

Web Services

.NET J2EE XML JOURNAL

March 2005 Volume 5 Issue 3

SOA Command and Control

Taking SOA to the next level
pg.10

The Role of EII in SOA

The perfect marriage
pg.22

Accessing Resources: New Web Service Patterns for a Service-Oriented World

pg.38

WEB SERVICE MANAGEMENT ARCHITECTURE PATTERNS

Balancing management against complexity

pg.26

RETAILERS PLEASE DISPLAY
UNTIL APRIL 29, 2005

\$6.99US \$7.99CAN



**MARCH
BUZZ**

SOAPtest®

Keep Your Skills Ahead of the Crowd

Keeping your IT skills ahead of the crowd is not as difficult as most people fear. Staying on top of the trends may seem like a daunting task if, like most people, you assume that each new technology is a completely new invention that you must learn from the ground up. Fortunately, nothing is really all that new. Inventors typically create new technologies by studying existing technologies, then building upon them in ways that extend and improve them. 100% new technological advancements are very rare.

Inventors almost always leverage legacy technologies as they invent new ones. Why not leverage your own knowledge of those legacy technologies as you try to learn about the new inventions? To learn about new technologies as painlessly as possible, consider how each new advancement is similar to what you already know.

For example, consider Web services. Web services are a new trend, but — at a technological level — the parts of a Web service are not all that unique. Web services are based on remote procedural calls — messages sent to a server, which calls the requested function. RPCs were developed years ago, and are hardly a new concept. Really, the only "new" thing in Web services is the standard that is being used to write the application. If you break down Web services in this way, it's easy to learn about them. To continue with this process, you might next explore the payload requirements, the process for determining what function to call, and how the call works. As you can imagine, it's a lot more efficient — and interesting — to learn about a new technology based on its relation to familiar technologies than to learn about it by reading the specification cover to cover.

As always, the devil is in the details. But most details are critical only if you want to specialize in a given technology. For instance, if you want to specialize in Web services, you need to familiarize yourself with the details of Web service development. In that case, your next step would be to learn how to format the messages, how to expose Web services, and so on.

— **Adam Kolawa, Ph.D.**
Chairman/CEO of Parasoft

It's automated. It's fast. And it's the **most versatile** Web Services testing tool.



SOAPtest® enables you to deliver better Web Services in less time.

The simple fact is, no other Web service testing product can do what SOAPtest can do. From verifying functionality and performance to ensuring security and interoperability, SOAPtest automates all critical testing processes across the entire lifecycle of your Web service.

But don't take our word for it.... Go to www.parasoft.com/soaptest_WSJ and try it for free — no commitment, no obligation. If you like it (and we suspect you will) just let us know. We'll be more than happy to help you and your development team get up and running.

For Downloads go to www.parasoft.com/soaptest_WSJ

Email: soaptest_WSJ@parasoft.com Call: 888-305-0041 x1209



Features

- WSDL schema verification and compliance to standards
- Automatic test creation using WSDL and HTTP Traffic
- Data-driven testing through data sources (Excel, CSV, Database Queries, etc)
- Scenario-based testing through XML Data Bank and Test Suite Logic
- Flexible scripting with Java, JavaScript, Python
- WS-I Conformance: Basic Profile 1.0
- WS-Security, SAML, Username Token, X.509, XML Encryption, and XML Signature support
- WS-Security Interop testing emulator
- MIME Attachment support
- Asynchronous Testing: JMS, Parlay (X), SCP, and WS-Addressing support
- Windows Perfmon, SNMP, and JMX monitors
- Detailed Report generation in HTML, XML and Text formats
- Real-Time graphs and charts

Benefits

- Uniform test suites can be rolled over from unit testing to functional testing to load testing
- Prevent errors, pinpoint weaknesses, and stress test long before deployment
- Ensure the reliability, quality, security and interoperability of your Web service
- Verify data integrity and server/client functionality
- Identify server capabilities under stress and load
- Accelerate time to market

Protocol Support

- HTTP 1.0
- HTTP 1.1 w/Keep-Alive Connection
- HTTPS
- TCP/IP
- JMS

Platforms

Windows 2000/XP
Linux
Solaris

Contact Info:

Parasoft Corporation
101 E. Huntington Dr., 2nd Fl.
Monrovia, CA 91016

www.parasoft.com

Nothing beats our racks

Absolutely nothing



ev1servers.net
an ev1.net company

CARRIER CLASS DATA CENTERS

- Highly Secure Guarded Facility
- 24/7/365 Network Operations Center
- 24/7/365 Technical Support
- Redundant Conditioned Air Systems
- Redundant Fiber Entry Points
- Multiple Uninterruptible UPS Systems
- Multiple 1250 KW Generators Onsite
- VESDA Early Warning Smoke Detection

Robert Marsh, Head Surfer

START YOUR OWN WEB HOSTING BUSINESS TODAY!

from
\$299*
Instant Activation!

Dedicated Server

Dual Xeon 2.4 GHz
2 GB RAM • 2 x 73 GB SCSI HD
Remote Console • Remote Reboot
2000 GB Monthly Transfer Included

Over 20,000 Servers!

1-800-504-SURF | ev1servers.net

PLESK7
RELOADED
Preferred Control Panel

IP Compliant. Price subject to change. Quantities Limited.
*Per month. Set-Up fees apply. See web site for complete details.

InsideWSJ

Web Service Management (WSM) Architecture Patterns

Balancing management against complexity

By Ramy Abaas

26

SOA Command and Control

Taking SOA to the next level

By Dan Foody

10

The Role of EII in SOA

The perfect marriage

By Tim Mathews

22

From the Editor How Important Is Security?

Sean Rhody 7

Industry Commentary How the BlackBerry Changed My Life

Michael Mosher 8

WSJ:Security You Just Don't Get It

Bridging the schism between development and security priorities in XML Development

Andrew Nash 14

WSJ: BPEL BPEL to the Rescue: A Real-World Account

The backstory of a flexible and agile application

Robert Wales, Kevin Geminiuc, Peter Zadrozny and Arthur Wang 18

WSJ: SOA Understanding Coupling in the Context of an SOA

Comparative Architecture
David Linthicum 34



Accessing Resources: New Web Service Application Patterns for a Service- Oriented World

PAUL LIPTON &
IGOR SEDUKHIN

38

Stylus Studio 6 from Progress Software

BRIAN BARBASH

44

An Easy Introduction to XML Publishing

PG BARLETT

46

Transforming XML- to-XML or -Text or -HTML

DEEPAK VOHRA &

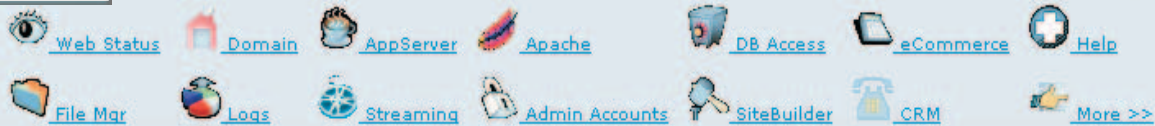
AJAY VOHRA

48

Process SOAP with VTD-XML

JIMMY ZHANG

54



Home Help
 [+ Site Management
 [+ Apache
 [+ Database
 [- App Server
 Configure
 Virtual Hosting
 JBoss
 Tomcat Manager
 Tomcat Admin
 JRun MC
 ColdFusion Admin
 JDKs
 Connector
 Restart
 System Logs
 Heartbeat
 [+ Applications
 [+ Advanced
 more...

:: Application Server Configuration ::

[Apache WebServer Configuration](#) [App Server Virtual Hosting](#) [Heartbeat](#) [Load-Balancing & Clustering](#) [Help ?](#)

AppServer Type: JAVA J2EE

JAVA J2EE

Set Type

Present AppServer

Tomcat 5.5.4

AppServer Virtual Path(s)/Extension(s)

/*.jsp /servlet/ /manager/ /*.do

:: Select Application Server ::

Tomcat

5.5.4

Install AppServer

Overwrite Old App S

:: Customize Application Server ::

Localhost Bind Port Range 4671-4679

AJP/HTTP Bind Address localhost:4671

Dedicated IP/Host Address myaccount.webappcabaret.net

App Server Home Directory /usr/ngasi/contexts/myaccount/

Java Virtual Machine (JVM) JDK1.5.0

Connector

AJP

Customize

Remove

Imagine a hosting company dedicated to meet the requirements for complex web sites and applications such as those developed with Java J2EE.

At WebAppCabaret our standards based process and tools make deploying Java J2EE applications as easy as a point-and-click.

We call it **Point-and-Deploy Hosting**.

Our advanced NGASI Web Hosting management Control was designed for the hosting and management of complex web sites and applications thus cutting down on maintenance time.

Backed by an experienced staff as well as a **Tier 1 Data center** and network. Our network is certified with frequent security audits by reputable security firms.

All hosting plans come with **advanced tools** to manage your application server and Apache web server. Not to mention the other features, such as virus-protected email, bug tracking, and many more portal components.

Complete power and control at the tip of your fingers. We take care of the system and hosting infrastructure so you can concentrate on development and deployment of your application. That is **real ROI**.

Log on now at <http://www.webappcabaret.com/jdj.jsp> or call today at 1.866.256.7973





XML'S ENDLESS POSSIBILITIES,

NONE OF THE RISK.

FORUM XWall™ WEB SERVICES FIREWALL - REINVENTING SECURITY

SECURITY SHOULD NEVER BE AN INHIBITOR TO NEW OPPORTUNITY: FORUM XWall™ WEB SERVICES FIREWALL HAS BEEN ENABLING FORTUNE 1000 COMPANIES TO MOVE FORWARD WITH XML WEB SERVICES CONFIDENTLY. FORUM XWall REGULATES THE FLOW OF XML DATA, PREVENTS UNWANTED INTRUSIONS AND CONTROLS ACCESS TO CRITICAL WEB SERVICES.

VISIT US AT WWW.FORUMSYS.COM TO LEARN MORE ABOUT HOW YOU CAN TAKE YOUR NEXT LEAP FORWARD WITHOUT INCREASING THE RISKS TO YOUR BUSINESS.



FORUM SYSTEMS™ — THE LEADER IN WEB SERVICES SECURITY



INTERNATIONAL ADVISORY BOARD

Andrew Astor, David Chappell, Graham Glass, Tyson Hartman, Paul Lipton, Anne Thomas Manes, Norbert Mikula, George Paolini, James Phillips, Simon Phipps, Mark Potts, Martin Wolf

TECHNICAL ADVISORY BOARD

JP Morgenthal, Andy Roberts, Michael A. Sick, Simeon Simeonov

EDITORIAL EDITOR-IN-CHIEF

Sean Rhody sean@sys-con.com

XML EDITOR

Hitesh Seth

INDUSTRY EDITOR

Norbert Mikula norbert@sys-con.com

PRODUCT REVIEW EDITOR

Brian Barbash bbarbash@sys-con.com

.NET EDITOR

Dave Rader davidr@fusiontech.com

SECURITY EDITOR

Michael Mosher wsjsecurity@sys-con.com

RESEARCH EDITOR

Bahadir Karuv, Ph.D. Bahadir@sys-con.com

TECHNICAL EDITORS

Andrew Astor aastor@webmethods.com

David Chappell chappell@sonicsoftware.com

Anne Thomas Manes anne@manes.net

Ajit Sagar ajitsagar@sys-con.com

Mike Sick msick@sys-con.com

Michael Wacey mwacey@csc.com

EXECUTIVE EDITOR

Gail Schultz gail@sys-con.com

EDITOR

Nancy Valentine nancy@sys-con.com

ASSOCIATE EDITOR

Jamie Matusow jamie@sys-con.com

ASSISTANT EDITORS

Natalie Charters natalie@sys-con.com

Seta Papazian seta@sys-con.com

ONLINE EDITOR

Martin Wezdecki martin@sys-con.com

PRODUCTION

PRODUCTION CONSULTANT

Jim Morgan jim@sys-con.com

LEAD DESIGNER

Richard Silverberg richards@sys-con.com

ART DIRECTOR

Alex Botero alex@sys-con.com

ASSOCIATE ART DIRECTORS

Louis F. Cuffari louis@sys-con.com

Tami Beatty tami@sys-con.com

Andrea Boden andrea@sys-con.com

CONTRIBUTORS TO THIS ISSUE

Ross Altman, Brian Barbash, Dave Chappell, Dan Dube, Jim Gabriel, Labro Dimitriou, David Linthicum, John O'Shea, Sean Rhody, Indronil Deb Roy, Bill Roth, Sekhar Sarukkai

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 © 2005 by SYS-CON Publications, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system without written permission.

For promotional 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 covered in Web Services Journal.



How Important Is Security?

Recently, Paris Hilton's cell phone was hacked, and all her contact information was released on the Internet. Although I wasn't important enough to rate a listing, many other celebrities were apparently flooded with phone calls after their private numbers became oh-so-public. While the incident didn't involve Web services, it certainly did involve security, or rather a failure of security. And once again, security itself is the focus of this month's issue.

Security has been a topic of this magazine almost since its inception – and we tend to get proposals regarding security on a year-round basis. Initially, when the number of Web services specifications could be counted on one hand, security was a huge concern, and a serious impediment for enterprise adoption of Web services (and rightfully so when the only security provided was SSL).

Over the past several years however, that situation has been remedied by a large number of standards, most of which have been implemented in some fashion. These include WS-Security and things like digital signatures. If you dig into the security area you will find there is actually a large number of things you can do to protect your Web service. Some are targeted at the message and some at authentication; some even look at the content by validating that the message is a valid request.

Part of the initial resistance to Web services revolved around the need for security, and that did make sense. Now however, when we have a plethora of standards for combating the deficiencies of the original specifications, it's possible to put the general concern that Web services cannot be secured to rest.

Companies can now concentrate on Web services and security from an opportunistic standpoint. Whereas previously IT professionals had to fight the battle of "Web services aren't secure," they can now point to the standards and tools and say, "How secure do you want them to be?"

The nice thing about the various Web services standards is twofold; they are layered, and you can choose whether or not to use many of them. The layered approach makes it easier to design and build Web services. Later, someone can decide how to secure those services, or how to deploy them, or manage them. Not having to decide at design time



WRITTEN BY
SEAN RHODY

how to implement the various strategies (or even whether to implement them) is a key advantage in Web services usage. Naturally, it's not a new concept – the application server people use it all the time with their deployment descriptors.

Regardless of where the idea came from, the separation of layers is a very useful thing. Web services security needs to run the gamut from absolutely none nec-

essary to absolutely paranoid, with most services falling in between. That's because Web services are an edge technology, and sometimes the edge must be protected, but sometimes you want to expose it.

There are some well-known Web services such as Google and Amazon, which need little in the way of security. It isn't important to the use of these services to establish identity, so applying SAML would be a needless overhead, and would likely reduce the number of users, rather than increase them. Many services are like this – more or less wide open because the information transferred is of value only to the user. The fact that I search for "books by Sean Rhody" is interesting only to Amazon, who processes the request, and to myself. Well, and to my legions of fans of course. The point though is that my identity is irrelevant and securing the service is fairly unimportant because what is transferred has value only between Amazon and me.

On the other hand, if I were transferring money between my checking account and my savings account, I would want very strong security. I'm not rich, but I want what money I have to stay mine. I also don't want anyone else knowing how much money I do have – I get enough solicitations as it is despite the Do Not Call act.

We now have the tools to secure Web services as needed, without choking the services that can be designed without the need for heavy-duty security. And that's comforting, especially when we see other services being hacked and phone numbers being distributed on the Internet. Oh, and Paris – why haven't you called back? ☺

About the Author

Sean Rhody is the editor-in-chief of *Web Services Journal*.

He is a respected industry expert and a consultant with a leading consulting services company.

