. It's important to recognize that each of these solution mechanisms offers specific security services catering to certain security policy requirements. Any multitier, multidomain security solution would include a combination of one or more of these security mechanisms with additional mechanisms to provide unified security.

Transport Security

Transport security provides privacy and data integrity for data transported between two communicating applications, as well as authentication of the communicating end points. The most commonly used transport

security model on the Internet is a combination of HTTP Basic Authentication and SSL. HTTP Basic Authentication requires a user ID and password to visit Web sites that are protected using this mechanism. SSL is a protocol for transmitting data securely using encryption methods.

Is transport security alone enough?

Basic authentication is useful, but on its own not secure enough because the user ID and password are virtually unprotected. With SSL, it is possible to address a combination of server authentication and basic authentication to achieve authentication, confidentiality, and integrity.

Depending on requirements, this level of transport security is usually not enough. Basic authentication may not fit all situations and there may be a need for certificate-based or smart-card authentication because of inherent weaknesses in password-based mechanisms.

SSL is useful for ensuring confidentiality between *service requester* and *service provider*. However, at least two problems remain in the context of Web services messaging:

1. SOAP messages can include third-party

intermediaries that might need to read all or part of the message. Inherently, SSL assumes that communication occurs only directly between two parties.

2. Because SSL encrypts the entire message, individual parts of a message cannot be encrypted or decrypted. SOAP security using encryption helps to partially solve this problem.

Message Security

Message security focuses on two critical security concerns: first, protecting message content from being disclosed to unauthorized individuals (confidentiality); and second, preventing illegal modification of message content (integrity).

With respect to the exchange of XML messages between two parties, the XML Digital Signature provides a means to identify the sending party (identification), and to demonstrate that the message has not been modified (integrity). The SOAP Digital Signature specification defines how to embed the signature element of XML Digital Signature in a SOAP message as a header entry.

XML Encryption defines how to encrypt

data and represent the result as XML. The SOAP Encryption specification defines how to embed these elements in a SOAP message (header and body). The entire SOAP message or parts of a SOAP message can be encrypted to ensure confidentiality of the message contents when the message is sent to SOAP intermediaries.

Is message security alone enough?

Message security can be considered complementary to transport security. However, it deals with a smaller subset of the security issues for Web services. Although it covers identification, confidentiality, and integrity, it fails to address other important requirements, such as propagation of asserted identity across domains audit, etc. These requirements cannot be met by SOAP security alone, but by a combination of SOAP extensions and bindings to third-party security services (if necessary).

Firewall Security

Firewalls are used at the trust boundaries of enterprise networks. Firewall security enforces an access-control policy between two networks by blocking certain types of traffic and permitting others.

In the early days of firewalls, a few well-

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.

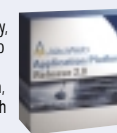


ALTOWEB

\$3,540.00

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.



GREENPOINT INC.

\$1,350.00

WEB CHARTS 3D VER. 4.5

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.

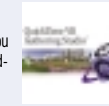


APPLE

\$359.99

QuickTime VR Authoring Studio v1.0

Apple QuickTime VR Authoring Studio software lets you create interactive virtual-reality scenes with point-and-click simplicity. It takes full advantage of the intuitive Mac OS interface to help you easily turn photos and computer renderings into attention-getting 360-degree views. QuickTime VR Authoring Studio is a powerful one-stop solution for producing all kinds of QuickTime VR content.

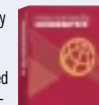


MACROMEDIA

\$93.99

Macromedia HomeSite 5.0

Macromedia HomeSite 5 provides a lean, code-only editor for Web development. Advanced coding features enable you to instantly create and modify HTML, CFML, JSP, and XHTML tags, while enhanced productivity tools allow you to validate, reuse, navigate, and format code more easily. Configure Macromedia HomeSite to fit your needs by extending its functionality and customizing the interface.



SILVERSTREAM

\$495.00

eXtend Application Server Developer Edition (5 User)

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



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

OFFERS SUBJECT TO CHANGE WITHOUT NOTICE

known protocols, such as HTTP, Telnet, and FTP, required support. With the advent of newer protocols the demand arose for firewall policies to permit these protocols, an approach that was not recommended. To avoid making changes to firewall policies, "tunneling" was used to transport proprietary protocols over common protocols.

Is firewall security alone enough?

The rise of SOAP as the basis of Web services meant that HTTP tunneling could be increasingly common. However, the current generation of firewalls provides protection only at the network level, with minimal application awareness. Security for Web services requires the addition of *application-level firewalls* that inspect SOAP data content to protect against external attacks, as well as limit the effects of malicious software being transferred unintentionally among business partners.

Component Security

The advent of Web services has created a far greater challenge to the effectiveness and completeness of security services in the area of components. Ironically, security models within existing enterprise components – considered fairly recent adoptions within the enterprise business frameworks – have already been tagged as "legacy" in terms of what they can offer.

Typically in a Web service, requests/invocations are delegated to well-defined endpoints hosting the request-processing service. For example, in the J2EE platform, these endpoints typically tend to be enterprise business components like J2EE EJBs, servlets, JSP, Java Web Services (JWS), portals, or Remote Portlet Web Services. It would seem logical to use the security provided by these existing components, such as J2EE Role-Based Access Control and/or

JAAS. Such a security mechanism could be programmatic or declarative.

Is component security alone enough?

The inherent security model of components (such as in J2EE) generally provides minimal security, such as coarse-grained access control (through roles). Often, once these applications have been Web services-enabled, additional security demands are placed on them in terms of complex, fine-grained access control and the portability of security information. Depending on component securities alone could lead to a single point of failure if compromised, as security is a cooperative effort among layers of protection.

Existing Security Approaches for Web Services Security Integration

Web services solutions in transport and message security have existed for a while and have been quite widely adopted by the Web services vendor community. But there are other parts of the Web services framework with security needs that have yet to be addressed for a standards-based solution. Hence, vendors of Web services environments (developer toolkits, runtime environments) are providing their own security mechanisms for access to registry, service, components, etc.

Security by vendor extensions

A common pattern in vendor-provided message security has been to place the security extensions in the SOAP header, in accordance with SOAP Header Extension Mechanism. The extensions may contain additional data to allow modular addition of features and services like transaction management, conversation state management, digital signatures, user credentials, and message routing (e.g., specifying the recipient). In this way, the message context

information can be propagated along the process chain of invocation.

Another pattern among vendor-provided Web services security approaches has been to make the security extensions part of the configuration information (XML file) for the Web services environment (container). Yet another pattern is to state security policies as tag extensions (similar to those used for javadocs) in the source code of the Web service implementations. Examples of such extensions include the authentication type to be used for the Web services, the delegation model (impersonation, etc.), and permissions for authorization based on roles.

Are security-by-vendor extensions alone enough?

When examined closely, we see that vendor-security extension has both pros and cons. Since this security is integrated with the deployment and runtime environment of Web services, it is readily available. However, configuration extensions are linked with the environment configuration, not the Web service definition, and thus lose portability across various deployment environments. Tag extensions (such as ones within source code) could lead to security policy inconsistencies, nonreusability, and maintainability issues.

Additionally, integration with third-party security products may become essential depending on the complexity of security required and the reuse of existing security harnesses deployed within the enterprise. However, vendor-provided security may not support such integration. The W3C has a Web Service Description Group working on enhancing Web services description information to include (among other things) security. This could not only improve runtime security, but also provide accurate access control at the time of publishing and brokering at the service registry.

Web Services Registry Security

With Web services, two kinds of service registries are planned. The global business registry disseminates service descriptions to the public. Private registries may be more selective and could restrict access and content.

Security concerns for all registries include prevention of unauthorized modification of registry content. Responsibility for administration of registry contents should rest with the

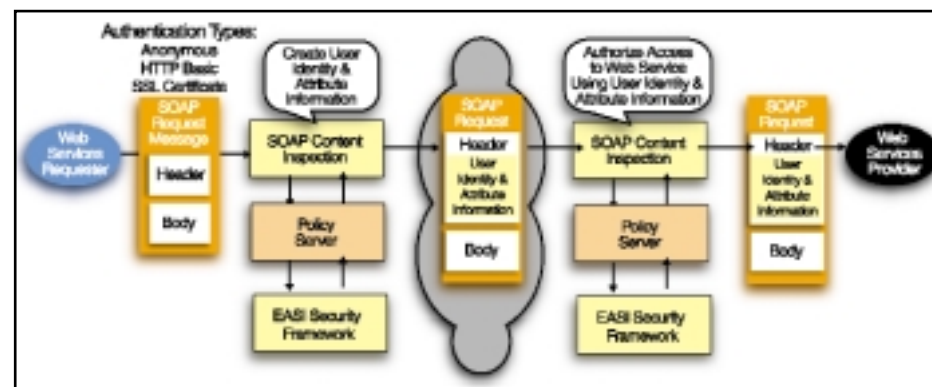


FIGURE 1 | SOAP Content Inspection

WebServices

.NET J2EE XML JOURNAL

The Best
.**NET**
Coverage
Guaranteed!

LEARN WEB SERVICES. GET A NEW JOB !

Subscribe today to the world's leading Web Services resource

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
- How to Develop and Market Your Web Services
- EAI and Application Integration Tips
- The Marketplace: Tools, Engines, and Servers
- Integrating XML in a Web Services Environment
- Wireless: Enable Your WAP Projects and Build Wireless Applications with Web Services!
- Real-World UDDI
- Swing-Compliant Web Services
- and much, much more!