■ ■ ■ sean@sys-con.com

PRESIDENT AND CEO

Fuat Kircaali fuat@sys-con.com

VP, BUSINESS DEVELOPMENT

Grisha Davida grisha@sys-con.com

GROUP PUBLISHER

Jeremy Geelan jeremy@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

NATIONAL SALES & MARKETING MANAGER

Dennis Leavey dennis@sys-con.com

ADVERTISING MANAGER

Megan Mussa megan@sys-con.com

ASSOCIATE SALES MANAGERS

Kristin Kuhnle kristin@sys-con.com

Dorothy Gil dorothy@sys-con.com

Kim Hughes kim@sys-con.com

SYS-CON EVENTS

PRESIDENT, SYS-CON EVENTS

Grisha Davida grisha@sys-con.com

NATIONAL SALES MANAGER

Jim Hanchrow jimh@sys-con.com

CUSTOMER RELATIONS/JDJ STORE

CIRCULATION SERVICE COORDINATORS

Edna Earle Russell edna@sys-con.com

Linda Lipton linda@sys-con.com

Monique Floyd monique@sys-con.com

sys-con.com

CONSULTANT, INFORMATION SYSTEMS

Robert Diamond robert@sys-con.com

WEB DESIGNERS

Stephen Kilmurray stephen@sys-con.com

Matthew Pollotta matthew@sys-con.com

ACCOUNTING

FINANCIAL ANALYST

Joan LaRose joan@sys-con.com

ACCOUNTS PAYABLE

Betty White betty@sys-con.com

ACCOUNTS RECEIVABLE

Steve Michelin smichelin@sys-con.com

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

1-201-802-3012

1-888-303-5282

For subscriptions and requests for bulk orders, please send your letters to Subscription Department

Cover Price: \$6.99/issue

Domestic: \$69.99/yr (12 issues)

Canada/Mexico: \$89.99/yr

All other countries: \$99.99/yr

(U.S. Banks or Money Orders)

Worldwide Newsstand Distribution

Curtis Circulation Company, New Milford, NJ

Newsstand Distribution Consultant:

Brian J. Gregory / Gregory Associates / W.R.D.S.

732 607-9941 • BJGAssociates@cs.com

For list rental information:

Kevin Collopy: 845 731-2684,

kevin.collopy@edithroman.com;

Frank Cipolla: 845 731-3832,

frank.cipolla@epostdirect.com

Sys-con Publications, Inc., reserves the right to revise, republish and authorize its readers to use the articles submitted for publication.



How the BlackBerry Changed My Life

For Christmas this year, I got just what I asked for – a new gadget. I collect gadgets, so this was nothing new, but this gadget has altered the way I live; I now live with a BlackBerry on my hip.

I have resisted the temptation to “converge” my PDA and phone for any number of reasons. The previous generations of so-called “smartphones” either didn’t seem very smart or weren’t much like phones. In fairness, my point here isn’t that Research in Motion Ltd., the makers of the BlackBerry, has single-handedly changed the face of converged communications devices. Many of the new generation of “smartphones” have overcome the shortcomings of their predecessors. It’s just that I finally found the integration of phone and PDA something worth investing in. It seemed seamless.

You might be wondering at this point if you picked up the wrong magazine. What could this have to do with Web Services and why is it in the security issue? The answer is the new paradigm.

The term “new paradigm” gets thrown around a lot. I’ve used it a lot myself, but with my new Christmas gadget, I came to a whole new appreciation of what “a new paradigm” can mean.

First of all, I found out what “connected” really means. It’s amazing to be able to get an e-mail while touring Edinburgh Castle, though not particularly appealing to my travel companions. I also got better acquainted with the “always-on” nature of business these days and the odd schedules of some of my co-workers. Nothing’s more startling than getting notified at 2am that a new e-mail has arrived on one of the four accounts you have feeding your BlackBerry (I quickly learned to turn those notifications off – especially at night). All in all, though, I’ve found my new gadget irresistible. It’s made me more productive at work and more connected with my family, and I foresee even greater potential.

Since my job is security, I also thought about my new gadget’s other implications and my new way of life. My e-mail is now being propagated to multiple servers and across the airwaves. BlackBerry’s manufacturer and the service providers claim its



WRITTEN BY

MICHAEL MOSHER

secure, but it still gives me pause. I realize that the same “push” technology that gets my important e-mail delivered to my hip wherever I am in an instant gets spam there just as fast.

And I was struck by the parallels with Web Services – another useful “new paradigm” with potential pitfalls.

Web Services are changing the way we “live”: how we build systems, how we tie them together. This year could be a real breakthrough for the Web Services world, one where Web Services become more useful and ubiquitous like in “smartphones.”

As these technologies “come into their own,” we must be cautious not to be so overcome by the euphoria of an addictive “gadget” that we fail to consider the hazards. Just as I have had to temper my BlackBerry use to take into account social or safety considerations (e.g., don’t drive and type), we should examine our use of Web Services to make sure we are giving proper considering to performance and security. And, just as the appearance of spam on my new device has shown, moving to a new paradigm doesn’t mean that the old problems go away. In fact, layering new paradigms on old ones may make solving the old problems more critical and complicated.

Luckily, we are fortunate that the Web Services world has a community of people and organizations that has taken the lead in overcoming issues like security. The standards and tools that have already been created give us a number of ways to work in this new paradigm. We just need to embrace them and use them effectively.

Now, if only there was a well-defined standard for reading e-mail while driving... ☺

About the Author

Michael Mosher is technology director of the CSC Consulting Business and Technology Risk Management practice. He specializes in security architecture and security strategy, and has designed security solutions for Fortune 500 clients in financial services, manufacturing, energy and healthcare. Michael has a broad background in government and commercial security, including six years as a special agent with the U.S. government, investigating computer and white-collar crimes.

■ ■ ■ wsjsecurity@sys-con.com

Is your SOA really agile?

Building, testing and maintaining SOA solutions requires a flexible and collaborative approach to diagnostics. Finger pointing and confusion occur when all parties do not have a complete and common understanding of a problem. Mindreef® SOAPscope® connects developers, testers, support personnel, and operations by combining the testing and diagnostic tools for each discipline and making it easy to share complete problem data. SOAPscope is the only end-to-end diagnostic system for Web services.



Developers • Testers • Operations • Support

Mindreef SOAPscope

Test Agile testing approach for both developers and testers allows what-if scenarios without coding or scripting. Includes the industry's most complete interoperability suite and the flexibility to really test the edge cases.

Capture Easily capture and assemble complete problem data into SOAPscope's workspace. Capture SOAP messages off the wire, test cases, or reproduced problems. Capture a snapshot in time of a WSDL including all related resources.

Share Exported workspaces support sharing among team members preserving complete problem data including messages, WSDLs, and notes. Sharing workspaces is a great way to support web service users.

Solve Diagnostic suite of analysis, comparison and visibility tools simplifies the understanding of complex XML, yet are powerful enough for the industry's most protocol-savvy experts.

"SOAPscope has been invaluable in testing and diagnosing our enterprise scale Web Services. The new Workspace feature in SOAPscope is an excellent way to support customers and collaborate as a team. A must have for every WS-Developer"

*Simon Fell
Senior Member of
Technical Staff at
salesforce.com and
author of PocketSOAP*

Try SOAPscope **FREE** at www.Mindreef.com/tryout

© Copyright 2005, Mindreef, Inc. The names of companies and products mentioned herein may be the trademarks of their respective owners. This product uses Hypersonic SQL. This product includes software developed by the Politecnico di Torino and its contributors.

Mindreef, Inc.
www.Mindreef.com

SOA Command and Control

Taking SOA to the Next Level

■ One of the ongoing challenges for business today is finding ways to do more with less. Companies are under relentless pressure to deliver products and services to market faster, better and cheaper than ever before. Investments in information technology are expected to drive the business forward, not only in terms of gaining efficiencies and increasing responsiveness, but in creating new top-line opportunities.

Ironically, most corporate IT organizations allocate 75%-85% of their annual budget just to “keeping the lights on.” At the same time, research shows that a typical server’s average utilization is 15%-25%. While these figures may not reflect your particular organization, they highlight the fact that there’s a great deal of inefficiency in IT today, consuming large portions of budgets that could be put to better use for more strategic initiatives.

What’s driving this inefficiency?

While there is no single cause, much of it results from mergers, acquisitions or functional silos in the organization – all of which contribute to a bloated IT infrastructure, rife with overlapping and duplicate applications



WRITTEN BY
DAN FOODY

and systems. In the face of tight deadlines and the ongoing demands of the business, IT often opts to live with these redundancies – integrating systems where necessary to keep them consistent. While this approach may result in quick time-to-delivery, it severely inflates recurring costs.

While some IT organizations may be content with this status quo, executive management sees “aligning IT with business” as its number one IT-related concern. However, with as little as 15% of the budget available to service the changing needs of the business, IT just doesn’t have room to maneuver. It’s this realization that’s leading many organizations toward a service-oriented architecture (SOA), an approach to architecting the IT infrastructure that eliminates redundancy and

accelerates project delivery via consolidation and reuse of services, often referred to as Web Services.

In the past, SOA adoption was hampered by a lack of agreement on a “common language” for applications to communicate with each other. Happily, Web Services in the form of XML, SOAP, WSDL and other related standards has begun breaking the logjam that made SOA impractical and so has become the foundation technology for building an SOA.

Of course, using Web Services is not a panacea. Though many companies are launching tactical Web Services projects, few are reaping their full value. The value of Web Services escalates when organizations harness the economies of scale of consolidation and reuse. In other words, when customers move from an ad hoc use of miscellaneous collections of Web Services to a more formalized SOA, the value of those services rises dramatically.

Case in point: If you have just one Web Service, but it’s used by five different applications due to consolidation or reuse then you have significantly reduced total cost of ownership (TCO) – the essence of SOA. How is that TCO benefit achieved? Every time you can reuse an existing service, whether proactively by planned reuse or reactively by decommissioning redundant services, you buy and operate fewer machines, license and operate less software and manage fewer individual service

Share Your SOAP

Now

LEARN

DEBUG

TEST

TUNE

SHARE

SOAPscope[®] 4.0

Try it free at www.mindreef.com/tryout

Web Services Diagnostics

lifecycles – all significant quantifiable benefits. So the number of times you reuse services in different projects – not the number of services, but, the number of different uses of services – is a quick measure of success in SOA adoption.

SOA Challenges

The fundamental goal of Web Services and SOA is to improve ROI and TCO. However, the use of Web Services doesn't guarantee your results. There are a number of challenges, many of which are organizational and not strictly technical, that your company needs to address proactively to achieve measurable benefit.

One problem is that standards don't guarantee interoperability. By their very nature, standards are designed to support many different uses across many different types of organizations.

For example, security standards allow many different types of credentials including how a user is identified. If the sender and receiver don't both understand the same types of credentials, they can't talk together. The common language breaks down.

Narrowing down the set of acceptable ways to use standards from all the available options is something your organization needs to address with policies and procedures. Of course, many of these policies and procedures will naturally need to change over time as regulatory or other compliance requirements change, so when thinking about TCO you need to factor in the inevitable cost of change.

A second challenge is determining who pays for a service shared by many applications. With traditional line of business applications, figuring out who pays is easy – since one team owns everything. For a shared service, ideally each line of business should pay proportional to their use. Those who use it most should pay the most. Essentially this is a transfer-pricing model. Beyond the accounting considerations of how to implement transfer pricing policies and procedures effectively across lines of business, you must also consider how to track usage by each line of business accurately – if you can't measure usage, you can't charge for it!

A third challenge is ensuring that service levels are met. To end users, the customers of the IT infrastructure, the order management application is still the order management application whether it's built as a monolith or it

leverages shared services. In either case it must meet the same expectations of performance, availability and functionality. Conversely, if the same user uses two different applications, he may have different expectations of each – even if both applications leverage the same set of shared services.

While shared services serve many applications, they need to be approached in the context of the applications they serve to ensure the “customer experience.” For example, changes may lead to unintended consequences. For instance if an application using a service has an unexpected load increase it consumes the service's capacity most likely reducing the service level for all the other applications. The use of shared services means that understanding the nature of the interdependencies plays a key role in understanding the effects of any changes.

A final challenge is that, as one team begins leveraging services built by other teams, they no longer have visibility into all the moving parts that make up their overall application. This factor has a number of implications. For example, how does an application team ensure that its overall application is secure, when it's built using services from other teams? If its application is exposed to partners and customers via the Internet, how does the team ensure that the services it's using aren't vulnerable to malicious attacks aimed at stealing or corrupting information? Maybe the services the team are leveraging haven't been built with the same requirements in mind. Similarly, when something goes wrong, how can you figure out if the problem is due to the application itself, or to services leveraged by the application? And who dictates security and business policy as it relates to shared services?

SOA Command and Control

While the core Web Services standards successfully address the mechanics of letting applications to talk to one another, successful SOA implementations mean addressing the challenges that lie beyond the pure mechanics of communication. The complexities are numerous: stakeholders with different agendas, policies with cross-functional implications, service levels that must be maintained at all costs, complex inter-relationships between services and no lines painted on the data center floor to connect any of the dots.

Left to grow on its own, a network of Web Services will quickly degenerate into a tangled spaghetti of brittle, single-use integrations and fail to achieve the economies of scale or the cost and flexibility benefits of SOA.

These challenges call for a new breed of solution – SOA Command and Control – that addresses the various technical, business, and organizational requirements and coalesces all pertinent knowledge in the SOA into a form that's understandable and actionable.

There are five key imperatives of SOA Command and Control: Align, comply, observe, respond, optimize. These five imperatives encapsulate the required, and the associated value, of SOA Command and Control. Let's take a look at each of them.

Align

IT is successful only if the business is successful, so IT must always be aligned with the business. SOA Command and Control must allow you to measure SOA activity against your business objectives to understand its current impact on the business, to determine how it's trending, and to predict where it will go in future.

In addition, SOA Command and Control should let you customize or tailor the services offered by your SOA quickly and easily to address tactical business opportunities. With SOA Command and Control you can:

- Understand which of your most important consumers, regions or divisions are getting the best service.
- Provision services to meet the unique requirements of a new high-value consumer.
- Plan and execute a migration to a new version of a service so that your standard customers roll onto the new version before your most important customers.

Comply

True SOA Command and Control empowers stakeholders to move from a passive role to an active role of driving policy changes immediately and automatically across the organization. In addition, stakeholders gain visibility as to where and when the policies and procedures are being applied and/or violated. With SOA Command and Control you can:

- Uniformly apply and enforce a security policy across many services without a service-by-service development effort.

- Become compliant with changed privacy regulations.
- Define what service levels can be promised to different consumers.

Observe

By definition, SOA Command and Control provides both detailed and at-a-glance visibility into the inner workings of your SOA at any point in time – automatically, without expensive, time-consuming manual configuration. This facility lets you understand SOA-wide patterns and trends that would never be uncovered with solutions that provide simple statistics and only threshold, rule-based or predictive alerting. With SOA Command and Control you can:

- Understand what your service level is right now, broken down by geographical region, customer segment and service.
- Discover who depends on which service, and what their different usage patterns are.
- Determine what “hotspots” are most vulnerable to attack or which will be the first to fail as loads increase.

Respond

When a symptom of a problem is detected – whether it's a functional issue with a newly rolled-out version of a service, a malicious attack or a performance issue – you need to be able to determine the root cause of the problem and how to respond to it effectively. Root cause analysis is important in an SOA because symptoms rarely appear at the location of the root cause – and the root cause may be a service owned by a different group. With SOA Command and Control, you can accurately and automatically determine the root cause of problems, without expensive, time-consuming manual configuration of rules or relationships. And, once you determine the root cause, you can respond in one of many different ways such as notifying administrators, black-listing users, rolling back service changes, rationing capacity, or modifying documents in transit. These responses can be triggered manually, fully automated or even manually overridden when automated responses don't produce the desired result. With SOA Command and Control you can:

- Roll back consumers to use a previous version of a service if a problem is encountered during the migration.

tered during the migration.

- Enrich or cleanse incomplete or ill-formed documents rather than reject them.
- Continue to meet the service-level expectations of your most important customers while under unexpectedly heavy loads.

Optimize

As with any IT infrastructure, services have a finite capacity to process consumer requests. Determining the capacity requirements of services is especially complicated because each consumer has a different pattern to use with different kinds of requests, and different peak usage periods. And, as new consumers come online, they consume capacity from the service and potentially affect the service level of everybody else. SOA Command and Control lets you both proactively and reactively optimize the allocation of scarce service resources. With SOA Command and Control you can:

- Offload resource intensive processing for frequently changing policies.
- Predict what changes will cause your service to become a bottleneck, breaching the service-level expectations of consumers.
- Ration capacity, giving more capacity to more important consumers or reserving specific servers for high-value customers.

Achieving SOA Command and Control

Traditionally the five imperatives of SOA Command and Control have been addressed directly by development or line-of-business application teams as part of the applications they build. Because application teams are intimately familiar with their own applications, the strategy of building security or operational policy into the applications was often effective.

In this environment, the role of cross-functional IT stakeholders such as security officers or deployment architects is to write documents that outline the policies that application teams must follow. Unfortunately, by delegating the implementation of policy to each application team, you lose economies of scale and control over whether the policy has been correctly understood and implemented. Worse still, to change policy requires a complex one-off project for each and every application or service.

Because of the limitations of this model, products have emerged that attempt to empower cross-functional teams to gain complete control over policies that span applications and services. Unfortunately, while this model appears effective on the surface, it breaks down because these cross-functional teams don't understand the business context of the different applications and services. For example, if a change to privacy policy requires that all personal identities be signed and encrypted, the cross-functional teams will have no idea what data in any of the tens to thousands of documents are classified as personal identities.

What is required instead is a fundamentally new approach to addressing this challenge. Instead of providing tools that let one team or another be responsible for every aspect of the problem, you need a solution that enables the different stakeholders to collaborate effectively – each maintaining control over its own area of responsibility, while seamlessly sharing the knowledge necessary to let the others do their job once.

With an effective SOA Command and Control infrastructure, policies can not only be defined once, centrally, but also automatically enforced in the fabric of the network itself. The capabilities of an effective SOA Command and Control platform lets organizations bypass the knowledge gap and successfully achieve the economies of scale as well as the critical cost, time and flexibility benefits of SOA. 

About the Author

As chief technology officer at *Actional*, Dan Foody leverages his extensive hands-on experience in enterprise systems integration software to ease integration through Web Services. He's an active participant in the Web Services standards community, including WS-I and OASIS, where he spearheads Actional's contributions on the OASIS Management Protocol Technical Committee and its efforts to deliver XML-based Web Services management standards. Dan's experience with application integration technologies, including middleware, platform and Web Services, gives him a broad knowledge of the intricacies of systems such as SAP R/3, DCOM, CORBA and Java. He is the author of various application integration standards and has contributed significantly to the OMG standard for COM/CORBA *interworking*. Dan holds both a B.S. and an M.S. in Electrical Engineering from Cornell University.

■ ■ ■ dan.foody@actional.com

You Just Don't Get It

Bridging the schism between development and security priorities in XML development

■ There are many reasons why security professionals and application developers seem to work at cross-purposes when bringing XML Web Services to production. By understanding the key differences in approach and the way to bridge them, organizations can realize significant time-to-market and cost benefits without compromising security and compliance needs.

In early 2001, a spokesman from a large financial publisher, attended a Microsoft event demonstrating the power of .NET. With a flourish and a single line of code, the speaker showed how easy it was to transform a Visual Basic application and expose it as an XML Web Service. His application development colleagues were thrilled – fast, easy system integration appeared imminent.

“No!” he shouted, leaping to his feet. “You’ve just undermined all of our risk management processes and practices. Now any developer can inadvertently expose our business application logic to the Internet.”

Application developers looked at him as if an alien had landed in their conference. The spokesperson looked back wondering, “What are they thinking? Where are they going to get the security services they took for granted now that core application logic was itself exposed?”

Communication gaps between security and application development professionals are not new. Application developers are paid to create and evolve business applications that enable the enterprise to operate and to compete. Security professionals are paid to ensure that the company’s electronic assets are protected and safely accessed. Before Web Services and XML came on the



WRITTEN BY
ANDREW NASH

scene, the tension was managed in two ways:

1. By ensuring that application developers worked in an application environment that provided sufficient security
2. By ensuring that the network itself was secure, and only authorized parties could access specific applications.

XML’s integration and interoperability possibilities are a double-edged sword. While XML accelerates application integration and implementation, it cannot provide per-platform application environments with the services that most application developers have come to rely on. The application execution environment now covers multiple vendor platforms. Transporting XML and Web Services across high-level protocols such as HTTP also makes it difficult for traditional network devices to distinguish, route and filter messages that use sophisticated addressing schemes. As a result, application developers and security professionals find themselves at odds in spite of their common goal of improving enterprise performance.

Tortoise or Hare?

Today the goal of application developer is to create applications quickly, iterate them and release often, and XML services

are developed using the same methodology. Change is frequent, welcome and expected.

Security professionals, however, prefer that applications and services change as infrequently as possible since each iteration requires a new security review to ensure that no new risks have been inadvertently introduced. Control, consistency and correctness are paramount. As a result, application development teams are following their best practice in iterating and releasing often while the security team aims to slow the pace of releases and audit the changes. This difference in perspective casts the security team as a roadblock in achieving development milestones; or makes the development team a radical force that must constantly be policed.

Information Sharing or Information Compromise?

The development team’s goal is to create services that can be quickly and easily reused by other systems and by strategic enterprise partners – while reducing system-to-system integration costs. These are the primary reasons that organizations adopt XML Web Services architectures. Since the goal is rapid integration, developers want their services to share considerable information about how to connect and interact with them. Once developers create a service, other developers can discover the Web Services Description Language (WSDL) and use the service – quickly, automatically and transparently.

To a security team, you may as well be a hostile invader with a map to the house, a key to the front door and the combination to the safe. Security professionals must now work harder to establish trust between commerce and enterprise partners because application logic is exposed and network security is circumvented. The challenge lies in how to benefit from rapid integration without compromising an organization’s security and compliance posture.

Need for Security Specialists

When it comes to enterprise security, developers expect that security measures will be handled “somewhere else” in the



Are you Red Hat® ready?



Global Knowledge™
Experts Teaching Experts

Take our 10-minute assessment.

It's no secret that Red Hat Enterprise Linux is the most widely deployed Enterprise Linux solution in the world today.

In fact, Certification Magazine recently named the Red Hat Certified Engineer (RHCE) program the highest quality certification in all of IT. Global Knowledge, a premier partner for Red Hat training, invites you to test your skills and pre-assess for Linux success! Visit globalknowledge.com/lw for details.



infrastructure. Or, they attempt to program security into their services. However, application developers aren't versed in threat models or how to evaluate a solution's ability to withstand threats. Security professionals must ensure that applications are consistently secured across the enterprise and meet a wide range of compliance requirements. Application developers can't afford the time it takes to develop the expertise necessary for dealing with constantly evolving authentication, identity management, encryption, access control, authorization, security information management and event correlation technologies necessary for secure enterprise integration. As the financial publisher's spokesperson pointed out, XML services defy all the existing enterprise mechanisms designed to assure security efficiently.

Different Evaluation Criteria

Complicating the situation is the fact that security professionals and application developers are evaluated by different standards of success. Developers are evaluated and paid based on how fast they can bring feature-rich, competitively advantageous applications to market. Security professionals are evaluated on their ability to prevent, detect and resolve security issues and minimize overall enterprise risk. Stability and maintaining a strict scope of features are key evaluation metrics.

These differences exacerbate problems between the two groups, often leading to conflict, delay and additional risk – unless the enterprise can reconcile the organizational and structural needs of both groups. Many organizations find that they can enable both parties to achieve their objectives by establishing a common XML services infrastructure that independently manages and mediates opposing needs.

Aligning for Understanding

Successful organizations recognize the differences and acknowledge the different approaches while emphasizing the common goal. One approach often taken is creating cross-functional teams to design the service in the infrastructure requirements. If these teams are tasked and compensated

to collaborate and bring services to market, they can effectively address issues and make conscious trade-offs. Organizations can also provide these teams with decision guidelines, prompting them to address friction points explicitly, such as frequency of release; the amount of information contained in error messages; security approaches and time of integration; feature creep and secure development; and change management, approvals and workflow.

A second and complimentary approach is to deploy an infrastructure that addresses XML services' unique opportunities and benefits while preserving security, visibility and efficiency. An XML services infrastructure should provide the following functions to translate between application development, security and

- Enable hot updating of services, connections and policies as well as straightforward rollback

This spokesperson found that an XML services infrastructure significantly reduced the tension between application developers and security professional by enabling each to perform their functions in parallel, according to their respective goals and metrics. When one of his business units, took the lead in using XML Web Services to create a reliable reservation and payment system for corporate and educational testing, his principal application architect implemented a Reactivity XML Service Infrastructure to ensure privacy, security, interoperability and visibility for the resulting XML Services. Within the first year, the business unit reduced the cost of resolving SLA violations by more than the

“

Either the security team is a dangerous speed bump stopping the development team from hitting precious milestones or the development team is a radical force that must constantly be policed

”

operational needs:

- Provide the right amount of security for the service and use case;
- Enable developers to provision services and connections directly with central IT review;
- Enable developers to create and change relevant policy with central IT review;
- Enable different levels of error reporting and logging depending on the development phase;
- Enable security to visualize all changes and rapidly identify any new security risks to be resolved;
- Enable security professionals to approve requests both in part and in full;
- Enable seamless migration between development, testing and production environments;

cost of the new Reactivity infrastructure – while complying with applicable regulatory requirements. At this publishing company, everyone understands the goals and agrees on how to reach them. ©

About the Author

Andrew Nash is CTO of Reactivity. He is an acknowledged leader in the PKI and Web Services security markets and co-author of numerous Web Services specifications, including Web Services Security, WS-Trust, WS-Federation, WS-Secure Conversation and WS-Security Policy. Andrew is an author of an RSA Press book on Public Key Infrastructure, a member of the OASIS Web Services Security TC and was chairman of the PKI Forum Technical Working Group. He was graduated from the University of Adelaide in Australia, and holds a postgraduate degree in software engineering from the University of Technology in Sydney.

■ ■ ■ anash@reactivity.com

The **Smart** Way to Transform XML to Web Pages

1. Define XML
2. Drop XML on Web Page
3. XSL and CSS are generated



Done.

100% WYSIWYG



Transform XML => Web pages

Features

- Visual tool with Canvas
- Auto-generates XML tree based on sample XML data
- Drag-n-drop from XML tree directly onto the Canvas
- Generates XSL code & generates CSS code

Advanced Features

- Merge multiple XML sources
- Include live XML/RSS feeds in Web page
- Include other Web pages in Web page
- Use other Web languages with XSL in Web page
- Supply run-time input parameters to XSL Web page

XSLmaker

The only comprehensive XSL development platform for efficient making of professional Web pages and Web applications from XML data. Easy and Intuitive.



FREE trial. Download at www.xslmaker.com

BPEL to the Rescue: A Real-World Account

The backstory of a flexible and agile application

■ After spending much of last year learning to use the Business Process Execution Language (BPEL) to orchestrate Web Services and realize the benefits of a service-oriented architecture (SOA), we felt it was time for us to climb down off the bleeding edge of the razor and share our story about the real-world realities of implementing a BPEL-based project.

Policy Studies, Inc. (PSI) provides outsourcing, technology and consulting to government and private agencies. Outsourcing is a significant part of its business; and this service includes performing certain functions on behalf of clients such as registering newly hired employees with state human service agencies (as mandated by federal law) or processing eligibility and enrollment (E&E) for State Children's Health Insurance Programs (SCHIP). One of the challenges of outsourcing is to understand each state's unique business process requirements, then develop systems that are flexible enough to be used by multiple states, ones that can be easily customized to implement each state's unique policies and procedures.

The E&E line of business is relatively new for PSI. It's an area uniquely volatile in terms of requirements "churn." Federal and state legislation and policy—as well as program-level policy, procedures, funding and goals—all conspire to create an unusually large number of stakeholders with many needs, all of whom tend to keep the change orders flowing. In addition, we sometimes have to throw a case "over the wall" to another agency and wait to find out whether they will accept it or not. Currently, the integration with other agencies is done using batch files, and given the condition

WRITTEN BY

**ROBERT WALES,
KEVIN GEMINIUC,
PETER ZADROZNY
AND ARTHUR WANG**

of our partners' technology, it'll be this way for another year or two.

These issues convinced us that our new E&E system had to be based on a flexible architecture – one that allowed for a generic

system that we could quickly customize later; one that was easily adaptable in the field; and one that would fully support detailed but ever-changing operational workflows while letting us tune and/or overhaul them as our operations evolved.

In late 2003 we started reviewing our needs for a workflow solution that would integrate both people and systems in the basic

architecture we had already developed. We considered some of the existing commercial workflow products, but they didn't fit our needs. We were seriously considering writing our own simple workflow definition language and execution engine when BPEL entered the picture. It was perfect timing: here was a process definition language that modeled all the things we thought our own invented language would need to have, but one that was open, supported by a consortium of major vendors, and had a good chance of becoming a widely supported standard. But the various ways that BPEL let us integrate with partners – by enlisting them directly in our workflow, for example – was what sealed the deal. After evaluating the various BPEL product offerings on the market, we selected Collaxa BPEL Server, which has since been acquired by Oracle and renamed Oracle BPEL Process Manager.

State Children's Health Insurance Program

As we've said, PSI is using BPEL as part of a new application to support our SCHIP E&E outsourcing business line. We're developing the solution with a specific state as the target, but with a strong emphasis on making the system generic and customizable enough to take it to other states, and even to other kinds of operations.

SCHIP is a state program that offers state-subsidized health insurance coverage to individuals whose incomes are too high to be covered by Medicaid but too low to afford


“

We weighted existing commercial workflow products, but they didn't fit our needs. We were seriously considering writing our own simple workflow definition language and execution engine when BPEL entered the picture

”

Gartner Application Integration & Web Services Summit 2005

Register, with payment, by
March 18 and receive \$200 off
the standard registration fee.
(Mention Priority Code APN14WSJ
to receive this discount.)



The Integrated Service-Oriented Enterprise: Strategies for Tomorrow, Best Practices for Today

April 18–20, 2005 • Westin Century Plaza Hotel & Spa • Los Angeles, CA

**Gartner Application Integration and Web Services Summit
allows you to benchmark your enterprise strategies on:**

- Web services security, ROI and management
- Service-oriented architecture (SOA) and event-driven architecture (EDA)
- Enterprise service bus (ESB)
- Open source and standards
- Composite applications
- Application process and data integration
- Portals
- B2B integration **And MORE!**

**PLUS: The Enterprise Architecture Summit, April 20-21, held
in conjunction with the Gartner Application Integration and
Web Services Summit. Visit gartner.com/us/ea for details.**

Keynote and Guest Speakers



Tim O'Reilly
Founder & CEO
of O'Reilly Media,
Open Source and
Standards
Advocate



Geoffrey Moore
Industry Consultant
and Best Selling
Author of *Crossing
the Chasm*



David Luckham
Professor Emeritus
of Electrical
Engineering,
Stanford University



Jeanne Ross
Principal Research
Scientist, MIT

Gartner
Application Integration &
Web Services Summit

Register now at gartner.com/us/aiwswc

private health insurance. Because it targets people of certain income ranges, applicants must pass an eligibility test.

Here's an example of a high-level flow that occurs when a SCHIP application is processed:

1. The applicant submits a completed paper application form for SCHIP coverage.
2. The form is scanned and the image is sent to a data entry operator, who enters the data into the application.
3. The system checks for data completeness, making sure that all required fields have been filled out, and, in some cases, verifying that required documentation (such as pay stubs and proof of citizenship) has been submitted.
4. If the data is incomplete, a sub-workflow to make the application complete is launched.
5. If the data is complete, the eligibility determination process is called. This process is based on the ILOG JRules product that uses a ruleset customized to the state's SCHIP eligibility determination criteria.
6. The application is approved, denied or, in some cases, referred to another agency such as Medicaid.
7. If the application is approved, a new workflow is launched to complete the applicant's enrollment, matching the applicant with a specific Managed Care Organization and physician.
8. If the application is denied, a letter of denial is sent and the workflow ends.
9. If the application is referred, a new workflow is launched to refer the case to the corresponding agency.

Here's how we use Oracle BPEL Process Manager to implement some of these workflows.

Heterogeneous Environments and XML

One of the BPEL's strengths is its ability to integrate heterogeneous environments. In the example illustrated in Figure 1, processing a SCHIP account involves a BPEL flow that calculates eligibility for the beneficiary members of a family account. The process will perform three steps without human intervention:

1. Calculate eligibility for the account
2. Invoke a transaction-safe letter-generation sub-process
3. Queue letters to the family by looping through family members

The rule engine functionality is exposed to BPEL as a Web Service wrapper and the Session EJB is exposed using the Web Services Invocation Framework (WSIF). To the workflow designer, both the rules engine and the correspondence EJB appear as Web Services and are accessed using the BPEL "partnerlink" construct.