Only \$69.99 for
1 year (12 issues)*
* Newsstand price \$83.88 for 1 year
Subscribe online at
www.wsj2.com or
call 888 303-5252
*Offer subject to change without notice



content provider, who should be able to modify or delete content. Some security requirements for private registries include:

- The registry authentication scheme should ensure that only known entities can access the registry information. A session-based security scheme could be provided.
- The registry authorization scheme should provide granular access control to both public and private areas within the registry.
- Messages between registry clients and services must be confidential.
- The content submitted to a registry must be maintained in the registry without any tampering en-route or within the registry, guaranteeing integrity.
- Though most of the security challenges for the service registry are in the areas of authentication, authorization, integrity, and confidentiality, more challenges unfold when we take into consideration registry replication and referral.

Registry security has yet to be addressed by a standards-based solution. For now, registry hosts and vendors provide their own security solutions, sometimes simply relying on under-

lying OS security. Given the potential diversity of possible registries, a framework-based multidomain approach seems to be required to ensure registry security.

Recommended Approach: Unified Security

From the preceding analysis of the shortfalls of existing Web services security solutions, it is apparent that transport, message, firewall, vendor, or component security alone may be insufficient to secure an entire system. Existing security requirements and any unaddressed residual security issues demand a specific framework-based approach – one that brings together various security services and solutions throughout a Web services environment, unifying security and making it ubiquitous.

What Is Unified Security?

Unified security means integrating existing and planned security technologies and products across the perimeter, middle, and legacy tiers, providing seamless, end-to-end security for your enterprise

The Need for EASI

Just as enterprise application integration (EAI) technologies addressed the problems of data access and resource management across the enterprise by integrating applications and business processes into a single, virtual “business engine,” companies now need a set of easy-to-use tools and technologies to control access to those same applications and processes. Today, a new class of technology, Enterprise Application Security Integration (EASI), is emerging to ensure that the distributed enterprise is protected.

The EASI Security Framework

The Enterprise Application Security Integration Framework is a standards-based, vendor-neutral security framework approach to unify the patchwork of security products and services deployed within the enterprise. The framework secures Web services traffic between .NET and J2EE environments, and integrates security across multitiered Web applications. It dynamically brokers security interactions across a range of security products, enterprise applications, application platforms, and Web

services platforms at a transaction level. It supports all categories of security products for authentication, authorization, accountability, administration, and cryptography.

By linking multiple security technologies and creating a common security infrastructure, the EASI Framework enables organizations to avoid potentially less secure and far more costly custom-built security solutions. It represents a cost-effective solution for unified security that is flexible, scalable, highly available, and reliable, positioning the enterprise-wide security infrastructure for future growth and change.

We have seen that we cannot buy “partial security” when it comes to deploying Web services. Traditional firewall security can't peek into the message envelope and decipher XML traffic. Content inspection is critical for improved Web services security.

SOAP Content Inspection

Within SOAP-based Web services environments, a key service of the EASI framework is SOAP Content Inspection (see Figure 1). This flexible, standards-based solution secures SOAP-based transactions

for a range of enterprise B2B applications.

SOAP Content Inspection acts as an application-level firewall capable of inspecting the contents of communication. It provides a new layer of security for Web services – one that sits above transport or session layer security and examines the XML data itself. This XML data may be security or authentication information, such as an XML Signature or a SAML Assertion that could be evaluated using a security policy, or it could be a rogue Web service request designed to attack the service.

SOAP Content Inspection addresses the challenges of Web services security by:

- Authenticating and authorizing SOAP message originators, using identity and attribute information to determine if a user is authorized to access the Web service.
- Establishing security context for authenticated users.
- Securing messages using XML-based encryption and digital signatures.
- Providing security audit capabilities for Web service access and usage.
- Enabling establishment of enterprise security services policies that are consistently

enforced during authentication, authorization and auditing. This policy-based approach means protection strength can be set independently of the application.

- Facilitating cross-domain security interoperability by mapping user attributes from the originating enterprise domain to attributes in the destination enterprise domain using the Attribute-Mapping Service. This allows each processing domain to continue to function normally without requiring changes to existing code and policies.
- Relieving application programmers of the burden of implementing additional, security-oriented functionality – reducing development time, while improving inter-business interoperability.

SOAP Content Inspection can use Security Assertion Markup Language (SAML) to provide a standards-based approach for security information portability. SAML is an initiative orchestrated by OASIS (Organization for the Advancement of Structured Information Standards), an organization that has been a driving force in defining emerging standards for security.

Once you're in it...

...reprint it!

- Wireless Business & Technology
- Web Services Journal
- Java Developer's Journal
- XML Journal
- ColdFusion Developer's Journal
- PowerBuilder Developer's Journal

Contact Carrie Gebert
201 802-3026
carrieg@sys-con.com

RePrints

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.

WebSphere Developers Journal

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

The World's Leading Independent WebSphere Developer Resource



FIGURE 2 | SOAP Content Inspection with EASI Security Framework

ty information portability and security policies. More information on SAML is available at www.oasis-open.org/committees/sec-urify.

It is ideal that SOAP Content Inspection uses the services of the underlying EASI Security Framework (see Figure 2), which supports access to multiple underlying security service providers. In this way, any

component that does SOAP Content Inspection, acts as an "appliance" (a plug-gable, easy-to-use component) that uses the framework services.

Conclusion

Protecting the security and privacy of exchanged information is absolutely critical to

the success of any Web services architecture. This security model must be incorporated into the foundational layers of the enterprise Web services framework, supporting an evolving implementation philosophy. Enterprise applications cannot be forced to rely on a single provider for their security services.

Given these realities, enterprises wishing to tap the advantages of Web services must adopt approaches to enterprise-wide security that enhance both security and flexibility. Implementing a comprehensive, standards-based security framework with purpose-built solutions such as SOAP Content Inspection, meet both critical requirements. Together, they provide a flexible way to unify diverse security solutions across multiple domains and to ensure the security of Web service applications, messages, and data.

No security architecture for Web services should require enterprises to drastically alter their existing security architecture. Implementing an enterprise-wide security framework enables organizations to leverage existing investments, relationships, and policies for unified security. ©

The World's Leading Java Resource!

JAVA

DEVELOPERS 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

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

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

WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

\$99 + S&H
Special Limited-time Price



Now Shipping **\$119 CD VALUE** FROM JDJ

EVERY ISSUE OF **WSJ & JDJ** EVER PUBLISHED

THE MOST COMPLETE LIBRARY OF EXCLUSIVE WSJ & JDJ ARTICLES ON ONE CD!

"The Secrets of the Web Services Masters"

CD is edited by well-known editors-in-chief Sean Rhody and Alan Williamson and organized into more than 50 chapters containing more than 1,400 exclusive *WSJ* & *JDJ* articles.

WWW.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

MORE THAN
1400
EXCLUSIVE
WEB SERVICES
& JAVA
ARTICLES

7
YEARS
83
ISSUES
1400+
ARTICLES

ONE
CD

**ORDER
ONLINE**
AT JDJSTORE.COM
**SAVE
\$20**



Written by Jim Webber

It's a fact: Web services have started to mature. Those emergent standards that once held so much promise are now actually starting to deliver useful implementations. With the basic Web services plumbing mastered, we're starting to see more advanced infrastructure, which enables these second-generation Web services to focus on complex interactions over the Internet. This article, the first of a two-part series, covers one such aspect of the second-generation infrastructure for Web services: transactions.

AUTHOR BIO: Jim Webber is a senior engineer with Hewlett-Packard, based at the Arjuna Laboratory in Newcastle upon Tyne. Within the Arjuna Lab, HP's center of excellence for transactioning, Jim is involved with research and development on transactional Web services, and architecting HP's Web Service Transactioning solution. Jim is also involved with standards processes for transactional Web services, and is currently participating in the OASIS BTP effort.
JIM_WEBBER@HP.COM

Overview

The OASIS Business Transactions Protocol, or BTP, has become the prominent standard for Web services transactions. BTP is the product of just over a year's work by vendors such as HP, BEA, and Oracle, and has resulted in the development of a transaction model particularly suited to loosely coupled systems like Web services.

In this article, we're going to look at how BTP fits into the whole Web services architecture, and how we can use one of the vendor toolkits (we'll use HP's toolkit,

but the underlying principles apply to other vendors' software) to build and consume transaction-aware Web services. But before we do, let's review the architecture in the context of a simple transactional scenario.

The diagram shown in Figure 1 is similar to a typical high-level Web services architecture. The only differences here are that one service, the transaction manager, has been singled out as being distinct from the other Web services (which we assume are responsible for some aspects of a business process), and

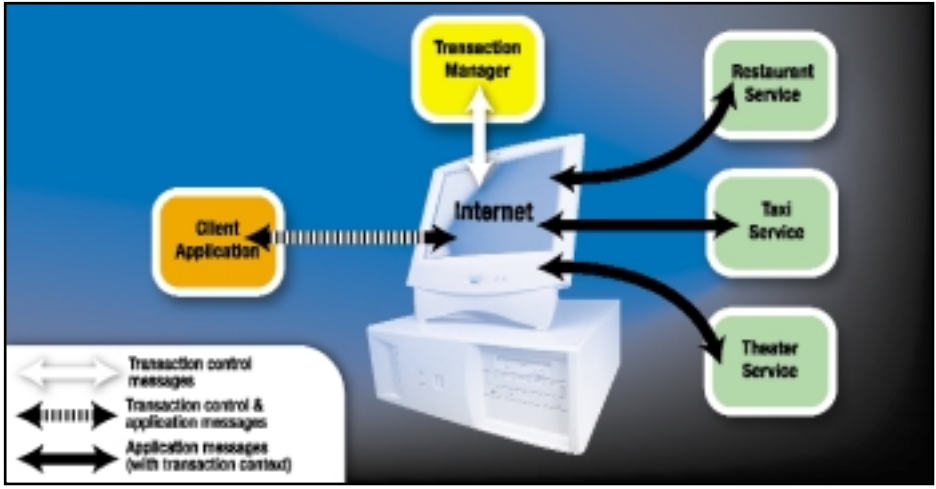


FIGURE 1 | Architecture of a simple transactional Web service-based application

“ Though Web services is a hot technology, we shouldn't lose sight of the fact that it exists to support business processes ”

the fact that we've chosen to identify two distinct categories of messages: control messages (which are used to control transactions) and application messages (which propagate application data around the system).