Since BPEL communicates with XML documents to its partner links, we defined all of the account and eligibility results data in a common XML Schema. This schema is imported into WSDL (Web Service Definition Language) files for both BPEL and our own

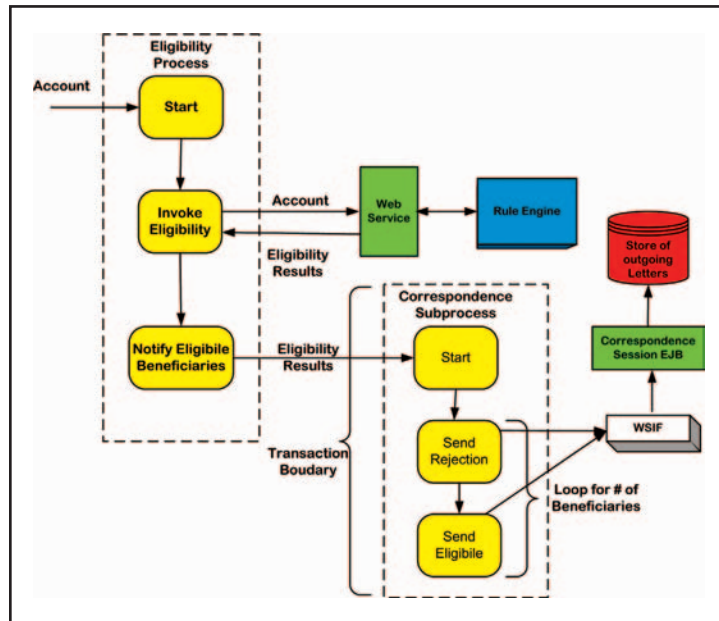


FIGURE 1 Integrating a heterogeneous environment

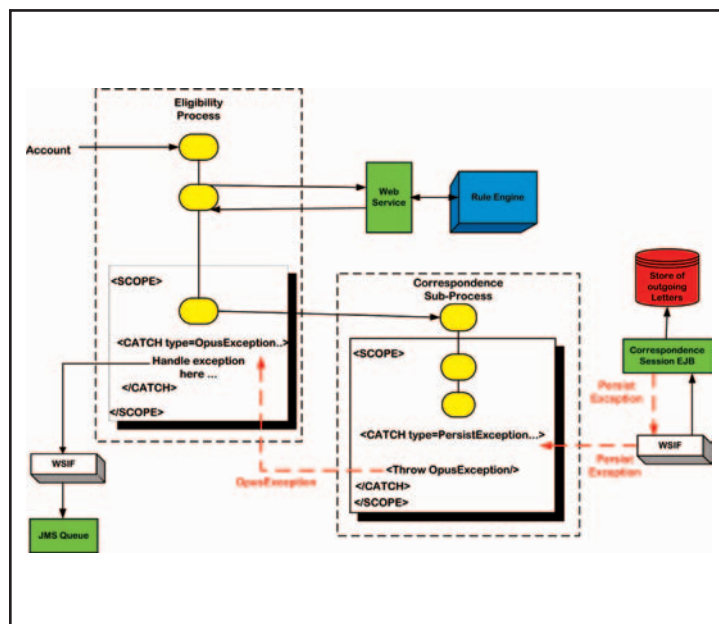


FIGURE 2 Transaction and exception handling using BPEL

custom Web Services. We also use XPath to access and manipulate data. Because each person can be individually eligible, the BPEL "while" construct is used to iterate through the Eligibility Result array and call the correspondence session bean interface.

Transactions and Exceptions

Additionally, the correspondence sub-process is defined as "supporting transactions." As a result, the session bean that participates in the transaction can be rolled back if there's a failure in persisting correspondence data, or any other exception.

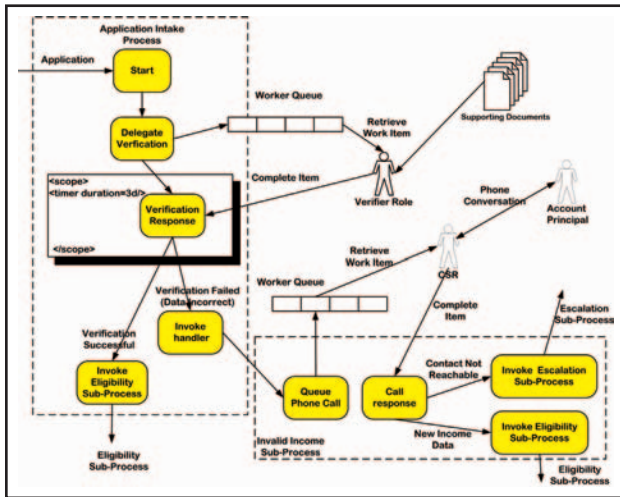


FIGURE 3 User and system task delegation

Exceptions have also been leveraged at multiple levels in our eligibility process. WSIF lets you map a Java exception to an XML-based exception, which can then be caught in our process with the BPEL “catch” construct. Also, the child sub-process can throw an exception back to the calling parent process. At the BPEL level you can catch specific exceptions or all exceptions with the “catchall” construct. BPEL “scopes” helped in our design process by providing more finely grained exception handling in sections of the workflow.

As presented in Figure 2, suppose a custom Java exception is thrown by the session bean. This notifies the caller that the correspondence request wasn’t queued. The WSIF layer then translates the Java exception into a XML fault, which is defined in the WSDL for the correspondence session bean. Once the exception has been caught in the correspondence process, the BPEL process creates a new *OpusException* and copies appropriate data into the new general exception. Finally, the *OpusException* is caught by the parent process, and is handled by queuing it on a JMS dead-letter queue. Having a custom exception facility lets us create our own exceptions and write a single handler for it in the parent process.

User Task Architecture

The SCHIP healthcare application workflow is a mixture of human- and system-oriented

tasks. We devised a delegation model between BPEL and our external user application to fulfill this requirement. A typical user task in the SCHIP application is the quality assurance step. When a potential recipient of benefits files an application, the information received is checked for validity against a paper record or some other verification source. The human interaction necessarily makes this response asynchronous. For example, when an applicant’s income is verified, the person verifying the

data will compare the income against paper records and/or online data to make sure the data reported is accurate. This process will not occur immediately. Also, the verifier role may involve special attributes and these must be managed during the delegation.

As seen in Figure 3, the application intake process has a quality assurance step that requires a human to verify income from a received application. This delegation to the user is considered asynchronous and has the extra constraint of duration. A scope that contains the *Verification Response* activity is defined for three days; if the user doesn’t respond during that time the BPEL engine will trigger an alarm event. The event handler for the alarm (not shown in Figure 3) typically re-queues the item to an administrator, who resolves the item by reassigning it. When the user completes a user task it is returned to the workflow with a specific resolution code. The workflow uses conditional branching and performs the appropriate activity so the application processing will continue along a defined path. This workflow illustrates the potential human action required to clear up data inconsistencies (the sub-flow beginning with “Queue Phone Call”); the optimal approach is to minimize the number of cases going to this branch.

Performance

Performance was an important factor in choosing Oracle BPEL Process Manager. Although we’re not yet in production, we feel

good about the capacity and scalability of the system we’re designing. Early on we undertook several stress tests to show how many BPEL workflows we could have in flight at any given time, and we’ve repeated these tests several times. Our tests show we can handle the tens of thousands of in-flight workflows a day that we anticipate encountering in production.

Conclusion

At this point, we’re creating the library of Web Services we need to complete the functionality required by the system. Orchestrating all of these Web Services with BPEL has given us a high degree of flexibility. In fact, our experience with BPEL has convinced us that it’s the right solution for the workflow needs of the SCHIP application, and we’re looking for other places to use it as well.

As we near production, we’re starting to look into day-to-day operational issues. Clear usage numbers provided by the Business Activity Monitoring (BAM) features in the Oracle BPEL Process Manager will help us assess potential costs for the manual workflow steps and re-engineer internal processes to create more inexpensive and automated workflows.

BPEL is getting a lot of attention, and rightly so. The flexibility it provides in orchestrating Web Services, coupled with its ability to implement real-time workflows, will give us a winning edge that will help us react quickly and efficiently to our ever-changing business requirements. ©

About the Authors

Robert Wales is vice president of enterprise architecture at Policy Studies Inc., where he has been a leading technical innovator for more than 15 years.

■ ■ ■ rwales@policy-studies.com

Kevin Geminiuc is a senior system architect at Policy Studies Inc. He has more than 15 years of experience building mission-critical enterprise solutions.

■ ■ ■ kgemiuc@policy-studies.com

Peter Zdrozny is vice president and chief evangelist for Oracle Application Server. In this role, he focuses on the use of open technologies for solving IT problems.

■ ■ ■ peter.zdrozny@oracle.com

Arthur Wang is a product manager for Oracle Application Server. He works closely with strategic customers implementing Oracle’s leading edge technologies.

■ ■ ■ arthur.wang@oracle.com

The Role of EII in SOA

The perfect marriage



■ EII and SOA are two of the newest acronyms bandied about in enterprise IT departments. Application architects are seeking to build loosely coupled applications with Service-Oriented Architecture (SOA). Data architects are trying to make information more widely available with Enterprise Information Integration (EII).

There's a common misconception that EII and SOA are competing integration approaches. But their roles are quite different and often complementary. SOA is suited to stitching together applications, while EII is used to integrate disparate data sources. In fact, EII can be leveraged to integrate information in an SOA application. This article will describe the role that EII can play in an SOA, resulting in a perfect marriage between the two – the “Virtual Data Service.”

Whence EII?

To understand the role of EII in an SOA, it's best to know how and why EII came about. Two primary forces were at work in setting the stage for EII – increased user expectations for instant data availability and cost-cutting pressures.

The average user's expectations have risen dramatically over the last 10 years. The Web



WRITTEN BY
**TIM
MATHEWS**

has spoiled all of us for good. Think about the financial information available on Yahoo Finance, or any of the dozens of other sites like it. Prices. Trends. Comparative analysis. For free, and in near real-time. Now, think of the information you want from your own organization. Perhaps you have a corporate portal, but does it compare to Yahoo?

The flipside of the coin is the cost-cutting pressures created by the slump in the economy. As users were expecting better and timelier information, it went against the grain of expense control to add databases and data marts, or build larger data warehouses. The cost of buying and operating these systems was simply prohibitive. And, likewise, adding more programmers to write custom apps or integrations wasn't possible either.

So the stage was set for EII. Was there a way to pull information from a variety of sources – inside the firewall and out – without the

expense of “big bang” integration or data replication projects, and without hiring an army of developers? What was needed was a way to query multiple sources without replicating data or writing lots of expensive custom code.

Write Queries, Not Code

EII uses queries to collect and integrate data and content from multiple sources. With EII, queries are distributed to data sources and the results are joined. EII is essentially a pull mechanism; a federated query goes out and finds the data needed by a user application and puts it into user view with context.

An EII platform hides the complexity of the various data sources being integrated and exposes a single data model, query language and programming interface to the end user. It's a good approach for many IT departments because:

- There are various data models, query languages and programming interfaces involved in the integration;
- There's a need for real-time access to operational information;
- The data architecture is often changing or in transition

The net result is a queryable platform that allows information to be gathered for use by the user or application. Since queries are

written in a standard way – using SQL or XQuery – it's much simpler to express than custom code. The declarative style of a query is also much easier and less expensive to maintain across large organizations than custom code. Further, since the EII platform manages the access to the data, the platform creates a virtual layer, removing the need for data replication. That's not to say that EII will replace data warehouses, but it may offer an alternative to data marts and is just the ticket for a range of applications, many of which don't require the level of data warehousing or data marts.

Application or Data Integration

Integration using Web Services most closely resembles the integration done using Enterprise Application Integration (EAI) platforms. Whether a bus or hub architecture, EAI products work by passing messages. Web Services pass messages to, albeit in a more standard form using XML over standardized protocols and transports. This use of a standard technology foundation and the malleable nature of the XML used in the payload have caused SOA to take off.

Comparing the Web Services/SOA approach with the EII approach clarifies the distinction between the two. If you wanted to pull data from an Oracle database, a SQL Server database and a data warehouse, which would be more natural, writing a query or passing a message? Naturally, a query is the way to go. Likewise, a message-based approach would be more natural for a purchasing application that touches multiple systems in a transaction. So, Web Services integration is more akin to application integration, while EII is really about data integration.

With that distinction in place, it's time to focus on two ways EII can be used in SOA.

Combining Web Services and Operational Data

There are many Web Services available today that provide data on demand. Weather. Stock quotes. Auction prices. The list goes on. These are rather simple targeted Web Services, but they provide programmatic access to important data.

While they are interesting and useful on their own, consuming them via an SOA is not much more valuable to end users than getting the same information through a Web browser. The real leverage comes when the data gathered from these Web Services are combined with other information. This gives users something they never had before – a unified view of the disparate data needed to do their job. In some cases this is faster than flipping between multiple web sites or application windows. In other cases, the data returned can be used as the input to another query, and then another, resulting in the combination of just the up-to-the-minute information needed.

A good example of this is a trading system for a commodities dealer. There are lots of similar examples in financial institutions, depending on the kind of products being sold, bought or traded, but the general idea is the fusion of market data and internal data in a market where seconds are worth a lot of money.

To illustrate this, we'll look at the job of banker trading heating oil – a market that's recently seen a lot of activity. This is a complex market, so we'll look at a simplified scenario. The broker needs three key pieces of information from the outside – the current

IN THE NEXT ISSUE OF WSJ...

FOCUS: Standards

Supporting the Business Process Lifecycle using Standards-based Tools

Business processes integrate systems, partners, and people to achieve key strategic and operations objectives. Examples of business processes include getting and filling orders, processing invoices, reconciling shipping notices and received goods and processing insurance claims and loan applications. The Holy Grail of enterprise computing is adaptive business processes that can be defined, refined, and optimized to respond to changing business environments, government regulations and competitive pressures.

Skyway Builder from Skyway Software

Services Oriented Architectures (SOA) rarely start from scratch. In most enterprises, they are built in gradual steps as part of an overall migration and architectural strategy. Along the way, existing legacy systems must be enhanced to support the required interface technologies, service gaps have to be filled by building new systems or acquiring and leveraging external services through third-party providers, and all elements in the architecture must be orchestrated and combined to form the final catalog of services. One solution that helps address these challenges is Skyway Builder from Skyway Software.

Understanding The SOAP Service Description Language

As we bring our SOAs on-line using Web services, we all know that SOAP is the standards message transfer protocol. Understanding that, the interface description language for Web services (WSDL) is not specifically for SOAP, but is more generic in nature. Thus, there is a requirement for a SOAP-centric contract description language for Web services...Enter SOAP Service Description Language or SSDL.

When Exceptions Are the Rule: Achieving Reliable and Traceable SOAs

Service-oriented architectures (SOAs) crystallize monolithic applications into collections of independent services. Because they are distributed and federated, services-oriented architectures are inherently vulnerable to exceptions. With the adoption of SOA, the number of exceptions is likely to increase exponentially.

WebServices JOURNAL
— .NET J2EE XML —

price of heating oil (the spot price), the futures price of the oil (actually a set of prices going out 18 months) and the weather forecast for the Northeast U.S. The weather plays a key role in the consumption of heating oil: the colder the winter, more oil used. We'll get the spot and futures oil prices from the New York Mercantile Exchange (NYMEX) via Xignite's public Web Service. The weather forecast will come from the National Weather Service's National Digital Forecast Database (NDFD) Web Service.

The internal operational data he needs are the potential buyers' current position in oil, i.e., their current holdings and outlook on the market, and the buyer's positions in other commodities (to prevent overexposure to a single trading partner). The position in oil comes from the trading database, in this

case a Sybase database. The other positions are contained in a clearing system database, an Oracle database in this example. Figure 1 below illustrates the data sources connected.

This application shows the power of EII because it's a collection of queries. The fact that these queries cut across multiple databases and external data sources exemplifies its effectiveness further. The EII platform takes the incoming query (the combination of product, price, weather, customer and position) and distributes the appropriate sub-queries to the appropriate data sources, using the appropriate query method (SQL in the case of relational databases and a SOAP call to the two Web Services).