Of course, if it really were as simple as deploying a transaction manager service into the architecture, then this article wouldn't be necessary. Unfortunately it's not that simple; or at least not quite that simple, as we shall see. To illustrate, it's convenient to use Figure 1 as a point of reference as we work through the architecture, filling in the details. We'll work from left to right, from the client through to the Web services, and cover everything in between.

Consuming Transactional Web Services

Though Web services is a hot technology, we shouldn't lose sight of the fact that it exists to support business processes. With that in mind, the right place to start our investigation is most definitely at the client end of a system – where the results of Web services interactions are brought together and where the value of a business process is ultimately focused. To place this in the proper context, it's useful to see an exploded view of the client-side infrastructure, shown in Figure 2.

In a nontransactional Web services-based application, the client process can be something as simple as a collection of calls (via proxies) to services that are involved in the activity. In a transactional Web services-based application, the same is (surprisingly enough) true, with

the caveat that the developer must demarcate any transactions that support business logic, as well as deal with application-specific calls. In this case the transaction demarcation is supported by the client transaction API (the Client Tx API in Figure 2), whereas the business methods supported by service proxies appear to logically remain free of any transactional infrastructure from the point of view of the client application developer. In fact, under the covers there is a mechanism that performs context associations with local threads of control within the client and messages passed between the client and (transactional) Web services. In Figure 2, this is the purpose of the Tx Context Interceptor.

Client API
The client API provides the developer

with the necessary tools with which to structure and control transactions within the application. The commands available to a developer in a transactional Web services environment are quite familiar to those of us that have used other transaction APIs in the past, with the caveat that BTP supports full control over both phases of the commit process and thus has a larger command set than we might otherwise envision. The UserTransaction API supports the common verbs (and by implication the methods that enact those verbs) for transaction demarcation:

- **Begin:** Creates a new top-level transaction (or subtransaction) for either atomic or cohesive transactions
- **Prepare:** Instructs an atomic transaction to prepare its associated participating services when the transaction is

BTP in Brief

Although this article isn't about the BTP model itself, we still need to be familiar with a couple of BTP concepts. The bare minimum we can get away with is a little background on the two types of transactions BTP supports, which are:

- **Atoms:** These are similar to traditional atomic transactions in which Web services participating in an atom are guaranteed to see the same outcome as all the other participants.
- **Cohesions:** These allow business logic to dictate which combination of participating services can succeed or fail, while permitting the transaction as a whole to make forward progress – this allows the business transaction to proceed even in the presence of failures.

Cohesions allow us to pick a group of participating services (known as the confirm set) that we would eventually like to reach completion, while atoms require that all participating services either confirm or cancel unanimously.

It's also worth bearing in mind that BTP adopts a two-phase approach to its transaction coordination. At termination time, services participating in a transaction are asked up front to state their intention (whether they will proceed with the transaction or whether they are not prepared to do so), and will later be instructed by the transaction manager to either proceed or not, based on the analysis of all of the collected intentions. As we shall see, this model is presented explicitly to the programmer through the toolkit API.

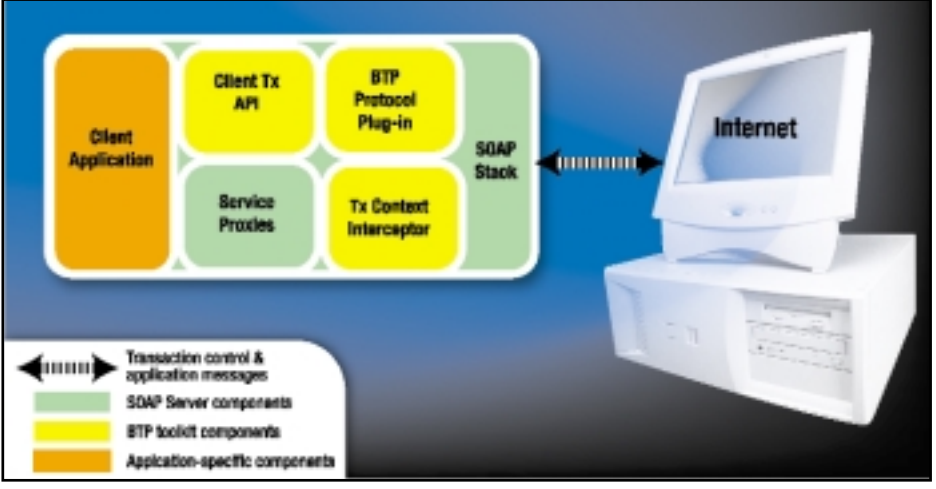


FIGURE 2 | BTP client-side infrastructure

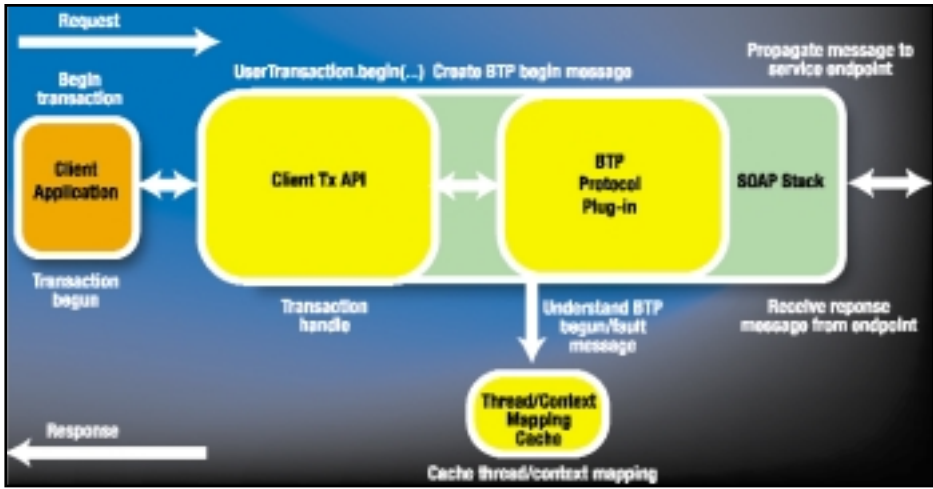


FIGURE 3 | Creating and consuming BTP messages at the client side

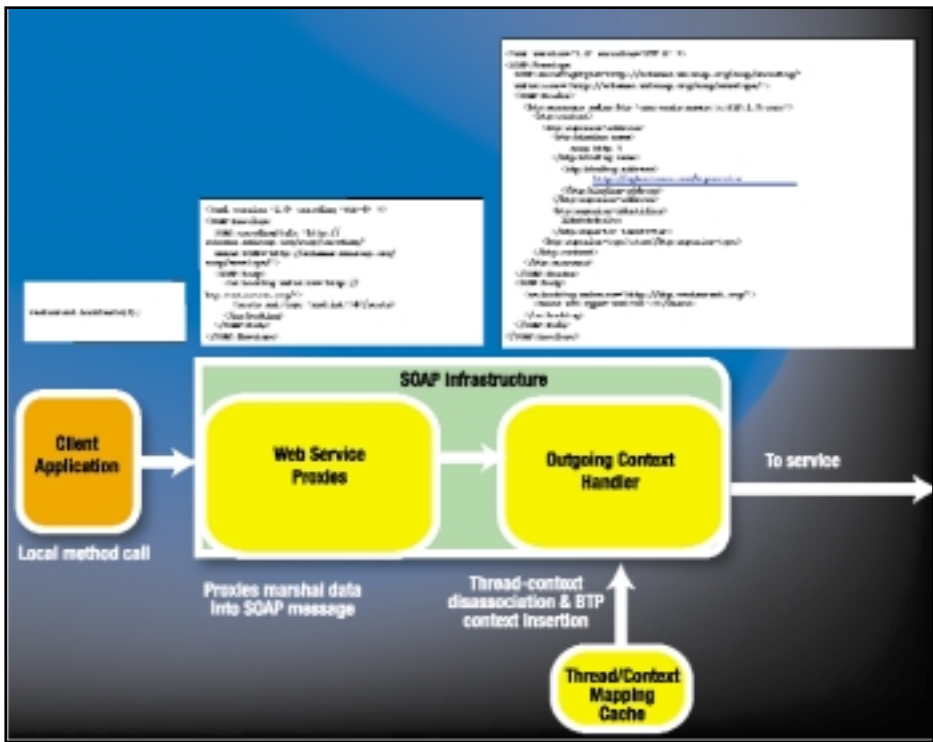


FIGURE 4 | Client-side request processing

- to terminate
- **Prepare Inferiors:** Instructs a cohesive transaction to prepare one or more of its participating services at transaction termination time
 - **Confirm:** Instructs an atomic transaction to confirm all of its participating services, and confirms all participant services that voted to confirm in the case of a cohesive transaction
 - **Cancel:** Instructs all participating services in an atomic transaction, or those

- services specified in the parameter to the method call in a cohesive transaction, to cancel
- In addition to these demarcation verbs, a number of other commands can be used to inquire about a transaction:
- **Status:** Asks the transaction manager to return the state (e.g., committed, preparing) of the current transaction
 - **Transaction type:** Exposes the type of the current transaction (i.e., atom or cohesion)

- **Transaction name:** Exposes the name of the current transaction in string form
- Two verbs allow advanced manual transaction management:
- **Suspend:** Disassociates the current thread from the current transaction
 - **Resume:** (Re)associates the current thread with the current transaction

Those who have previously worked with transactions will immediately find themselves at home with this API, since it is in the same spirit as other transaction APIs like JTA. Let's take a look at an example.

In the code shown in Listing 1, we see an atom being used to ensure a consistent outcome across calls to the Web services shown in Figure 1. Initially we obtain a reference to an instance of UserTransaction from a (previously initialized) UserTransactionFactory, which we then use to delimit the scope of the single transaction in our application. Our atomic transaction is started by calling the begin(...) method on the user transaction API and specifying the type of transaction as an atom. From now on the business logic is straightforward and contains no further transaction control primitives; we simply go ahead and make the bookings we want for our night out through the book(...) methods of the service proxies we created.

Once the business logic has completed, we can terminate the transaction by calling prepare(...) and confirm(...) which, in the absence of failures, should confirm to all parties that they should henceforth honor all our booking requests. If there are failures, then all parties are informed and should take the necessary steps to undo any work undertaken on our behalf, while the client application will receive an exception that details what exactly has gone wrong.

The great thing about this example is that it shows just how simple and relatively noninvasive it can be to wrap work with Web services within a transaction. In fact, the business logic aspects of the code would be the same irrespective of whether or not transactions are used.

Under the Covers:
BTP's Two-Pipe Model
To support transactional Web serv-

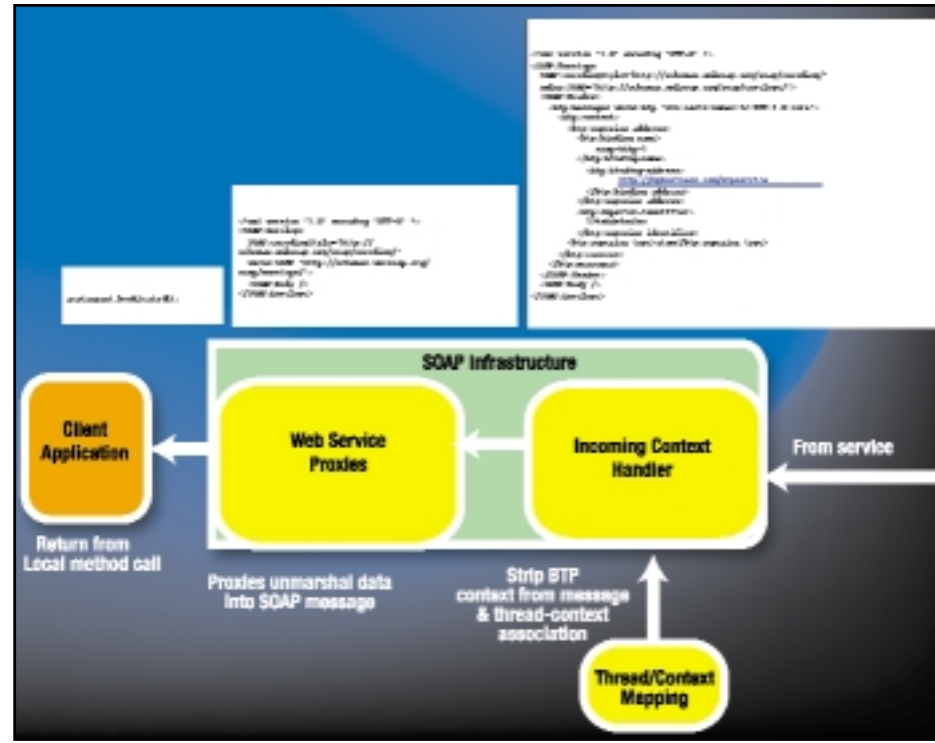


FIGURE 5 | BTP client-side response processing

es-based applications, BTP utilizes two distinct types of messages that the client application exchanges with business Web services. The first of these messages is exchanged exclusively within the transaction infrastructure. The other type consists of messages that the client exchanges with business Web services onto which BTP messages might be piggy-backed.

The messages that the application exchanges with the transaction infrastructure are encapsulated by the primitives supported by the client API. For example, a begin(...) method being executed by the client causes a corresponding BTP begin message to be sent to a transaction manager via the SOAP server, and for response messages from the transaction manager to be processed in the reverse order. This is shown in Figure 3, and a sample BTP message (begin) is shown in Listing 2. The only slightly unusual aspect to this example is that the response to begin messages (and only begin messages) is cached for later use so local threads of execution can be associated with the BTP transaction under which its work is being carried out.

When transporting application messages, the situation is a little different. Unlike BTP messages in which the message content travels in the body of the SOAP envelope, when application messages are sent, application-specific content travels in the body, while any BTP messages are relegated to the header part of the envelope. We can see this in Listing 3, in which the SOAP body carries the application payload, while the header is used to carry the BTP context.

This scheme works well since most SOAP stacks are well equipped to perform efficient header processing, and placing the BTP content, including the transaction context, in the header means that SOAP actors can pick out the parts of the header space that are of interest without having to parse the whole application payload. From a development point of view, most SOAP servers support pluggable header processors, which means that building BTP context processing into your infrastructure should be straightforward. To demonstrate that point, let's take a look at the general client-side architecture (which is based on Apache Axis in the toolkit we've used), as per the examples in Figure 3 and Listing 2.

Figure 4 shows the outward path of a call to a Web service, starting from the left with the local method call to a service proxy. The call then follows the logical path of being converted to the appropriate SOAP body, which contains the application payload, before it progresses to the outgoing context handler. The context handler takes advantage of the fact that the information supplied in response to the BTP begin message was recorded, and is able to produce a BTP context from that data, which it duly inserts into the SOAP envelope's header. If there is no contextual data stored for the current thread (i.e., it isn't part of a transaction or the transaction has been deliberately suspended), then the context insertion is simply bypassed.

For return messages, the strategy is simply the reverse, as shown in Figure 5, in which the flow is from right to left. Responses are quickly scanned to see if they contain any BTP context entries in their headers. If context data is present, it is stripped out of the message and may be used to resume the transaction locally by associating the current thread while the rest of the message passes through to the

“ From a development point of view, most SOAP servers support pluggable header processors, which means that building BTP context processing into your infrastructure should be straightforward ”

service proxies. Once at the service proxies, the local method call returns control to the client, which is unaware of all of the additional processing that has occurred on its behalf.

Having reached the point where we can send application messages with BTP contexts, as well as BTP messages themselves, we're able to follow the messages as they travel across the wire. Following the cables

inevitably leads us to business Web services.

Summary

The first article in this series on implementing transactional Web services-based applications has shown how client applications can be constructed using off-the-shelf BTP toolkits. We've seen how much of the hard work involved in managing transactions has been delegated to

the toolkit and the underlying SOAP infrastructure, leaving to the developer the real value-add work of getting the application logic and transaction structure right. However, this is only half the story. In the next article, we'll investigate what happens at the Web service end, and show how true enterprise-class Web services applications can be made transactional from end to end. ☺

Listing 1: Controlling atoms from a Web service client

```
// obtain a UserTransaction object (assume it is previously
// initialized)
userTransaction = UserTransactionFactory.getTransaction();

// obtain references to the Web services we are going to use:
restaurant = new RestaurantService();
taxi = new TaxiService();
theatre = new TheatreService();

// Start a new transaction, using a simple atom
userTransaction.begin(com.hp.mw.xts.TxTypes.ATOM);

// now invoke the business logic
restaurant.bookSeats(3);
theatre.bookSeats(3, 2);
taxi.bookTaxi();

// prepare the transaction (first phase of two phase
coordination)
userTransaction.prepare();

// confirm the transaction (second phase of two phase
coordination)
userTransaction.confirm();
```

Listing 2: A BTP begin message

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP:Envelope
SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Body>
<btp:begin transaction-type="atom"
xmlns:btp="urn:oasis:names:tc:BTP:1.0:core" />
```

```
</SOAP:Body>
</SOAP:Envelope>
```

Listing 3: An application message with BTP context

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP:Envelope
SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Header>
<btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core">
<btp:context>
<btp:superior-address>
<btp:binding-name>soap-http-1</btp:binding-name>
<btp:binding-address>
http://mybusiness.com/btpservice
</btp:binding-address>
</btp:superior-address>
<btp:superior-identifier>
12fa6de4ea3ec
</btp:superior-identifier>
<btp:superior-type>atom</btp:superior-type>
</btp:context>
</btp:messages>
</SOAP:Header>
<SOAP:Body>
<ns:booking xmlns:ns="http://btp.restaurant.org/">
<seats xsi:type="xsd:int">99</seats>
</ns:booking>
</SOAP:Body>
</SOAP:Envelope>
```

Download the code at
sys-con.com/webservices

introductory
subscription offer!

coming soon...