The EII platform holds the knowledge of how its target data model maps to the underlying Web

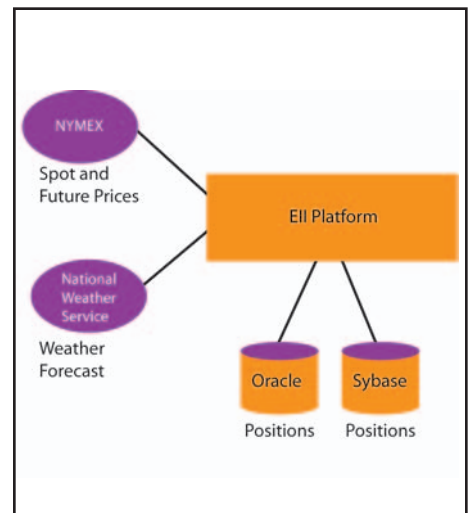


FIGURE 1 Combining Web Services and operational data in energy trading

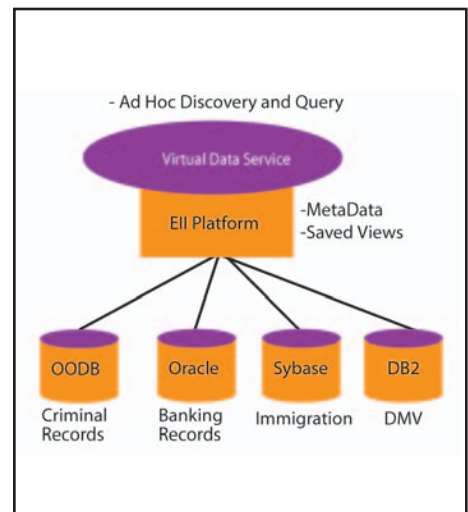


FIGURE 2 The Virtual Data Service in a government security application

Service APIs and the relational schemas and accordingly orchestrates invocation of multiple Web Service method calls and SQL queries to get to the underlying data. The platform then handles the translation and reconstitutes the results and presents them back to the trading application. XQuery is a good query language in this case, since it can not only express queries over a variety of data sources including relational databases and Web Services, but can transform the XML-based result set into the appropriate format for the UI or application. And because the EII platform uses queries that can be parameterized, it is very flexible and can be modified to



handle any number of combinations of products, customers, positions and data sources.

The Virtual Data Service

Most large organizations have many databases. Some have hundreds or even thousands of databases of all kinds – relational, object, hierarchical – that were put in place over the years. Often when companies merge or acquire, the databases of each party remain intact. The data needed for applications are scattered across multiple physical databases with multiple query languages. Migrating or replicating the data is simply unfeasible, and the cost of integration has traditionally exceeded the value of the combined information.

At the same time, IT architects are looking at ways to unify their application architecture using SOA. Having to deal with all of the individual databases individually would be quite a task, involving wrapping all the needed databases in Web Services then managing the query language, performance and security of each.

The solution is the Virtual Data Service. This is an EII layer that's available as a Web Service. So an application needing information from a combination of databases can call the Virtual Data Service using SOAP and it will take care of querying the back-end sources, translating and returning the results.

The federal government offers many good use cases for the Virtual Data Service, especially in defense and intelligence. The 9/11 attack on the World Trade Center and the Pentagon put a renewed focus on better intelligence in the U.S., much of which comes from analysts sifting through information in a range of databases. Much of this analysis is ad hoc, meaning that an analyst gets a new idea or theory and then needs to do a new set of queries. Multiply the number of ad hoc queries by the number of analysts and you end up with a big need for distributed on-demand query capabilities.

The other challenge federal agencies face is the sheer number of databases. The Department of Homeland Security (DHS), for example, reportedly has thousands of

databases, such as criminal records, driver license, and tax and banking records, across every municipality. They are of all types and geographically distributed. The sheer volume of data makes replication unfeasible. Besides, DHS is a virtual agency. Consolidating data across inter-agency political boundaries is near impossible.

An analyst would need to access several back-end databases to answer a particular query. The EII layer would be made available via Web Services, our Virtual Data Service. Figure 2 shows the data sources made available through the Virtual Data Service. As a first step, the analyst could query the Virtual Data Service for metadata on the available databases and any pre-existing queries, often called "views" in EII architectures. The analyst would send the query to the Virtual Data Service, which would then translate (unpack) the request into the appropriate SQL, OQL, XQuery or other query languages. The Virtual Data Service can also publish common or base queries so the analysts don't necessarily have to start from scratch with every new task.

The EII-SOA Marriage

EII and SOA are a perfect match. While SOA handles the integration and coupling of applications, EII manages the data and content integration needed. Whether an organization needs to integrate Web Services data with operational databases, or create a Virtual Data Service, which is a SOA-available virtual data source, they can gain significant leverage by combining the two.

Technically, the two have a symbiotic relationship. The

asynchronous nature of Web Services allows more time for query processing in the background during the built-in latency. The natural use of XML in Web Services makes including disparate data types – files, relational, documents – easy with one common XML model.

The future of SOA can bring much to bear on EII as well. We will see an SOA approach to services needed by the EII layer – such as reference data services or data quality services. Not unlike many relationships that begin with misconceptions, this one should make for a long, lasting marriage. ☺

About the Author

Tim Matthews is co-founder of Ipedo, Inc., a leading provider of Enterprise Information Integration (EII) software. He has written extensively on data management and integration in publications such as XML Journal, DevX and Intelligent Enterprise, and is a frequent speaker on EII.

■ ■ ■ tim@ipedo.com

WSJ Advertiser Index

Advertiser	URL	Phone	Page
EV1Servers	www.ev1servers.net	800-504-SURF	3
Forum Systems	www.forumsys.com	801-313-4400	6
Gartner Conference	www.gartner.com/us/aiwswc	203-316-6757	19
Gilbane Conference	www.lighthouseseminars.com	781-821-6734	36
Global Knowledge	www.globalknowledge.com/lw	919-461-8600	15
HostMySite.com	www.hostmysite.com/mxdj	877-248-4678	Cover IV
IT Solutions Guide	www.sys-con.com	888-303-5252	53
Itemfield	www.itemfield.com/itsolutions	800-838-1055	33
Mindreef	www.mindreef.com/tryout	603-465-2204	9 & 11
NetWorld + Interop	www.interop.com	415-905-2300	27
Parasoft	www.parasoft.com/soaptest_WSJ	888-305-0041	Cover II
Promind Systems	www.xmlmaker.com	415-359 9117	17
Secrets of the XML Masters	www.sys-con.com/freecd	888-303-5282	51
SYS-CON Reprint	www.sys-con.com	201-802-3026	35
Web Services Journal	www.wsj2.com	888-303-5252	43
WebAppCabaret	www.webappcabaret.com/dj.jsp	866-256-7973	5

Advertiser is fully responsible for all financial liability and terms of the contract executed by their agents or agencies who are acting on behalf of the advertiser. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

Web Service Management (WSM) – Architecture Patterns

Balancing management against complexity

■ Web Services are rapidly emerging as important building blocks for business integration. They are finding important applications in business-to-business, business-to-consumer and enterprise application integration solutions. As such, Web Services are critical to business architecture, and their reliable execution must be assured. Reliability must be a first-rank consideration for organizations deploying such solutions. The management of computer systems and applications is a discipline that is well developed and extensive. It has recently evolved to encompass Web Services.

The Web Services market provides many choices for developers requiring a service management system. While the features of these systems vary, each requires developers to create manageable services. In this article we will show how Web Services developers can incorporate manageability in their applications. We will first briefly survey the parts of instrumentation that are relevant to our discussion. Then we will detail service management principles and patterns.

Instrumentation Requirements:

Management systems usually collect



WRITTEN BY
RAMY ABAAS

information from Web Services. To collect this information, the management system uses an agent-manager model. The agent-manager model might deserve its own article, but for our purposes, here's the agent-manager model of the WSM in a nutshell: A manager is an application with the Web Service Management that communicates with management agents via management protocols. An agent is an application installed in the managed resources. A management protocol is the two-way communication channel between the manager and the management agent."

The table on following page illustrates the

information categories and the details.

Web Services Management Principles

Web Services-based applications will have characteristics over the traditional application that make management a little more challenging as follows:

- Web Services are described with XML and are accessible using standard interoperable protocols and transports. Therefore, applications based on Web Services will be able to use services that execute in many diverse environments – systems, languages, platforms and enterprises.
- Web Services-based applications will cross enterprise boundaries more now than applications used to. As a result, it must be possible to interact with the management aspects of these applications across enterprise boundaries, preferably using the same communications pipes and technologies already agreed on among the companies.
- Web Services define a standard mechanism for publishing, finding and interacting with their Web Services. Management systems have traditionally defined such a mechanism, both standards-based and proprietary, for publishing, finding and interacting with manageable resources. Management systems that support Web Services should interact with the Web Services publication

NETWORLDSM + INTEROP[®]

LAS VEGAS • MAY 1-6, 2005

Network Infrastructure and Services
Wireless
Security
Performance
VoIP and Collaboration
Data Management and Compliance

See it **All in One Place**
ALL SYSTEMS

**350+ Top Exhibitors on
the Exhibit Floor with
8 Targeted Technology
Zones and Pavilions**

**100+ Educational Sessions,
Including 6 Comprehensive
Conferences Revolving Around 6 Key
Themes, 3 Special Interest Days
and 36 Tutorials and Workshops**

**6 Visionary Keynotes
by Leading Industry
Executives**



**NetworkWorld
Survivor Las Vegas**



GO

Visionary Keynotes



John Chambers
*President and Chief Executive Officer,
Cisco Systems*



Hossein Eslambolchi
*President—AT&T Global Networking Technology
Services, Chief Technology Officer and Chief
Information Officer, AT&T*



Scott Kriens
*Chairman and Chief Executive Officer,
Juniper Networks*



Sean Maloney
*Executive Vice President
General Manager, Mobility Group,
Intel*



Andy Mattes
*President and Chief Executive Officer,
Siemens Communication Networks*

Your Source for Building a Better IT Infrastructure



Copyright © 2005 MediaLive International, Inc., 795 Folsom Street, 6th Floor,
San Francisco, CA 94107. All Rights Reserved. MediaLive International, NetWorld,
Interop and associated design marks and logos are trademarks or service marks
owned or used under license by MediaLive International, Inc., and may be
registered in the United States and other countries. Other names mentioned
may be trademarks or service marks of their respective owners.

Register Today at www.interop.com

Use priority code **MLAHNV40 and receive
\$100 off any educational product.**

and discovery practice.

- Web Services have a natural loose coupling that means a service should be able to “find” its management services – using the Web Services paradigm – at runtime rather than having them statically bound or internalized during development. As a result, the Web Services, as well as the management services, are more portable to different execution environments.

All of these characteristics drive the need to develop a management approach that stays within the Web Services paradigm. It means defining the management interactions with WSDL, the management applications as Web Services, the manageable services with WSDL and discovery using service registries and WSDL.

Thus, when considering the nature and usage models of Web Services and the challenges of managing them, several specific management principles emerge. We will summarize them here, and then treat each in detail with some illustrative example.

- Separate management interface principle means to define the business interface separately from the administration or management interface.
- Use of architecture pattern principle means architectural patterns that define components that make up the solution and the relationship between the Web Services and the management system.

Separate Management Interface

A Web Service is fundamentally an interface accessible over a set of open standard discovery and invocation mechanisms. Web Services discovery and binding processes are driven by interface descriptions. A Web Service interface is described as a port type in a WSDL file. When an organization searches a UDDI registry for a Web Service, the target of the search is an interface described in WSDL. Management operations should be exposed through a separately described and published Web Service interface that separates the business interactions concern from the management interactions with the service.

There are several reasons for this, as we now describe.

- The search for a Web Service usually in-

Category	Details
Identifications	Identification includes the static information that uniquely describes a manageable service. This information can include its instance name, the product name, product version, installation date, descriptive text, configuration file names, port and uniform resource locator (URL).
Availability	The availability of a service refers to its being accessible through the network, system and application infrastructure. Availability also indicates whether the service can do its work in a valid, responsive way
Operation	Operations can be management-oriented (start or make available, stop or make unavailable, obtain statistics), or they can be very service-specific and part of the business context. Operations may or may not change the current behavior of the application. But if they do, the change is rarely persistent over subsequent invocations of the application. An operation can turn certain trace points “on” during execution, but it will be reset to “off” when the application is started again.
Events	Events are messages from the Web Service to a management system. They can flag a failure condition or warn of an impending application outage. They can also signal a lifecycle, status, configuration or metric change. Reactions include running a recovery or tuning policy, notifying operators, logging or filtering. Events can also indicate that the application is healthy; these are usually called “heartbeats.” In this case, when the events aren’t received when expected, the management system should react.
Metrics	Metrics are usually numeric information provided by a manageable service that can be used to indicate or calculate the health and performance of the service. Metrics are often polled and, once collected, are graphed and subjected to threshold analysis
Configuration	Configuration information includes parameters that control how an application operates. A configuration can be static (not changeable while the service is available) or dynamic (the configuration can change while the service is available without service disruption). Configuration changes affect the current behavior of a service, and the changes are persistent over invocations of the service. A configuration may exist for how the Web Service interacts with its clients, or for its business logic.

volves looking for a standard or mutually agreed-to interface. Mixing management operations with those that are formally part of the business interface will interfere with the search for the service based on interface contents by changing the signature of the interface.

- It will allow business partners to see and use management interfaces that are irrelevant to their interactions with the Web Service. Likewise, it will clutter management consoles

with business operations when it should only display management operations.

- Providing a separate service and port in the WSDL document for the management interface enables more targeted publishing of business and management service ports in the WSDL. The service in the WSDL document for the management interface of a Web Service doesn’t have to be published in a public UDDI registry along with the business interface described in the business service

port, since only management applications will be interested in them. If it is published in a public UDDI registry, the service in the WSDL must be categorized as “management” so the management system can find it. The management interface will most likely be published in a private UDDI registry that caters to management systems as a client rather than business systems as a client.

A manageable service can support a generic management interface that provides simple access to identification, configuration and metric data. Ideally, this port would be widely supported by management applications that support Web Service management.

An example of a generic, overly simplified management interface that could be implemented by any manageable Web service is shown below.

```
public interface ManageableService
{
    public String getServiceID();
    // return an ID string for
    // the managed service
    public String[] [] getMetrics();
    // return an array of metric
    // names and a corresponding array
    // of values
    public String[] []
        getConfiguration();
    // return an array of config
    // property names and a
    // corresponding array of values

    public String getAdminInterface();
    // return the WSDL document
    // that describes this interface
    public Boolean isAvailable();
    // return true if service is
    // responding, false if the service
    // is experiencing delays
}
```

There might also be a custom manageable service WSDL interface that contains the interactions for the specific administrative or management operations of a Web Service. An example of operations in this kind of WSDL would be “TraceOn()”.

The following WSDL illustrates a generic management interface.