A TRULY INDEPENDENT VOICE IN THE WORLD OF .NET

.NET Developer's Journal is the leading independent monthly publication targeted at .NET developers, particularly advanced developers. It brings .NET developers everything they need to know in order to create great software.

Published monthly, *.NET Developer's Journal* covers everything of interest to developers working with Microsoft .NET technologies – all from a completely independent and nonbiased perspective. Articles are carefully selected for their prime technical content – technical details aren't watered down with lots of needless opinion and commentary. Apart from the technical content, expert analysts and software industry commentators keep developers and their managers abreast of the business forces influencing .NET's rapid development.

Wholly independent of both Microsoft Corporation and the other main players now shaping the course of .NET and Web services, *.NET Developer's Journal* represents a constant, neutral, expert voice on the state of .NET today – the good, the bad, and the ugly...no exceptions.

Here's what you'll find in every issue of .netdj:

Security Watch
Mobile .NET
.NET Trends
Tech Tips
Standards Watch
Business Alerts
.NET News
Book and Software
Announcements

.NET Developer's Journal is for .NET developers of all levels, especially those "in the trenches" creating .NET code on a daily basis:

- For beginners: Each issue contains step-by-step tutorials.
- For intermediate developers: There are more advanced articles.
- For advanced .NET developers: In-depth technical articles and columns written by acknowledged .NET experts.

Regardless of their experience level, *.NET Developer's Journal* assumes that everyone reading it shares a common desire to understand as much about .NET – and the business forces shaping it – as possible. Our aim is to help bring our reader-developers closer and closer to that goal with each and every new issue!

SUBSCRIBE ONLINE!

www.sys-con.com/dotnet/

or Call

1-888-303-5282

**SAVE 16%
OFF**
THE ANNUAL COVER PRICE

Get 12 issues of **.NETDJ**
for only \$69⁹⁹!

**ANNUAL
COVER PRICE:**

~~\$83.88~~

YOU PAY

\$69⁹⁹

YOU SAVE

\$13.89

OFF THE ANNUAL
COVER PRICE

The Business Transaction Protocol: Transactions for a New Age

Bringing ACID models into the Web services world

Atomic transactions are a well-known technique for guaranteeing consistency in the presence of failures. The ACID properties of atomic transactions ensure that, even in complex business applications, consistency of state is preserved.

Transactions are best viewed as “short-lived” entities operating in a closely-coupled environment, performing stable state changes to the system; they are less well suited for structuring “long-lived” application functions (e.g., running for hours, days, etc.) and running in a loosely coupled environment like the Web. Long-lived atomic transactions (as typically occur in business-to-business interactions) may reduce the concurrency in the system to an unacceptable level by holding on to resources (e.g., locks) for a long time; further, if such an atomic transaction rolls back, much valuable work already performed could be undone. As a result, there have been various extended transactions models where strict ACID properties can be relaxed in a controlled manner. Until recently, translating these models into the world of Web services had not been attempted. However, the OASIS Business Transactions Protocol, specified by a collaboration of several companies, has tried to address this issue.

Introduction

With the advent of Web services, the Web is being populated by service providers who wish to take advantage of this large B2B space. However, there are still important security and fault-tolerance considerations that must be addressed. One of these is the fact that the Web frequently suffers from failures that can affect both the performance and consistency of applications that run over it.

Atomic transactions are a well-known technique for guaranteeing consistency in the presence of failures. (Note: I will not use the term *transaction* in place of *atomic transaction* since in the B2B space this has different connotations.) The ACID properties of atomic transactions (Atomicity, Consistency, Isolation, Durability) ensure that even in complex business applications consistency of state is preserved, despite concurrent accesses and failures. This is an extremely useful fault-tolerance technique, especially when multiple, possibly remote, resources are involved.

The structuring mechanisms available within traditional atomic transaction systems are sequential and concurrent composition of transactions. These mechanisms are sufficient if an application function can be represented as a single atomic transaction. As Web services evolved as a means to integrate processes and applications at an inter-enterprise level, traditional transaction semantics and protocols

have proven inappropriate. 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.

As a result, there have been various extended transactions models, in which strict ACID properties can be relaxed in a controlled manner. Until recently, translating these models into the world of Web services had not been attempted. However, the OASIS Business Transactions Protocol (BTP), specified by a collaboration of several companies, has tried to address this issue. In this article we'll first consider why traditional atomic transactions are insufficient for long-running B2B activities, and then describe how the BTP protocol has attempted to solve these problems.

Why ACID Transactions Are Too Strong

ACID transactions by themselves are inadequate for structuring long-lived applications. To ensure ACID-ity between multiple participants, a multiphase (typically two) consensus mechanism is required (see Figure 1). During the first (*preparation*) phase, an individual participant must make durable any state changes that occurred during the scope of the atomic transaction, such that these changes can either be rolled back (undone) or committed later once consensus to the transaction outcome has been determined among all participants, i.e., any original state must not be lost at this point, as the atomic transaction could still roll back. Assuming no failures occurred during the first phase (in which case all participants will be forced to undo their changes), in the second (*commitment*) phase, participants may “overwrite” the original state with the state made durable during the first phase.

In order to guarantee consensus, a two-phase commit is necessarily a

blocking protocol. After returning the phase 1 response, each participant that returned a commit response *must* remain blocked until it has received the coordinator's phase 2 message telling it what to do. Until they receive this message, any resources used by the participant are unavailable for use by other atomic transactions, since to do so may result in non-ACID behavior. If the coordinator fails before delivery of the second phase message these resources remain blocked until it recovers. In addition, if a participant fails after phase 1, but before the coordinator can deliver its final commit decision, the atomic transaction cannot be completed until the participant recovers: *all* participants must see *both* phases of the commit protocol in order to guarantee ACID semantics. There is no implied time limit between a coordinator sending the first phase message of the commit protocol and it sending the second, commit phase message; there could be seconds or hours between them.

Therefore, structuring certain activities from long-running atomic transactions can reduce the amount of concurrency within an application or (in the event of failures) require work to be performed again. For example, there are certain classes of application where it is known that resources acquired within an atomic transaction can be released “early,” rather than having to wait until the atomic transaction terminates; in the event of the atomic transaction rolling back, however, certain compensation activities may be necessary to restore the system to a consistent state. Such compensation activities (which may perform forward or backward recovery) will typically be application specific, may not be necessary at all, or may be more efficiently dealt with by the application.

For example, long-running activities can be structured as many independent, short-duration atomic transactions, to form a “logical” long-running transaction. This structure allows an activity to acquire and use resources for only the required duration of this long-running activity. In Figure 2 an application activity (shown by the dotted ellipse) has been split into many different, coordinated, short-duration atomic transactions. Assume that the application activity is concerned with booking a taxi (*t1*), reserving a table at a restaurant (*t2*), reserving a seat at the theater (*t3*), booking a room at

a hotel (*t4*), and so on. If all of these operations were performed as a single atomic transaction, then resources acquired during *t1* would not be released until the atomic transaction has terminated. If subsequent activities *t2*, *t3*, etc., do not require those resources, then they will be needlessly unavailable to other clients.

However, if failures and concurrent access occur during the lifetime of these individual transactional activities, then the behavior of the entire “logical long-running transaction” may not possess ACID properties. Therefore, some form of (application-specific) compensation may be required to attempt to return the state of the system to consistency. For example, let's assume that *t4* aborts. Further assume that the application can continue to make forward progress, but in order to do so must now undo some state changes made prior to the start of *t4* (by *t1*, *t2*, or *t3*). New activities are started; *tc1* is a compensation activity that will attempt to undo state changes performed by, say, *t2* and *t3*, which will continue the application once *tc1* has completed. *tc5'* and *tc6'* are new activities that continue after compensation, e.g. since it was not possible to reserve the theater, restaurant, and hotel, it is decided to book tickets at the cinema. Obviously, other forms of composition are possible.

Properties of a Web Service-Based Transaction

The fundamental question addressed here is what properties must a transaction model possess in order to support business-to-business interactions? To begin to answer that, we need to understand what we mean by a *business transaction*.

A *business relationship* is any distributed state maintained by two or more parties and is subject to some contractual constraints previously agreed to by those parties. A business transaction can therefore be considered as a consistent change in the state of a business relationship between parties. Each party in a business transaction holds its own application state corresponding to the business relationship with other parties in that transaction. During the course of a business transaction, this state may change.

In the Web services domain, information about business transactions is communicated

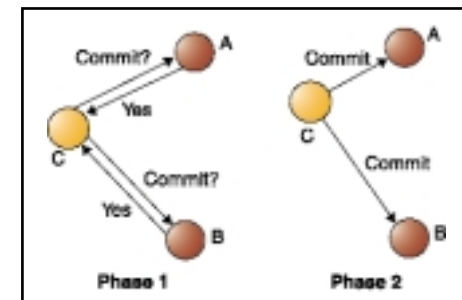


FIGURE 1 Two-phase commit protocol

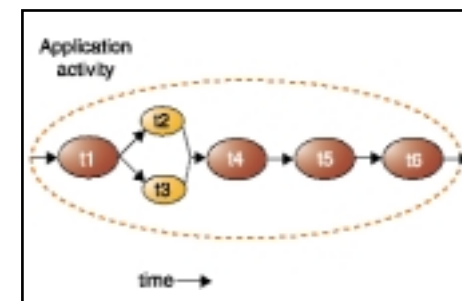


FIGURE 2 An example of a logical long-running “transaction,” without failure

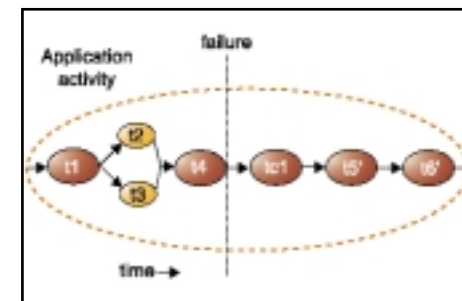


FIGURE 3 An example of a logical long-running “transaction,” with failure

in XML documents. However, how those documents are exchanged by the different parties involved (e.g., e-mail or HTTP) may be a function of the environment, type of business relationship, or other business or logistical factors. Therefore, mandating a specific XML *carrier protocol* may be too restrictive.

Since business relationships imply a level of value to the parties associated by those relationships, achieving some level of consensus among these parties is important. Not all participants within a particular business transaction have to see the same outcome; a specific transaction may possess multiple *consensus groups*.



Author Bio

Mark Little is a distinguished engineer/architect within HP Arjuna Labs, where he leads the HP-TS and HP-WST teams. He is one of the primary authors of the OMG Activity Service specification, is on the expert group for JSR 95, and leads the JSR 156 activity on an XML API for Java Transactions. He is HP's representative on the OTS Revision Task Force, and the OASIS Business Transactions Protocol specification. MARK_LITTLE@HP.COM

In addition to understanding the outcomes, a participant within a business transaction may need to support provisional or tentative state changes during the course of the transaction. Such parties must also support the completion of a business transaction, either through confirmation (final effect) or cancellation (counter-effect). In general, what it means to confirm or cancel work done within a business transaction will be for the participant to determine.

For example, an application may choose to perform changes as provisional effects and make them visible to other business transactions. It may store necessary information to undo these changes at the same time. On confirmation, it may simply discard these "undo", changes, or on cancellation it may apply these "undo" changes. An application can employ such a compensation-based approach or take a conventional "rollback" approach. It is with these properties in mind that we can discuss the Business Transaction Protocol.

The Business Transaction Protocol

B2B interactions may be complex, involving many parties, spanning many different organizations, and potentially lasting for hours or days, e.g., the process of ordering and delivering parts for a computer may involve different suppliers, and may only be considered to have completed once the parts are delivered to their final destination. Most business-to-business collaborative applications require transactional support in order to guarantee consistent out-

come and correct execution. These applications often involve long-running computations, loosely coupled systems, and components that do not share data, location, or administration; it is then difficult to incorporate ACID transactions within such architectures. Furthermore, most collaborative business process management systems support complex, long-running processes in which undoing tasks that have already completed may be necessary in order to effect recovery or to choose another acceptable execution path.

For example, an online bookshop may well reserve books for an individual for a specific period of time, but if the individual does not purchase the books within that time period they will be "put back onto the shelf" for others to purchase; to do otherwise could result in the shop never selling a single book. Furthermore, because it is not possible for anyone to have an infinite supply of stock, some examples of online shops may appear to users to reserve items for them, but in fact if other users want to purchase them first they may be allowed to (i.e., the same book may be "reserved" for multiple users concurrently); a user may subsequently find that the item is no longer available, or may have to be ordered especially for them. If these examples were modelled using atomic transactions, then the reservation process would require the book to be locked for the duration of the atomic transaction – it would have to be available, and could not be acquired by (sold to) another user. When the atomic transaction commits, the book will be removed from stock and

mailed to the user. However, if a failure occurs during the commitment protocol, the book may remain locked for an indeterminate amount of time (or until manual intervention occurs).

As a result, the use of traditional atomic transactions with strict ACID properties (e.g., systems that implement the

JTS specification [SUN99]) is considered too restrictive for many types of applications.

The Organization for the Advancement of Structured Information Standards (OASIS) Business Transaction Protocol (BTP) is a transaction protocol that meets the requirement for Web-based, long-running collaborative business applications. BTP is designed to support applications that are disparate in time, location, and administration and thus require transactional support beyond classical ACID transactions. In short, BTP is a protocol for ensuring consistent outcomes from participating parties in a business transaction.

Note: It is important to realize that the term "transaction" in this sense does not mean atomic transaction, although ACID semantics can be obtained if required.

Consensus of Opinion

In general, a business transaction requires the capability for certain participants to be structured into a consensus group such that all of the members in a grouping have the same result. Different participants within the same business transaction may belong to different consensus groups. The business logic then controls how each group completes. In this way, a business transaction may cause a subset of the groups it naturally creates to perform the work it asks, while asking the other groups to undo the work.

Consider the situation shown in Figure 4, in which a user is booking a holiday, has provisionally reserved a flight ticket and taxi to the airport, and is now looking for travel insurance. The first consensus group holds Flights and Taxi, since neither of these can occur independently. The user may then decide to visit multiple insurance sites (called A and B in this example), and as he goes may reserve the quotes he likes. So, A may quote \$50, which is just within budget, but the user may want to try B just in case he can find a cheaper price, without losing the initial quote. If the quote from B is less than that from A, the user may cancel A while confirming both the flights and the insurance from B. Each insurance site may therefore occur within its own consensus group.

SAVE 17% OFF

DEVELOPER'S PowerBuilder Journal

OFFER SUBJECT TO CHANGE WITHOUT NOTICE 12 Issues for **\$149**

• New PowerBuilder features • Tips, tricks, and techniques in server-side programming • DB programming techniques • Tips on creating live PowerBuilder sites • Product reviews • Display ads for the best add-on products

That's a savings of **\$31** off the annual newsstand rate. Visit our site at www.sys-con.com/pbd/ or call 1-888-303-5282 and subscribe today!

WebServices JOURNAL

home
advertise
subscribe
contact

SYS-CON
EVENTS
SYS-CON
MEDIA

What's Online

www.sys-con.com/webservices

Join the Web Services Discussion Group

Web Services Journal proudly announces WSJList, the NEW Web services mailing list community at the center of discussions, technical questions, and more...

Join the WSJList now to monitor the pulse of the Web services industry!

WSJ2.com

Check in daily for breaking news from the WSJ News Desk on Web services products, industry events, developments, and happenings. Be the first to know what's going on!

Digital Edition

Don't have your print edition? Can't wait to read the next issue? Our digital edition is just what you need. As long as you have your computer with you, you can read Web Services Journal anytime, anywhere.

WSJ Industry Newsletter

Subscribe now for more in-depth Web services coverage. The most up-to-date coverage of companies producing products and technology that are at the forefront of this paradigm. Brought to you every month by industry leaders.

Write for Us!

Are you a developer with experience in using Web services? Do you have a success story about how your firm implemented its Web services? Do you have an idea that you think will be of interest to our readers?

We'll be happy to look at your article proposals. Go to <http://editorial.sys-con.com/proposal.cfm>, fill out the form, and our editors will let you know if we can use your material.



QUICK POLL

Will Web Services be "The Next Big Thing?"
☐ A- Yes
☐ B- No

Vote

Results

SonicXQ™

SOAPswitch
Web Services Gateway

spiritsoft
go beyond jms

XML and Web Services
UNLEASHED
SAMS

Click for a
FREE 30-day
trial
AltoWeb

CLICK HERE
NOW
SilverStream
eXtend
Web Services

WEB SERVICES
Reality
SPRING 2002
CONFERENCE

sitraka.com
sitraka



FIGURE 4 | Flight booking

This is not possible when using ACID transactions.

BTP uses a two-phase *completion* protocol to guarantee *atomicity of decisions* but does not imply specific implementations. To enforce this distinction, rather than call the second phases of the termination protocol “commit” and “rollback” as is the case in an ACID transaction environment, they are called “confirm” and “cancel” respectively, with the intention of decoupling the phases from any preconceptions of specific backward-compensation implementations.

It's important to stress that although BTP uses a two-phase protocol, *it does not imply ACID transactions*. How implementations of prepare, confirm, and cancel are provided is a back-end implementation decision. Issues to do with consistency and isolation of data are also back-end choices and not imposed or assumed by BTP. A BTP implementation is primarily concerned with two-phase coordination of abstract entities (participants).

Open-Top Coordination

In a traditional transaction system, the

application or user has very few verbs with which to control the transaction. Typically, these are “begin,” “commit,” and “roll back,” corresponding to starting a transaction, committing a transaction, and rolling back a transaction respectively. When an application asks for a transaction to commit, the coordinator will execute the entire two-phase commit protocol, as described earlier, before returning an outcome to the application (what BTP terms a *closed-top commit protocol*). The elapse time between the execution of the first phase and the second phase is typically milliseconds to seconds, but is entirely under the control of the coordinator.

However, the actual two-phase protocol does not impose any restrictions on the time between executing the first and second phases. Obviously, the longer this period takes, the more chance there is for a failure to occur and the longer (critical) resources remain locked or isolated from other users. This is the reason why most ACID transaction systems attempt to keep this time frame to a minimum and why they do not work well in environments like the Web.

BTP, on the other hand, took the approach of allowing the time between these phases to be set by the application by expanding the verbs available to include explicit control over both phases of the term, i.e., “prepare,” “confirm,” and “cancel” – what BTP terms an open-top commit protocol. The application has complete control over when it can tell a transaction to prepare and, using whatever business logic is required, it can later determine which transaction(s) to confirm or cancel. This ability is a powerful tool for applications.

Atoms and Cohesions

To address the specific requirements of business transactions, BTP introduced two types of extended transactions, both using the open-top completion protocol:

- **Atom:** An atom is the typical way in which “transactional” work performed on Web services is scoped. The outcome of an atom is

guaranteed to be consistent such that all enlisted participants will see the same outcome, which will be either to accept (confirm) the work or reject (cancel) it.