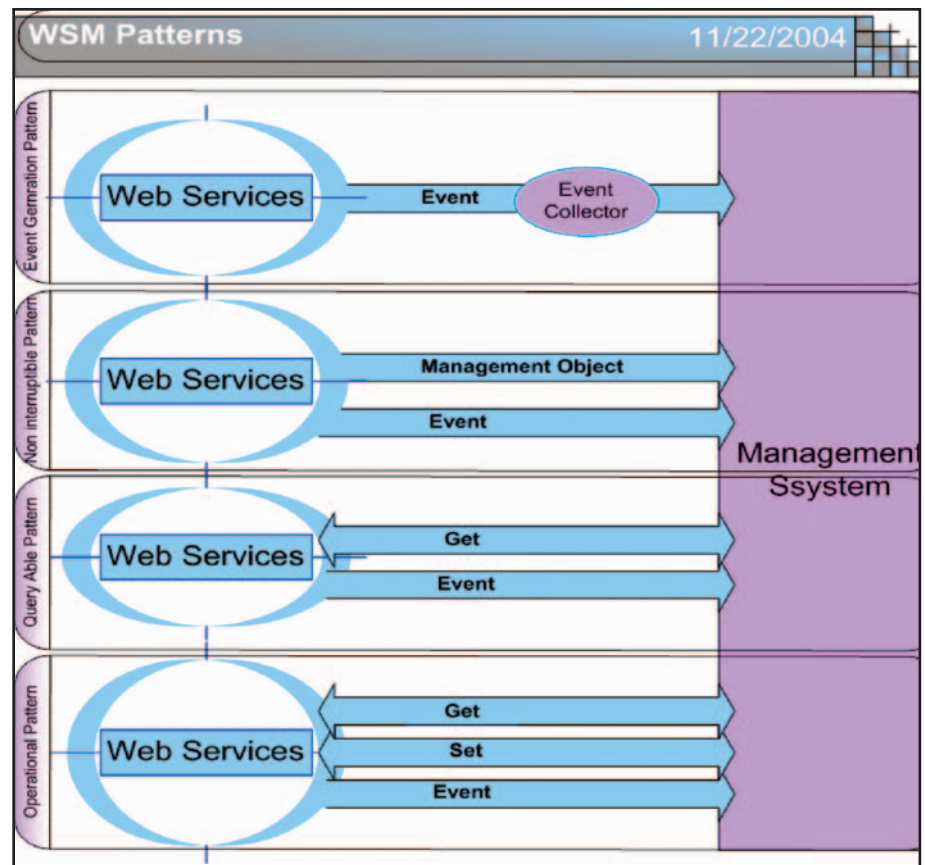


FIGURE 1 Web Services Management Patterns

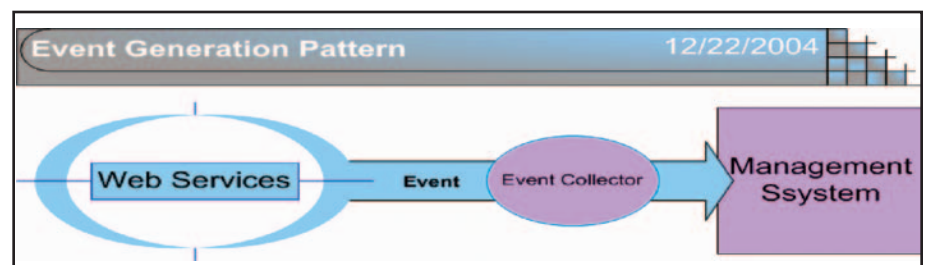


FIGURE 2 Event Generation Pattern (EGP)

```
<wsdl:message name_='
    GetServiceIdOutput'>
  <part name_='value' type_='
    xsd:float' />
</wsdl:message>
<wsdl:message name_='GetMetricsOut
  put'>
  <part name_='name' type_='xsd:
    string' />
  <part name_='type' type_='xsd:
    string' />
  <part name_='value' type_='xsd:
    string' />
```

```
</wsdl:message>
<wsdl:message name_='GetConfiguration
  Output'>
  <part name_='name' type_='xsd:
    string' />
  <part name_='type' type_='
    'xsd:string' />
  <part name_='value' type_='
    'xsd:string' />
</wsdl:message>
<wsdl:message name_='GetAdmin
  InterfaceOutput'>
  <part name_='AdminInterface
```

```

        WSDLurn'' type_''xsd:string'' />
</wsdl:message>
<wsdl:message name_''GetAvailability
    Output''>
    <part name_''value'' type_
        ''xsd:integer'' />
</wsdl:message>

<wsdl:portType name_''Generic
    ManagementInterface''>
    <wsdl:operation name_''
        GetServiceId''>
        <wsdl:output
            message_''tns:GetServiceId
            Output'' />
        </wsdl:operation>
        <wsdl:operation name_''
            GetMetrics''>
            <wsdl:output message_''
                tns:GetMetricsOutput'' />
            </wsdl:operation>
            <wsdl:operation name_''
                GetConfiguration''>
                <wsdl:output message_''
                    tns:GetConfigurationOutput'' />
                </wsdl:operation>
                <wsdl:operation name_''
                    GetAdminInterface''>
                    <wsdl:output message_''tns:
                        GetAdminInterfaceOutput'' />
                    </wsdl:operation>
                    <wsdl:operation name_
                        ''IsAvailable''>
                        <wsdl:output message_''tns:
                            IsAvailableOutput'' />
                        </wsdl:operation>
</wsdl:portType>

```

This WSDL illustrates the custom management interface:

```

<wsdl:message name_''Change
    AvailabilityInput''>
    <part name_''action''
        type_''xsd:string'' />
</wsdl:message>
<wsdl:message name_''Change
    AvailabilityOutput''>
    <part name_''newState''
        type_''xsd:string'' />
</wsdl:message>
<wsdl:message name_''setTraceInput''>
    <part name_''action''

```

```

        type_''xsd:string'' />
</wsdl:message>
<wsdl:message name_''changeStock
    Exchange''>
    <part name_''
        exchangeName'' type_''xsd:
            string'' />
</wsdl:message>

<wsdl:portType name_''StockQuote
    ManagementInterface''>
    <wsdl:operation name_
        ''setTrace''>
        <wsdl:input
            message_''tns:setTraceInput'' />
        </wsdl:operation>
        <wsdl:operation name_''
            changeStockExchange''>
            <wsdl:input message_''tns:
                changeStockExchangeInput'' />
            </wsdl:operation>
            <wsdl:operation name_''
                changeAvailability''>
                <wsdl:input message_''tns:
                    ChangeAvailabilityInput'' />
                <wsdl:output message_
                    ''tns:ChangeAvailabilityOutput''
                    />
                </wsdl:operation>
</wsdl:portType>

```

Architectural Patterns

There is a set of patterns that provides sufficient guidance to simplify the development of manageable Web Services. Developers can choose the pattern that applies to their Web Services behavior and requirements. These management patterns are architectural patterns that define components that make up the solution and the relationship between the Web Services and the management system. They show the placement of function and the flow of information.

There are four patterns and the diagram in Figure 1 illustrates the different WSM architecture patterns. These are the Architecture patterns with their definitions, when and how to use them:

1. Event Generator Pattern (EGP)

This pattern offers a very simple approach to instrumentation. By using an event collector, the Web Service can be completely isolated

from the details of the management system under which it operates.

A convenient approach to implementing this interaction is to create an event collector Web Service that exposes a simple interface for receiving events. The Web Services call on the event collector interface causes the events to be forwarded to the management systems that have registered for them.

The event collector interface can be quite general and simple. Minimally, it must include a method that allows it to receive the management event. For example:

```

Public interface EventCollector {
    Public void deliverEvent (string id,
        String source, String severity,
        String text);
}

```

This interface should identify the Web Service, specify the cause of the event, indicate the severity of the condition and provide additional information in the form of unformatted text. When the Web Service initializes, it can dynamically discover a Web Service that implements this interface as described by a published WSDL document. Alternatively, the Web Services developer can statically bind the Web Service to the event collector. The event types are application-defined and are not constrained by the event collector. If an event collector supported delivering events to interested management services via Web Services, it could also provide a WSDL-described port that other Web Services could use to ask the event collector to invoke their “receive event” operation based on certain conditions or contents of event.

When do you use this pattern?

This pattern is applicable to Web Services whose:

- Processing contains events and metrics of interest that are not already collected by the Web Services execution environment.
- The service doesn't require runtime operations or dynamic configuration control.

How does it work?

The management system is responsible for putting the information in context since the information in the event may be minimal. In this pattern:

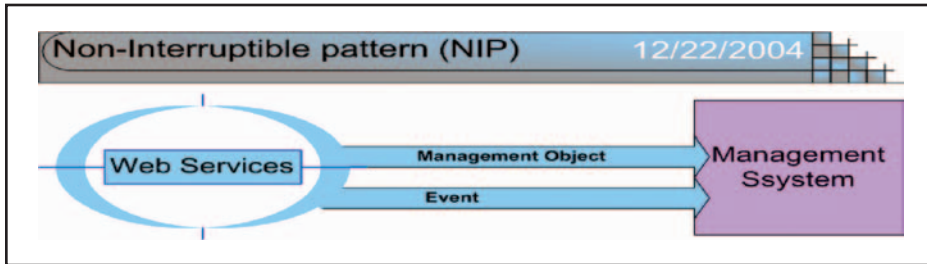


FIGURE 3 Non-Interruptible Pattern (NIP)

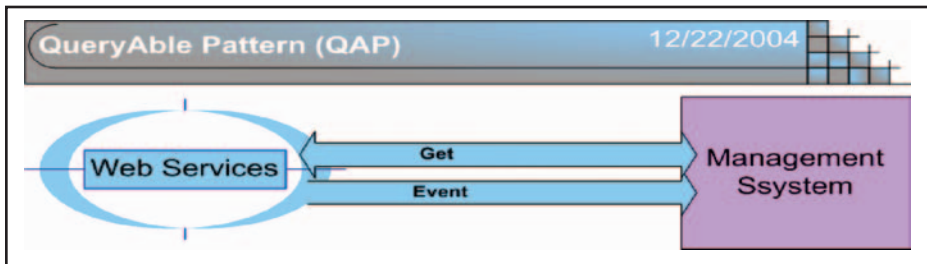


FIGURE 4 QueryAble Pattern (QAP)

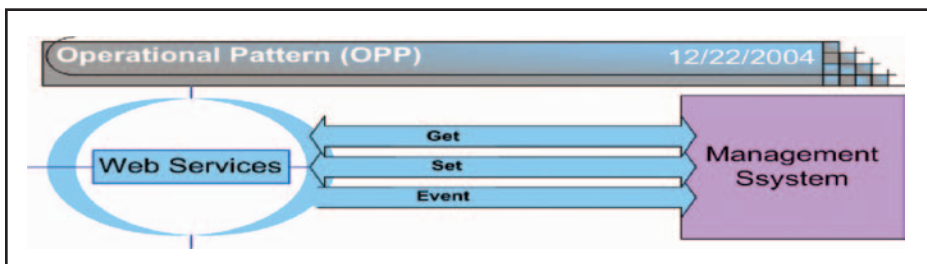


FIGURE 5 Operational Pattern (OPP)

- The Web Service doesn't publish a management object.
- The Web Service doesn't support being invoked for operations or reconfiguration by the management system.
- The Web Service neither implements facilities to listen for or respond to management requests nor implements the generic manageable service port.
- The Web Service doesn't have a custom management port.

The WSDL interface for this pattern is a set of notification messages representing the events.

The WSDL for the event collector service would contain the messages and ports given in the following Figure:

```
<wsdl:message name="Event">
  <part name="id" type="xsd:string" />
</wsdl:message>
```

```
<part name="source"
type="xsd:string" />
<part name="severity"
type="xsd:string" />
<part name="text" type="xsd:
string" />
</wsdl:message>
<wsdl:message name="requestEventInput">
  <part name="idpattern"
type="xsd:string" />
  <part name="sourcepattern"
type="xsd:string" />
  <part name="severityrange"
type="xsd:string" />
  <part name="textpattern"
type="xsd:string" />
  <part name="receiverWSDLPort"
type="xsd:string"/>
</wsdl:message>
<wsdl:portType name="EventCol
lectorInterface">
```

```
<wsdl:operation name="
deliverEvent">
  <wsdl:input message="tns:
Event" />
</wsdl:operation>
</wsdl:portType>
<wsdl:portType name="EventRequest
orInterface">
  <wsdl:operation
name="requestEvents">
    <wsdl:input message="tns:
RequestEventsInput" />
    <wsdl:output
message="tns:RequestEventsOutput" />
  </wsdl:operation>
  <wsdl:operation
name="deliverEvent">
    <wsdl:output
message="tns:Event">
  </wsdl:operation>
</wsdl:portType>
```

2. Non-Interruptible pattern (NIP)

The Web Service contains instrumentation and sends events and publishes management information to a management presence. Unlike the Event Generation Pattern, the Web Service makes a standard object available to the management system that contains the identity and metric information as well as details of the current configuration. Notifications of changes to the management object can be sent via events specified for that purpose or by resending the updated management object to the management system.

The management object could be any of several types. For Web Services being monitored by a JMX management environment, the object could be an MBean. Some management systems also support the Common Information Model (CIM) so the web service could alternatively create a CIM object to fill this role. It could also create a custom object, but such an approach may require some kind of bridge code to make the object understandable to the management system. MBeans or CIM objects are more workable choices

When do you use this pattern?

This pattern is useful over the event generator pattern when:

- There are great amounts of potentially

complex management data that may be frequently updated.

- When the management is not real-time sending the entire management object periodically rather than every time a value is updated tends to be more efficient
- It also allows a set of changes to be sent so they are consistent with one another rather than sending a set of events, some of which may not arrive in order or at all, leaving the management system's version of the management object in an invalid or inconsistent state.
- A particular management system requires the data in a particular format. The receiver of this information gets data already in context of the management object it understands
- Web Service doesn't support being invoked for operations or reconfiguration by the management system

How does it work?

The Web Service contains instrumentation, and sends management data via events to a management system and notification messages for updating the management object. In this pattern:

- The Web Service doesn't have to implement any facilities to listen for and respond to management requests or implement the generic manageable service port
- The Web Service doesn't have a custom management port.

The WSDL interface for this pattern contains notification messages for events and a notification message for updating the management object.

3. Queryable Pattern (QAP)

The Queryable Pattern is related to the Event Generation Pattern in that the Web Service doesn't require configuration or operational control but, in this case, the service may send events and implement a management interface that can be called by the management system. The management system can retrieve configuration and metric data directly from the managed services. This is essentially read-only management. The managed Web Service supports being invoked by the management system to obtain current metric and configuration data,

but not being controlled or altered.

When do you use this pattern?

This pattern is applicable to Web Services with a high rate of change in metric values or large numbers of complex metrics so that sending events to signal changes represents too much overhead. [Simplify your sentences where possible.]

How does it work?

The Web Service can support a management object and supports being invoked for query operations by the management system. The management data may be represented as complex data types and objects. Management systems would typically retrieve the management data when they need it or poll for entire sets of related management data rather than one or two metrics. In this pattern:

- The Web Service implements facilities to listen for and respond to management requests
- The Web Service may implement the generic manageable service port
- The Web Service may have a custom management port

The WSDL interface for this pattern contains a set of notification messages for events and request-response messages for data requests.

4. Operational Pattern (OPP)

In this pattern, the management system can retrieve configuration and metric data and obtain events from managed service and can set configuration data thus changing the application's operation. It can also invoke operations on the managed Web Service.

When do you use this pattern?

This pattern is applicable to Web Services with reconfiguration requirements and can be done through the management systems.

How does it work?

The Web Service must implement methods that can be called by the management system and must have a structure designed to allow for configuration changes. In this pattern:

- The Web Service can support a manage-

ment object and supports being invoked for operations or reconfiguration by the management system.

- The Web Service can implement facilities to listen for and respond to management requests
- The Web Service can implement the generic manageable service port.
- The Web Service can have a custom management port.

This pattern employs a WSDL interface to express a set of queries, operations and notifications.

Summary

Web Service developers need to balance management requirements and complexity implications. They would be advised to separate the management interface from the business interface and push the core metric collection down to the Web Services management system. The Patterns discussed here give increasing amounts of information and control over the Web Service to the management system. Any of the patterns may send events directly to the management system or through an event collector.

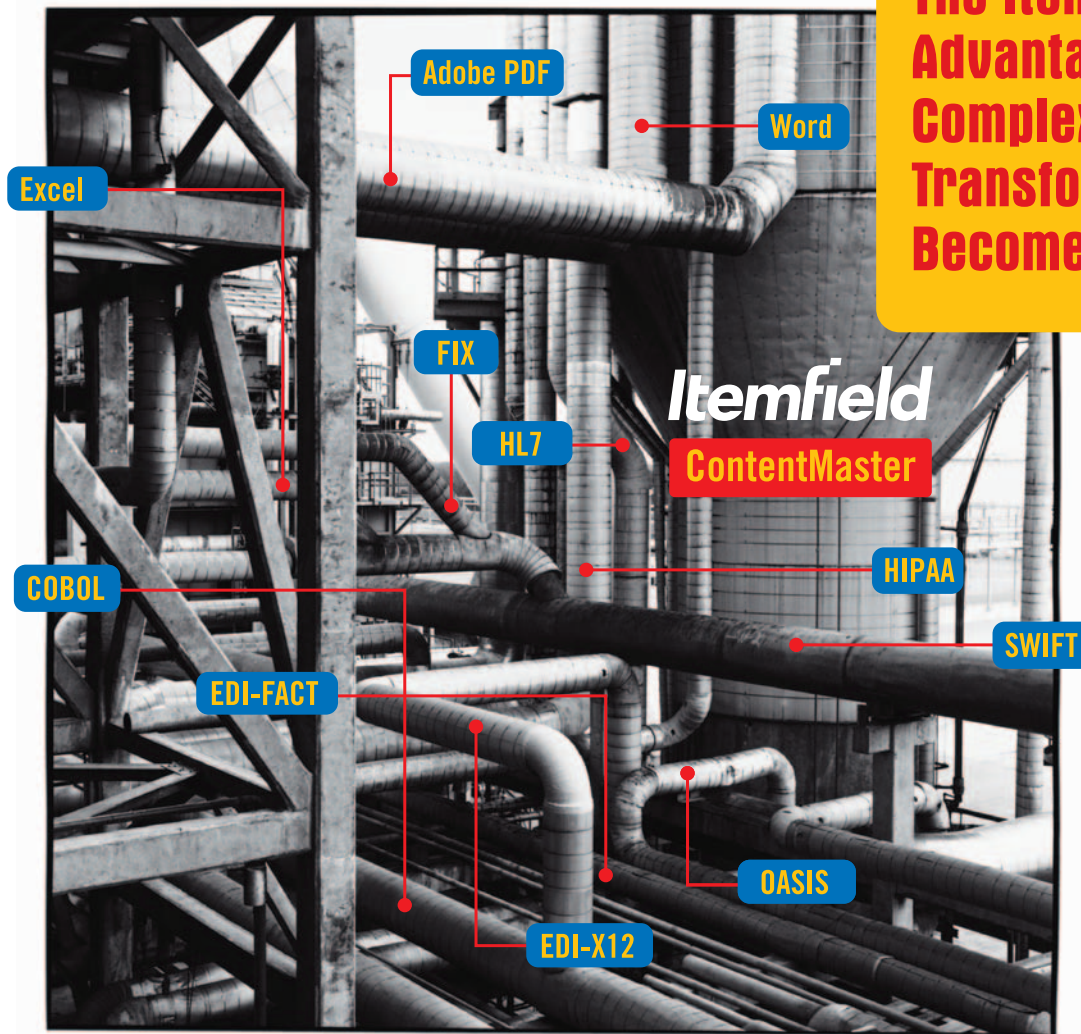
The Queryable Pattern and Operational Pattern will usually use a management object as well. These two patterns must support two-way communications. They must both implement a way for the management system to initiate communications with the Web Service. Only the operational pattern allows the management system to control the Web Service. Both Event Generation Pattern and Non-Interruptible Pattern use send-only communications and don't implement any way for the management system to initiate communications with the Web Service. Events are generally more generic and easily translated to be understood by any management application; therefore, the Event Generation Pattern is more portable than the others.©

About the Author

Ramy Abas is the Director for Covansys. He holds an M.S. in CIS and multiple Microsoft and IBM Certifications. He has over 16 years of experience designing and developing IT systems for Big Five clients and has served as an IT architect for IBM.

■ ■ ■ rabaas@covansys.com

**The Itemfield
Advantage Is That
Complex Data
Transformation
Becomes Simple**



Achieve the “Complexity Advantage” in Business Integration

Find Out What a Growing Number of Global Fortune 500 Companies Already Know

Complex data is at best a significant cost factor in business integration—at worst it can cripple a business integration initiative. For many enterprises, integrating complex data remains an error prone and costly process that requires IT development.

ContentMaster™ is next-generation data transformation software that dramatically simplifies application development and integration—reducing implementation costs by as much as 80%. ContentMaster allows any form of enterprise data to be transformed to any other form in real-time via XML—without the need for costly IT development.

ContentMaster includes a powerful visual integrated design environment and embeddable runtime engine that is available standalone for deployment into virtually any software architecture, or as integration agents for deployment to IBM® WebSphere® Business Integration, Microsoft® BizTalk and webMethods®.

Visit our website www.itemfield.com/itsolutions to obtain a copy of our latest white paper “*The Complexity Advantage: Solving the Requirements of Complex Data Transformation in Business Integration*” and to view the ContentMaster demo.

Itemfield

1 800 838 1055

www.itemfield.com/itsolutions

Understanding Coupling in the Context of an SOA

Comparative Architecture

■ Since the beginning of computing we've been dealing with the notion of coupling, or the degree to which one component is dependent on another component in both the domain of an application or an architecture. Lately, the movement has been towards loose coupling for some very good reasons, but I'm not sure that many architects out there building SOAs understand the motives behind this.

Breaking this concept down to its essence, we can state that tightly coupled systems/architectures are dependent on each other, thus changes to any component may prompt changes to many other components. Loosely coupled systems/architectures, in contrast, leverage independent components, and so can operate independently. However, it's not as simple as all that.

Keep in mind that how loosely or tightly coupled your architecture is is a matter of requirements, and not as much about what's popular. Indeed, architects need to understand the value of SOAs and loose coupling, and make the right calls to insure that the architecture matches the business objectives. So it's helpful to walk through this notion of coupling as you approach your SOA.

Hang Loose!

With the advent of Web Services and SOA we've been seeking to create architectures and systems that are more loosely coupled. Loosely coupled systems provide many advantages including support for late or dynamically



WRITTEN BY
**DAVID
LINTHICUM**

binding to other components while running, and can mediate the difference in the component's structure, security model, protocols and semantics, thereby abstracting volatility.

This contrasts to compile-time or run-time binding that requires that you bind the components at compile-time or run-time (synchronous calls) respectively, and also requires that changes be designed into all compo-

nents at the same time due to their dependencies. As you might imagine, this kind of coupling makes testing and component changes much more difficult.

The advantages of loosely coupled architectures, as found in many SOAs, are apparent to many of us who have built architectures and systems in the past, at least from a technical perspective. However, they have business value as well.

First and foremost, a loosely coupled architecture allows you to replace components, or change components, without having to make reflective changes to other components in the architecture/systems. This means businesses can change their business systems as needed, with much more agility than if the architec-

ture/systems was more tightly coupled.

Second, developers can pick and choose the right enabling technology for the job without concerning themselves with technical dependencies, such as security models. Thus, you can build new components using J2EE, which will work and play well with other components written in Cobol or perhaps C++. Same thing goes for persistence layers, middleware and protocols. You can mix and match to meet your needs, even leverage services that may exist outside of your organization without regard for how that service was created, how it communicates, or where it's running.

Finally, with this degree of independence components are protected from each other and can better recover from component failure. If the SOA is designed correctly, the failure of a single component shouldn't take down other components in the system. Thus, loose coupling creates architectures that are more resilient. Moreover, this also lends itself better to creating a failover subsystem and moving from one instance of a component to another without affecting the other components in the SOA.

It should be noted, however, that not all tight coupling is bad. Indeed, in some cases it makes sense to more tightly couple components, such as when dependencies are critical to the design. An example would be two services that can't work apart, and must function as one, and so are better tightly coupled. You have to look at your requirements, and then determine the degree of coupling needed in your architecture, and it may not always be loose coupling.

Testing for Loosely Coupled Architecture

So, now that we know the basic differences between a tightly and loosely coupled architecture, as well as the advantages, perhaps it's a good idea to break down loose coupling in to a few basic patterns: location independence, communication independence, security independence and instance independence.

- Location independence refers to the notion that it doesn't matter where a service exists. The other components that need to leverage the service can discovery it in a directory, and leverage it through the late-binding process. This comes in handy when you're

leveraging services that are consistently changing physical and logical locations, especially services outside your organization that you may not own. Your risk calculation service may exist in L.A. on Monday and in New York on Friday, and it should make no difference to you.

Dynamic discovery is key here, meaning that calling components can locate service information as needed, and without having to bind tightly to the service. Typically, these services are private, shared or public services as they exist in the directory.

- Communications independence means that all components can talk to each other no matter how they communicate at the interface or protocol levels. Thus, we leverage enabling standards, such as Web Services, to mediate the protocol and interface difference.
- Security independence refers to the concept of mediating the difference between secu-

rity models in and between components. This is a bit difficult to pull off, but necessary to any SOA. To enable this pattern, you'll have to leverage a federated security system that's able to create trust between components, no matter what security model is local to the components. This has been the primary force behind a number of federated security standards that have emerged in support of the loosely coupled model and Web Services.

- Instance independence means that the architecture should support component-to-component communications, using both a synchronous and asynchronous model, and not require that the other component be in any particular state before getting the request, or message. If done right, all the services should be able to service any requesting component, asynchronously, as well as retain and manage state no matter what the sequencing is.

The need for loosely coupled architecture in your SOA is really not the question. If you have a SOA, you should have a loosely coupled architecture if done correctly. However, analysis and planning are also part of the mix. Understanding your requirements and how each component of your architecture should leverage the other components of your architecture. With a bit of up-front work, you'll find your coupling loose and your SOA successful. ☺

■ About the Author

David Linthicum is the CTO at Grand Central Communications (www.grandcentral.com), and a leading expert in the application integration and open standards areas. He has held key technology management roles with a number of organizations, including CTO of both Mercator and SAGA software. David has authored or coauthored 10 books, including the groundbreaking and best-selling Enterprise Application Integration released in 1998. His latest book is Next Generation Application Integration, From Simple Information to Web Services. You can reach David at dlinthicum@grandcentral.com.

■ ■ ■ dlinthicum@grandcentral.com

ONCE YOU'RE IN IT...

ONCE YOU'RE IN IT...



...REPRINT IT



Contact Kristin Kuhnle
201.802.3026
kristin@sys-con.com

SYS-CON
MEDIA

RePrints

The Gilbane Conference

ON CONTENT MANAGEMENT TECHNOLOGIES

April 11-13, 2005
San Francisco, California
The Palace Hotel



Lighthouse Seminars



Content technologies have become mainstream and need to be part of all major enterprise applications and integrated into IT architectures and infrastructures. Join us to learn what you need to know to formulate a successful strategy from our gathering of content technology experts and practitioners.

What you will take away from the conference:

Attendees benefit from an unbiased, deep, and up-to-date understanding of content management technologies, vendors, trends, and best practices, from the most experienced and respected experts in the field. Our speakers have implemented every kind of content management system across all industries, and have written the books and reports that others depend on. We are strictly neutral with regard to vendors, analysts, enterprises, and consultants. We ensure the focus is on what you need to know to successfully plan and implement content technology solutions.

"The Gilbane Conference has provided me with crucial information that I was looking for as my company rolls out its content management solution. I have been able to network with industry experts, the major content management technology players and other companies, like mine, that are faced with the challenge of effectively managing content," says Angie Khumdee, Senior Software Engineer, Motorola CGISS E-Business

Register NOW!

Register now for the **CONFERENCE PLUS PACKAGE** and receive a **FREE IPOD!** **LIMITED QUANTITY, REGISTER EARLY!**



GET STARTED to make vendor choices, benchmark your progress, and make strategic decisions.

LEARN from industry leaders and analyst about successful implementation of content technology through the unique combination of:

- **Project Management**
- **New Technology Trends**
- **Case Studies**

For more detailed information, please visit: **www.lighthouseseminars.com** or call **Joe Richard at 781.821.6734**

Platinum Sponsor



Media Sponsors



X_{ML} JOURNAL

SUPPLEMENT



HOME



ENTERPRISE SOLUTIONS



CONTENT MANAGEMENT



DATA MANAGEMENT



XML LABS

This Month

Stylus Studio 6 from Progress Software

Brian Barbash

Stylus Studio provides an integrated environment for XML development with broad support for the various technologies. This review covers its capabilities in some of the major areas.

An Easy Introduction to XML Publishing

PG Bartlett

In part I of this series, we discussed some of the key problems of capturing and sharing information – problems like delivering information to multiple types of media, making updates faster and easier, and reducing the cost and time to translate and publish.

Transforming XML-to-XML or -Text or -HTML

Deepak Vohra & Ajay Vohra

The Extensible Stylesheet Language Transformations (XSLT) specification provides for morphing XML documents into other XML documents.

An XML document can also be transformed into a format other than XML such as HTML or text. An XSLT processor is required for an XSLT transformation.

Process SOAP with VTD-XML

Jimmy Zhang

VTD-XML is the latest open-source, “non-extractive” XML processing API written in Java that overcomes many problems and issues of the status quo. The combination of its high performance, low memory footprint, random access, incremental update, and inherent persistence simply means this: With VTD-XML, application developers can finally unleash to the fullest extent the power of SOAP.

Accessing Resources: New Web Service Application Patterns for a Service-Oriented World

XML-Based Interop, Close up

In addition to the strategy side of Web services, there is also the protocol-oriented side of things, the XML side. Embracing not only XML itself but also the full range of mainstream XML-based technologies like XPath, XSLT, XML Schema, and SOAP, XML-Journal has been delivering insightful articles to the world of developers and development managers since the year 2000.

It is our privilege to bring XML-Journal directly to readers of Web Services Journal, and vice versa. Anyone already familiar with the Web services world of SOAP, UDDI, and WSDL will find here articles and features each month that will interest them – about the cutting-edge technologies and latest products that are changing not only our industry, but the way the world exchanges information. To make it easy for you to find your way around, we



Content Management:

Organization, dissemination, and presentation of information

Data Management:

Storage, transformation, representation, and general use of structured and unstructured data

Enterprise Solutions:

Systems and applications that manage mission-critical functions in the enterprise

XML Labs:

Product reviews, book reviews, tutorials, and standards analysis

Accessing Resources: New Web Service Application Patterns for a Service-Oriented World



WRITTEN BY
Paul Lipton & Igor Sedukhin

Mae West said, "When choosing between two evils, I always like to try the one I've never tried before."

But, sometimes, when choosing between two equally appealing options, the best policy is to take both.

WS-ResourceFramework and WS-Transfer, two new specifications for accessing XML representations of resources through Web Services have been announced. Although it's not uncommon to find competing specifications and standards in the world of Web Services, this time we shouldn't be forced to make an exclusive choice because WS-ResourceFramework and WS-Transfer can serve complementary roles.

What do we mean by resource representation?

It's often considered to be the representation of the state of a resource – state being one or more characteristics, or properties, associated with a resource. For example, if a customer record in a CRM database is the actual resource, its representation may be an XML document transferred over the network or the information returned as part of a SQL query. Either the XML document or the query result can be viewed as representations of the actual resource – the customer record. The set of information residing in the database can be viewed as the state of that customer record.

Web-centric folks sometimes draw additional distinctions between the representation and the resource itself. For example, the World Wide Web Consortium's (W3C) Technical Architecture Group (TAG) has noted more precisely that an XML document can represent a particular state of a resource at a particular time [<http://w3.org/TR/2004/REC-webarch-20041215/>]. So, while it's important to understand that there may be a difference between saying "access to the resource" and saying "access to the representation of the resource," it's certainly understandable that people are quite casual about the difference in day-to-day conversations.

State – Useful and Important

The concept of state is applicable in many areas. A typical example of state can be seen in many enterprise management systems. Such systems must monitor and manipulate IT resources such as disk drives, virtual machines, operating systems, routers and CPUs. Typically, each of these entities has a management state consisting of one or more characteristics associated with or representing the resource. For example, a wireless device can have characteristics such as on or off, a unique identifier of some sort, memory capacity, utilization information, geographic location, performance metrics, as well as many other useful characteristics.

An IT-enabled business relies on the correct behavior, availability and performance of its IT resources, which support the business systems and applications. Naturally, IT resources exist in numerous forms and at all levels of granularity in an organization. To access and manipulate these resources, software developers typically master specific application programming interfaces (APIs) that are optimized to access certain types of resources.

Going back to the earlier example, the contents of a row in a CRM database customer table represent the state of a specific IT business resource (a specific customer record). To access that business resource, a software developer would have to understand how a customer record is represented (the database schema) and would also need to use database access APIs defined for the specific programming language or platform being used. For example, Java programmers often use JDBC or Entity Beans to access the contents of rows in database tables. Programmers working in the Microsoft environment might use ODBC or ADO. Traditionally, different APIs and different approaches are used to access different types of resources. This increases the complexity of the software and the cost of development and integration. Web Services may provide a way to make it simpler.

State in a Service-Oriented Architecture

How does this change in a service-oriented architecture

(SOA)? In service-oriented business applications (SOBA), many important business resources are likely to be encapsulated and virtualized using Web Services and service-oriented design principles. Business services operate on XML representations of business-related information such as account balances or order quantity. At the same time, certain business services may also have an intrinsic operational state of interest to other business services that can be represented in XML as well. For example, a flight reservation business process encapsulated as a service (scripted in BPEL) might have an operational state such as “looking for matching itineraries” or “awaiting confirmation from the credit bureau.”

Service-oriented management applications (SOMA) monitor and control the SOBA to optimize operation according to the policies in effect. The SOMA, not unlike the SOBA, must be able to access the intrinsic operational state of business services along with relevant business information. For example, the SOMA may detect that a flight reservation process isn't meeting its performance objectives with respect to an important client because the credit check service isn't responding. It may, then, take corrective action. All of this requires access to a number of IT business resources and their state. The point is that SOBA and SOMA have strikingly similar requirements – to access representations of various resources.

The ideal situation would be for the underlying software platform to provide a common set of mechanisms for accessing representations of resources regardless of the type of the resource. The benefit of shared mechanisms is that it makes it possible to architect and build SOBAs that are innately manageable by design since the underlying software platform can provide common mechanisms to enable the exchange of state representations for both business and management functions. Two essential specifications have been announced that help provide this common mechanism:

- WS-Transfer, which defines how to access a complete representation of a resource using Web Services.
- WS-ResourceProperties, part of the WS-ResourceFramework family of specifications, which defines how to access individual properties of a resource using Web Services.