- **Cohesion:** This type of transaction was introduced in order to relax atomicity and allow for the selection of work to be confirmed or cancelled based on higher-level business rules. Atoms are the typical participants within a cohesion, but unlike an atom, a cohesion may give different outcomes to its participants such that some of them may confirm while the remainder cancel. In essence, the two-phase protocol for a cohesion is parameterized to allow a user to specify precisely which participants to prepare and which to cancel. The strategy underpinning cohesions is that they better model long-running business activities in which services enroll in atoms that represent specific units of work and as the business activity progresses, may encounter conditions that allow it to cancel or prepare these units, with the caveat that it may be many hours or days before the cohesion arrives at its confirm-set: the set of participants that it requires to confirm in order to successfully terminate the business activity. Once the confirm-set has been determined, the cohesion collapses down to being an atom: all members of the confirm-set see the same outcome.

Looking Ahead

In my next article, I'll take a closer look at the architecture of BTP and how XML is involved in it. I'll also look at the Web services stack and how BTP is used.

References

- BTP: www.oasis-open.org/committees/business-transactions
- OMG (1995) “CORBA Services: Common Object Services Specification.” OMG Document Number 95-3-31. March.
- Sun Microsystems Inc. (1999) “Java Transaction API 1.0.1 (JTA),” April.
- Sun Microsystems Inc. (2002) “XML Transactioning API for Java (JAXTX).” www.jcp.org/jsr/detail/156.jsp. ©

Web Services Edge 2003

► Focus on Web Services

Companies that get an early jump on Web Services will be winners in today's challenging landscape. Get ready to take your early pilot projects to the next level!

► Focus on Java, XML, and .NET

Hear from the leading minds in Java how this essential technology offers robust solutions to *i*-technology. Make the right choices. Explore the evolving world of standards, interoperability, application integration, security, and more as you build a collaborative enterprise.

► Focus on the Knowledge You Need

Immerse yourself in information-packed conference sessions. Meet today's *i*-technology leaders, and gather resources in an action-packed Expo!

web services
conference & expo

The Largest
Web Services,
Java, XML, and .NET
Conference and Expo!

March 25-27
Boston, MA
Hynes Convention Center

THE NEXT
FRONTIER:
BEYOND THE FIREWALL

OWNED BY
SYS-CON
MEDIA
PRODUCED BY
SYS-CON
EVENTS

JAVA
DEVELOPERS
JOURNAL

WebServices
JOURNAL

XML
JOURNAL

WebLogic
DEVELOPERS JOURNAL

ColdFusion
DEVELOPERS JOURNAL

PowerBuilder
DEVELOPERS JOURNAL

wireless
BUSINESS TECHNOLOGY

NET
DEVELOPERS JOURNAL

WebSphere
DEVELOPERS JOURNAL

LINUXWEEK
BUSINESS

GE Advisor

Call for Papers
Opens October 15, 2002

SAVE 16% OFF

ColdFusion Developer's Journal

12 Issues for **\$89⁹⁹**

- Exclusive feature articles
- Interviews with the hottest names in ColdFusion
- Latest CFDJ product reviews
- Code examples you can use in your applications
- CFDJ tips and techniques

That's a savings of **\$17⁸⁹** off the annual newsstand rate. Visit our site at www.sys-con.com/coldfusion/ or call **1-888-303-5282** and subscribe today!

↓
EACH CITY WILL BE SPONSORED
BY A LEADING
WEB SERVICES COMPANY

TO REGISTER: www.sys-con.com or Call 201 802-3069

web
services **EDGE**
world tour 2002

\$795

FOR SYS-CON
SUBSCRIBERS
BEST EDUCATIONAL VALUE
OF THE YEAR!

Take Your Career to the Next Level!



SEATING IS LIMITED. REGISTER NOW FOR THE CITY NEAREST YOU! WWW.SYS-CON.COM

Learn How to Create, Test and Deploy Enterprise-Class Web Services Applications

TAUGHT BY THE INNOVATORS AND THOUGHT LEADERS IN WEB SERVICES

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

SPONSOR A CITY!

POSITION YOUR COMPANY
AS A LEADER IN WEB SERVICES
Act today!
Call 201 802-3066 today to ask how.

SHARPEN YOUR PROFESSIONAL SKILLS.

KEEP UP WITH THE TECHNOLOGY EVOLUTION!

"Presented an excellent overview of Web services. Great inside knowledge of both the new and old protocols. Great insight into the code piece."

— Rodrigo Frontecilla

"Very articulate on the Web services SOAP topic and well-prepared for many questions. I've learned a lot from this seminar and I appreciate this seminar for my job. Thank you!"

— Kenneth Unpingco, Southern Wine & Spirits of America

"I liked the overview of Web services and the use of specific tools to display ways to distribute Web services. Good for getting up to speed on the concepts."

— B. Ashton, Stopjetlag.com

Echoed over and over by Web Services Edge World Tour Attendees:

"Good balance of theory and demonstration."

"Excellent scope and depth for my background at this time. Use of examples was good."

"It was tailored toward my needs as a novice to SOAP Web services – and they explained everything."

WHO SHOULD ATTEND:

- Architects
- Developers
- Programmers
- IS/IT Managers
- C-Level Executives
- i-Technology Professionals

TOPICS HAVE INCLUDED:

Developing SOAP Web Services
Architecting J2EE Web Services

IF YOU MISSED THESE...

BOSTON, MA (Boston Marriott Newton) **SOLD OUT!**
WASHINGTON, DC (Tysons Corner Marriott) **SOLD OUT!**
NEW YORK, NY (Doubletree Guest Suites) **SOLD OUT!**
SAN FRANCISCO, CA (Marriott San Francisco) **CLASSES ADDED
SOLD OUT!**

BE SURE NOT TO MISS THESE...

Each city will be sponsored by a leading Web services company

...COMING TO A CITY NEAR YOU

2002
SEATTLEAUGUST 27
AUSTINSEPTEMBER 10
LOS ANGELESSEPTEMBER 19
SAN JOSE AT WEBSERVICES EDGE 2002, WESTOCTOBER 3
CHICAGOOCTOBER 17
ATLANTA.....OCTOBER 29
MINNEAPOLIS.....NOVEMBER 7
NEW YORK.....NOVEMBER 18
SAN FRANCISCO.....DECEMBER 3
BOSTONDECEMBER 12

2003
CHARLOTTE.....JANUARY 7
MIAMI.....JANUARY 14
DALLASFEBRUARY 4
BALTIMORE.....FEBRUARY 20
BOSTON.....MARCH 11

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE LOWEST REGISTRATION FEE.

On May 13th, the San Francisco
tutorial drew a record 601 registrations.

OASIS Forms Committee to Advance UDDI for Web Services

(Boston) – The OASIS standards consortium has organized a new technical committee to advance the Universal Description, Discovery, and Integration (UDDI) specification for Web services. UDDI enables companies and applications to dynamically find and use Web services using preferred applications. The OASIS UDDI Specification Technical Committee is the first committee created within the OASIS UDDI Member Section, a group formed as a result of the recent UDDI.org transition to OASIS.

UDDI joins several Web services standards currently being developed within OASIS. Other specifications include ebXML for electronic business commerce, WS-Security for secure Web services, WSIA for interactive Web applications, WSRP for remote portals, and others.

www.oasis-open.org



Hitachi Introduces SOAP Content Inspector for Secure Web Services

(Waltham, MA) – Hitachi Computer Products (America), Inc., has announced the availability of the Quadrasis/Xtradyne SOAP Content Inspector, a Web services security solution that inspects and secures Simple Object Access Protocol (SOAP) messages and allows enterprise businesses to take Web services outside their Intranets.

Developed by Quadrasis in cooperation with Xtradyne Technologies, SOAP Content Inspector secures SOAP-to-SOAP communication via proxy servers with authentication, authorization, audit, alarm, and policy techniques. It also provides attribute mapping for cross-domain single-sign on (SSO) authorization and can distinguish between standard HTML and SOAP messages in order to process accordingly. Additionally, the SOAP Content Inspector adds a SAML attribute assertion, and then signs and verifies each of the defined SOAP messages.

www.quadrasis.com, www.xtradyne.com

Rogue Wave Software Expands Web Integration Technologies

(Boulder, CO) – Rogue Wave Software, Inc., a global infrastructure software and consulting services company, has announced the availability of Rogue Wave XML Object Link, a new release that supports their Web services integration strategy. XML Object Link addresses the growing need for businesses to smoothly integrate XML and Web services with their existing applications in order to extend their mission-critical systems over a network.

Rogue Wave XML Object Link generates C++ components that represent XML documents, simplifying the task of writing XML-enabled programs by automatically generating C++ code from XML Schemas. It gives development teams the ability to easily handle XML documents that can readily be exchanged with other systems, including those based on Java or Microsoft .NET systems using XML or Web services.

www.roguewave.com

Software AG Adds SOAP Gateway

(Boston) – Software AG, Inc., has announced that its newly released EntireX XML Mediator version 7.1.1.3 includes a SOAP (Simple Object Access Protocol) Gateway to reduce the complexity of managing Web services.

The XML Mediator SOAP Gateway eliminates the coding necessary to call a Web service and generate a response. It also provides increased efficiencies because all Web services can now be managed in one place via an intelligent GUI. In addition to its SOAP Gateway, XML Mediator now comes packaged with a license for TIBCO's XML Transform, a comprehensive design solution for creating and debugging XSLT stylesheets.

EntireX XML Mediator supports solutions for B2B exchange, application synchronization, distributed data interchange, and business reporting.

www.softwareag.com

Crystal Systems Solutions Announces New Release from Mainsoft

(Herzlia, Israel) – Crystal Systems Solutions has announced that its subsidiary, Mainsoft Corporation, the software porting company, has begun shipping Visual MainWin 5, a new product that enables companies to develop enterprise-class C++ applications using Microsoft Visual Studio .NET and rapidly deploy them to multiple Unix platforms.

Developed in close cooperation with Microsoft through the



Visual Studio .NET Integration Program (VSIP), Visual MainWin 5

delivers multiplatform performance, security, and scalability for today's enterprise-class applications, while providing a solid XML foundation for building Web services. It supports the vision of interoperable XML Web services based on Microsoft .NET technology. Visual MainWin 5 has been designed for linear scalability and process isolation, enabling nonstop operation.

www.mainsoft.com, www.crystal-sys.com

Data Junction and Metis Technologies Deliver Real-Time Integration

(New York) – Metis Technologies, Inc., a leader in Web services grid computing, has announced a partnership with Data Junction Corporation, developers of integration solutions addressing new Web services integration issues. This partnership will enable applications to be built with access to many different sources with little or no programming. With Data Junction, the Metis Collaboration Platform (MCP 5) assists enterprises in building applications quickly and easily and provides access to any data type in real time, resulting in lower development cost and the capability to dynamically deploy applications.

While the MCP 5 provides the dynamic data flow framework for large-scale distributed enterprise engagements,

Data Junction integration solutions offer the ability to connect to a limitless number of technologies. Together, the companies will deliver real-time Web services and integration by merging real-time transformations with dynamic distributed computing on a peer-to-peer basis.

www.datajunction.com, www.metistech.com

Siebel Systems Redefines Standard for CRM

(San Mateo, CA) – Siebel Systems, Inc., a provider of multichannel e-business applications software, has delivered hundreds of business processes embedded in Siebel 7.5, the latest release of Siebel eBusiness Applications. Siebel 7.5 enables organizations to leverage and modify proven business processes to deliver customer experiences. With comprehensive support for Web services and Universal Application Network – the industry's first standards-based solution for multiapplication integration – Siebel 7.5 enables the interoperability of Siebel eBusiness Applications with Microsoft .NET and J2EE-based applications.

www.siebel.com

WSJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Actional	www.actional.com/whitepapers/wsblues	800-808-2271	13
Altova	www.xmlspy.com		19
BEA Systems	dev2dev.bea.com/usedworkshop		68
BEA WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	37
engindata Research	www.engindata.com	201-802-3082	34, 35
Gartner	www.gartner.com/us/eai	800-778-1997	31
IBM	www.ibm.com/websphere/integrate		2
Improv Technologies	www.improv-tech.com/gji/download		29
Java Developer's Journal	www.javadevelopersjournal.com	888-303-5282	48
JDJ Store	www.jdjstore.com	800-303-JAVA	43, 49
Macromedia	www.macromedia.com/go/CFMXlight		21
.NET Developer's Journal	www.sys-con.com/dotnet	888-303-5282	55
Rational Software	www.rational.com/offer/javaad11		11
Sitraka	www.sitraka.com/class/SS/ws		17
Sitraka	www.sitraka.com/class/ws		67
Sonic Software	www.sonicsoftware.com/websj		4
SpiritSoft	www.spiritsoft.com/climber		6
SYS-CON Media	www.sys-con.com	888-303-5282	39
SYS-CON Reprints	www.sys-con.com	201-302-3026	46
Web Services Edge World Tour	www.sys-con.com		62, 63
Web Services Journal	www.sys-con.com	888-303-5282	45
WebSphere Developer's Journal	www.sys-con.com	888-303-5282	47

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

WebServices JOURNAL

COMING IN THE
NOVEMBER ISSUE

- e Business Rules: A Perfect Complement to Web Services**
The features and functions of business rules technology can be used to complement Web services applications for enterprises today.
- e Beyond the Hype: What to Do With Web Services Today**
When you look at the number of enterprises that have rolled out successful Web services projects to solve real business problems in a relatively short period of time, it becomes increasingly hard to sound Web services' death knell.
- e Web Services in a Wireless World**
Companies incorporating portal technology as part of their infrastructure need to ensure that their wireless support is truly open, extensible, and standards-compliant.
- e The BTP: Transactions for a New Age?**
The ACID properties of atomic transactions ensure that even in complex business applications consistency of state is preserved.
- e Boundaryless Information Flow: The Role of Web Services**
Stories of Web services successes and challenges offer surprises, affirm a few fears, and highlight specific victories.



Pick 3 or More &
SAVE BIG!

Web Services Journal • Java Developer's Journal
.NET Developer's Journal • XML-Journal
WebLogic Developer's Journal • WebSphere Developer's Journal
ColdFusion Developer's Journal • Wireless Business & Technology
PowerBuilder Developer's Journal

SPECIAL LOW PRICE
when you subscribe to 3
or more of our magazines

RECEIVE YOUR
DIGITAL EDITION
ACCESS CODE
INSTANTLY
WITH YOUR PAID
SUBSCRIPTION

www.sys-con.com/suboffer.cfm



Tyler McDaniel

Tyler McDaniel is the managing director of Analyst Services at Hurwitz Group, an analyst, research, and consulting firm. In this capacity, he steers the analyst resources, to best serve current market needs. Tyler also directs the Application Strategies practice; his specific areas of coverage are enterprise integration and application development tools. TMCDANIEL@HURWITZ.COM

Web Services as the Catalyst for an IT Economic Bounce Back?

A low-cost solution to the integration problem

Since the king discovered that the coffers were bare, or at least shrinking, in 2001, IT spending for big ticket items has been in lockdown mode, while all the king's men work to put everything bought in '99 and 2000 together.

At a time when newer-generation architectures, such as .NET and J2EE, are either being rolled out or gaining deeper enterprise acceptance, the push is on to make better use of existing computing assets – hammer down and be sure that every last ounce of computing power is squeezed out of those main-frame applications, those hordes of custom-built applications, the expansive packaged applications. Above all, cut

the operational cost of every e-business activity we're involved in. Return on investment is a must.

Enter the king's alchemist with vials of new, strange-looking Web services technology. Or perhaps it's not so strange-looking – Hurwitz Group's research in early 2002 showed that Web services adoption was moving along quicker than pundits expected, given the lockdown on budgets. The adopters are tackling enterprise integration as a primary problem. Developers are very willing to experiment with the new stuff, and C-level decision makers are green-lighting Web services. But this new alchemy shouldn't be surprising. In fact, it demonstrates a return to basics in IT departments – not yet the dynamic world of Web services flying all about the network, being presented, consumed, and joined for new revenue generation.

We took our eyes off the ball in '99 and 2000 in terms of the types of projects we took on and how we declared success on those projects. We brought in slick and visionary technology to extend our enterprise services to the many market opportunities just at browser's reach. But just because new technology walked in the door, what was already present didn't suddenly disappear. We spent too long pulling apart our core, existing business processes to grab Internet success at the expense of having efficient back-office fulfillment. We walked away from setting up business success parameters (ROI) for new technology acquisitions. But isn't this the original sin of technology – a promise for totally new value that comes at any price and demolishes how we previously did business? It's real work to make new technology work with the multiplicity of incumbent technology. It's even more work to roll out reliable and effi-

cient business processes that adequately reflect the value of the technology in the first place. Finally, it requires fiscal due diligence to ensure that the "return" is in the bag.

We need better strategies and technologies to deal with this issue. Web services is one of the technologies around it, and Hurwitz Group is seeing executives starting to form an actual long-term strategy. With the smallest additive, we can trigger an otherwise static mixture – this is the function of the catalyst. How then can Web services be a catalyst?

First, look at our static mixture. One of the components is frozen budget expenditures. The other components are legacy applications and data, large packaged applications, portal technology, home-grown modern applications, all kinds of things. This mixture has put us into some degree of stasis: What does what? What is it connected to? How does this work together? How can I get this information from point A to point B for the lowest cost and least amount of pain? How much have I spent on all this?

The issue for Web services is not whether it's the greatest technology since yesterday's greatest thing. The issue centers around the use and value of Web services. Web services is precisely the catalyst we need to spark intelligent spending on making our IT assets work better together. Let's not forget that computing heterogeneity is a reality, and we're in a time when market pressures are dictating that our heterogeneous applications work more cohesively together. How can we accomplish this? By having a business strategy that reflects the reality of how we use our IT. This means looking at the ROI and the return on assets that any software gives us. Web-services technology is a low-cost investment that helps solve a key problem: integration of various computing assets. Web services isn't the *only* thing, but it gives us a cost-managed means to look closely at how effectively our applications work together. Web services can give us the confidence to truly address our IT integration challenges in a pragmatic fashion – one that delivers business results.

The catalyst sparks the mixture so we're not afraid to spend our budgets on things that truly need to be fixed in our IT laboratory. Use of Web services technology for integration gets us back on track toward spending our IT dollars on solving today's problem as opposed to building tomorrow's problem, which we were guilty of not so long ago. As these implementations mature, Hurwitz Group sees BPM (business process management) taking an even greater role in the enterprise, but that's another subject. For now, Web services can ignite a return to business value spending for IT by keeping us focused on projects that unify application fabrics and business processes. ©

Sitraka

www.sitraka.com/jclass/ws

BEA Systems

dev2dev.bea.com/useworkshop