Two additional specs are critically important for this common mechanism to work:

- WS-Addressing, which defines how to address Web Services including those Web Services that represent resources.
- WS-MetadataExchange, which defines how to retrieve various metadata documents from a Web Service. The metadata documents may contain descriptions of message exchanges (WSDL), XML Schemas, policy assertions (WS-Policy), etc.

Representation of Resources in XML

The best way to understand how a set of specifications compliment each other is to understand how information is exchanged in one simple example. Here we'll use an example of accessing representations of a customer record. The record contains the customer's name, address and an optional set of numeric identifiers of open support calls that the customer has initiated. The record is represented by the XML fragment below:

```
<x:Customer xmlns:x="http://crm.mycorp.com/data">
  <x:Name>
```

```
<x:First>John</x:First>
  <x>Last>Doe</x>Last>
</x:Name>
  <x:Address>
    <x:Street>123 No Outlet Ln.</x:Street>
    <x:Zip>12345</x:Zip>
  </x:Address>
  <x:OpenCall>230</x:OpenCall>
  <x:OpenCall>320<x:OpenCall>
</x:Customer>
```

To operate on representations of customer records such as the one above, an application would need to retrieve and update the records using their XML representations.

Before allowing access to XML representations of customer records, an XML Schema document could be created to describe and validate these XML representations. The schema document might look like this:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://crm.mycorp.com/data"
  xmlns:x="http://crm.mycorp.com/data"
  elementFormDefault="qualified">
  <xs:element name="Name">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="First" type="xs:string"/>
        <xs:element name="Last" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Address">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Street" type="xs:string"/>
        <xs:element name="Zip" type="xs:positiveInteger"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="OpenCall" type="xs:nonNegativeInteger"/>
  <xs:element name="Customer">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="x:Name"/>
        <xs:element ref="x:Address"/>
        <xs:element ref="x:OpenCall"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

This is a simplified schema of an XML representation of a customer record. Note the use of top-level XML element declarations for Name, Address and OpenCall. These are intended to declare the XML elements that could be interpreted as properties of a customer record. The top-level element declarations (also known as Global Element Declarations

or GEDs) are needed so that Name, Address and OpenCall can be used in various XML fragments without necessarily being contained in a Customer element.

Web Services That Represent Resources

A Web Service could represent a resource by offering access to an XML representation of the resource, operations on the resource and possibly notifications of various conditions or changes in the state of the resource. Let's assume that the address of a Web Service endpoint that represents a resource is known and represented using WS-Addressing. In WS-Addressing this is known as an endpoint reference or EPR. Simply put, WS-Addressing specifies an EPR along with the optional Web Service-specific data contained in the ReferenceProperties or ReferenceParameters elements. The beauty is that the EPR can be specified in a transport neutral fashion – very much in the SOAP/XML tradition.

```
<wsa:EndpointReference xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:x="http://crm.mycorp.com/data">
  <wsa:Address>http://crm.mycorp.org/endpoint</wsa:Address>
  <wsa:ReferenceProperties>
    <x:CustomerID>001</x:CustomerID>
  </wsa:ReferenceProperties>
</wsa:EndpointReference>
```

This EPR specifies a reference to a Web Service endpoint that represents a customer record resource identified by the CustomerID value 001. The EPR could be obtained by various discovery mechanisms such as WS-Discovery, UDDI or domain-specific mechanisms such as relationships defined between manageable resources in a SOMA. It is analogous to how a URL to a web page can be discovered through a wide variety of means like search engines, hyperlinks, etc.

Locating the Resource Representation Schema

If the schema of the resource representation is unknown, one can try to request it from the Web Service endpoint that represents the resource. To do so, send a Get request as defined by the WS-MetadataExchange spec. The following is an example of such a request to the Web Service endpoint that represents the customer record identified by the CustomerID value 001.

```
<s12:Envelope
  xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
  xmlns:wsx='http://schemas.xmlsoap.org/ws/2004/09/mex'
  xmlns:x="http://crm.mycorp.com/data">
  <s12:Header>
    <wsa:To>http://crm.mycorp.org/endpoint</wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Request
    </wsa:Action>
    <wsa:MessageID>uuid:...A</wsa:MessageID>
    <wsa:ReplyTo>
```

```
<wsa:Address>http://client/app</wsa:Address>
</wsa:ReplyTo>
    <x:CustomerID>001</x:CustomerID>
  </s12:Header>
  <s12:Body>
    <wsx:GetMetadata>
      <wsx:Dialect>
        http://www.w3.org/2001/XMLSchema
      </wsx:Dialect>
      <wsx:Identifier>
        http://crm.mycorp.com/data
      </wsx:Identifier>
    </wsx:GetMetadata>
  </s12:Body>
</s12:Envelope>
```

The response with the schema shortened for brevity would look like this:

```
<s12:Envelope
  xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  xmlns:wsx='http://schemas.xmlsoap.org/ws/2004/09/mex' >
  <s12:Header>
    <wsa:To>http://client/app</wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Response
    </wsa:Action>
    <wsa:MessageID>uuid:...B</wsa:MessageID>
    <wsa:RelatesTo>uuid:...A</wsa:RelatesTo>
  </s12:Header>
  <s12:Body>
    <wsx:Metadata>
      <wsx:MetadataSection
        Dialect='http://www.w3.org/2001/XMLSchema'
        Identifier='http://crm.mycorp.com/data'>
      <xs:schema targetNamespace="http://crm.mycorp.com/data">
      <xs:element name="Customer">
        [. . . XML Schema as shown earlier . . .]
      </xs:element>
      </xs:schema>
    </wsx:MetadataSection>
  </wsx:Metadata>
  </s12:Body>
</s12:Envelope>
```

Retrieving the Resource Representation

If an EPR of a Web Service endpoint that represents a customer record resource is known, a WS-Transfer Get request can be used to obtain an XML representation of the customer record resource. An EPR specifies how a request message needs to be addressed. An EPR is similar to an address book entry and the message is like a letter. EPR says what information to write on the envelope for the letter to

reach the intended recipient. In this case the recipient is the customer record resource.

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:x="http://crm.mycorp.com/data">
  <s:Header>
    <wsa:To>http://crm.mycorp.org/endpoint</wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
    </wsa:Action>
    <wsa:MessageID>uuid:...A</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://client/app</wsa:Address>
    </wsa:ReplyTo>
    <x:CustomerID>001</x:CustomerID>
  </s:Header>
  <s:Body/>
</s:Envelope>
```

This request is sent to a Web Service endpoint at a `http://crm.mycorp.org/endpoint` URL that represents a customer record identified by the `CustomerID` value 001. The Action says that this is a request to retrieve the resource representation. The response with the resource representation would look as follows. Note that the entire customer record is returned.

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s:Header>
    <wsa:To>http://client/app</wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
    </wsa:Action>
    <wsa:MessageID>uuid:...B</wsa:MessageID>
    <wsa:RelatesTo>uuid:...A</wsa:RelatesTo>
  </s:Header>
  <s:Body>
    <x:Customer
      xmlns:x="http://crm.mycorp.com/data">
    <x>Name><x:First>John</x:First><x>Last>Doe</x>Last></x>Name>
    <x:Address><x:Street>123 Nooutlet Ln.</x:Street>
    <x:Zip>12345</x:Zip></x:Address>
    <x:OpenCall>230</x:OpenCall>
    <x:OpenCall>320<x:OpenCall>
      </x:Customer>
    </s:Body>
  </s:Envelope>
```

Accessing Individual Resource Properties

While WS-Transfer offers a simple mechanism to get the entire resource representation, there are cases where the resource representation might be very large or contain many individual XML elements that are of no interest to the client application. In other words, the entire representation may not be needed or desired when the client applica-

tion knows exactly which properties it's interested in. A SOMA would be a good example of this; perhaps requiring representations of selected metrics, but not necessarily all the metrics. In this case, the overhead of obtaining, serializing and sending all the metrics when the application knows exactly what it's interested in is simply not reasonable.

Similarly, a SOBA might not want to (or be permitted to) retrieve all the properties in a resource representation. For example, an insurance application may not need to know the number of open calls for every customer record in the CRM when it's simply processing addresses. In such cases, where only individual properties or a selection of those properties are required, the WS-ResourceProperties specification defines request and reply message exchanges: `GetResourceProperty`, which retrieves a single property, `GetMultipleResourceProperties`, which retrieves many properties, and `QueryResourceProperties`, which selects properties based on an XPath expression.

Here is an example of requesting Name and OpenCall properties from a Web Service endpoint that represents a particular customer record.

```
<s12:Envelope
  xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wsrp="http://docs.oasis-open.org/wsrf/2004/06/wsrf..."
  xmlns:x="http://crm.mycorp.com/data">
  <s12:Header>
    <wsa:To>http://crm.mycorp.org/endpoint</wsa:To>
    <wsa:Action>
      http://docs.oasis-open.org/wsrf/2004/06/WS-
      ResourceProperties/GetMultipleResourceProperties
    </wsa:Action>
    <wsa:MessageID>uuid:...A</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://client/app</wsa:Address>
    </wsa:ReplyTo>
    <x:CustomerID>001</x:CustomerID>
  </s12:Header>
  <s12:Body>
    <wsrp:GetMultipleResourceProperties>
      <wsrp:ResourceProperty>x:Name</wsrp:ResourceProperty>
      <wsrp:ResourceProperty>x:OpenCall</wsrp:ResourceProperty>
    </wsrp:GetMultipleResourceProperties>
  </s12:Body>
</s12:Envelope>
```

The reply with a bag (collection) of specified properties looks like this:

```
<s12:Envelope
  xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wsrp="http://docs.oasis-open.org/wsrf/2004/06/wsrf..."
  xmlns:x="http://crm.mycorp.com/data">
  <s12:Header>
    <wsa:To>http://client/app</wsa:To>
    <wsa:Action>
      http://docs.oasis-open.org/wsrf/2004/06/WS-
      ResourceProperties/GetMultipleResourcePropertiesResponse
```

```

</wsa:Action>
<wsa:MessageID>uuid:...B</wsa:MessageID>
<wsa:RelatesTo>uuid:...A</wsa:RelatesTo>
</s12:Header>
<s12:Body>
  <wsrp:GetMultipleResourcePropertiesResponse>
    <x:Name><x:First>John</x:First><x>Last>Doe</x>Last></x:
      Name>
    <x:OpenCall>230</x:OpenCall>
  </wsrp:GetMultipleResourcePropertiesResponse>
</s12:Body>
</s12:Envelope>

```

Of course, the resource representation schema might not always need to be dynamic. It may be sufficient to interrogate the resource representation schema at design-time along with interrogating the operations and binding details of the Web Service endpoint that represents the resource. The WSDL document describes the Web Services and their endpoints. WS-ResourceProperties defines a way to attach the resource representation schema to the portType declaration in WSDL. This can be handy. Using the design-time information, a client could build proxy code to simplify access and update a resource's individual properties.

Updating the Resource Representation

Updating resource representations is similar in principle to accessing a resource's representation. The EPR to the Web Service endpoint that represents the resource has to be known along with when the whole resource representation has to be updated. One can send a Put request defined by WS-Transfer. It is also possible to update an individual property or a set of properties using the SetResourceProperties message exchange defined by WS-ResourceProperties. One example where this might be useful is in the case of multi-value properties (arrays) such as the OpenCall property in the customer record example. If the values of the OpenCall property in the customer record have to be added or removed, it may be more efficient to do it using WS-ResourceProperties. Any requested updates are then validated by the customer record Web Service endpoint against the resource representation schema. The service responds with an indication of success or failure.

Creating and Deleting Resources

WS-Transfer describes an explicit resource factory approach by defining a Create message exchange for creating resources. Similarly, WS-Transfer defines a Delete message exchange to destroy resources. This makes particularly good sense with resources like customer records, files and documents.

The creation and deletion of resources is beyond the scope of WS-ResourceProperties. However, the WS-ResourceLifetime specification (also part of the WS-ResourceFramework family of specifications) does define a mechanism for destroying resources via its Destroy request. But there's no request currently defined in WS-ResourceLifetime or elsewhere in the WS-ResourceFramework for creating resources. This decision was deliberate. An implicit resource factory approach where resources are created as a side effect of some other message exchange was preferred.

WS-ResourceLifetime defines how to set and retrieve the expiration time of a resource. The resource is automatically destroyed when its lifetime expires. However, even if a resource has a scheduled expiration time, it may still be possible to explicitly destroy that resource. This sort of mechanism is suitable for resources such as subscriptions and registrations, and is applicable in both SOBA and SOMA.

Conclusion

WS-Transfer and WS-ResourceProperties define specific actions for generalized resource access using transport-independent SOAP messages. These kinds of interactions are now well-defined leaving resource representation to the applications. It is also interesting how seemingly competitive specifications can be meaningfully used together to create flexible and functional implementations of Web Services representing various business resources.

Superficially, both WS-Transfer and WS-ResourceProperties provide similar capabilities, but their basic design principles make them more or less suited to certain types of applications. For example, management applications may benefit from the flexibility of WS-ResourceProperties, which was leveraged in the OASIS Web Services Distributed Management (WSDM) specification. However, simpler distributed business applications might be developed more quickly and easily with WS-Transfer, leaving room to compose more advanced Web Services interaction aspects later. For example, other specifications under development such as WS-Enumeration, which is beyond the scope of this article, may be applied. Together, WS-Transfer, WS-ResourceProperties and WS-ResourceLifetime offer an appropriate and broad range of options for accessing representations of resources using Web Services – and are often complimentary rather than competitive. ☛

AUTHOR BIOS

Paul Lipton is a senior architect and technology strategist in the Office of the CTO at Computer Associates. As an architect and developer of enterprise systems for more than 20 years, he has worked closely with key CA customers to solve important business challenges by creating mission-critical distributed solutions. He has represented CA in numerous standards organizations such as W3C, OASIS and the Java Community Process, and he is currently serving on standards committees involved in the definition of new Web Services standards for management, orchestration and choreography. He is a highly sought-after author and conference speaker, and has shared his knowledge with appreciative audiences on such diverse topics as Web Services, Java, .NET, management and security, EAI, XML transformation frameworks, object databases, distributed systems, On-Demand computing and wireless.

Igor Sedukhin is a senior architect and technology strategist in the Office of the CTO at Computer Associates. He is responsible for CA's research in areas including adaptive distributed computing, peer-to-peer grid networks and dynamic service agreements. Mr. Sedukhin is an active member of the World Wide Web Consortium, OASIS, Global Grid Forum and open source organizations, with a focus on the interoperability of Web Services. He joined CA in 1997, and has been a senior architect for XML Web Services tools, storage and information management solutions. He received his B.S. in applied mathematics and M.S. in computer science from Novosibirsk State University in Novosibirsk, Russia, where he was extensively involved in the applied research of parallel, high-performance computing and algorithms.

■ paul.lipton@ca.com

■ sedukhin@ca.com

Offer subject to change without notice

Learn Web Services. Get a New Job!

Only \$69.99

1 year (12 issues)*

* Newsstand price \$83.88 for 1 year



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



**Subscribe
ONLINE**
www.wsj2.com or
CALL
888 303-5252

*Offer subject to change without notice

**The Best
.NET
Coverage
Guaranteed!**

Subscribe today to the world's leading Web Services resource

- 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!





WRITTEN BY BRIAN BARBASH

Stylus Studio 6 from Progress Software

An integrated tool with breadth

XML Development – the term can mean many different things given the technologies currently available. At the center of it all is XML Schemas, DTDs and instance documents. Building out from the base there's XSL, Web Services and XQuery just to name a few. Because of this, it's not uncommon for a developer to have several tools, each specialized in one of those technologies. Stylus Studio, however, provides an integrated environment for XML development with broad support for the various technologies. This review covers its capabilities in some of the major areas.

XML and XML Schema Editing

At the center of all XML development is XML itself, schemas, DTDs and instance documents. To work with the core, Stylus Studio offers developers a set of easy-to-use tools for editing XML documents, XML Schemas and DTDs. When working with XML documents, the data is presented in three different views:

- Text – Standard text editor with XML syntax highlighting, keyword completion and validation;
- Tree – A tree view of document nodes that can be expanded and collapsed;
- Grid – A specialized view that displays XML documents in a spreadsheet-like view; excellent for working with documents with repeating or list nodes.

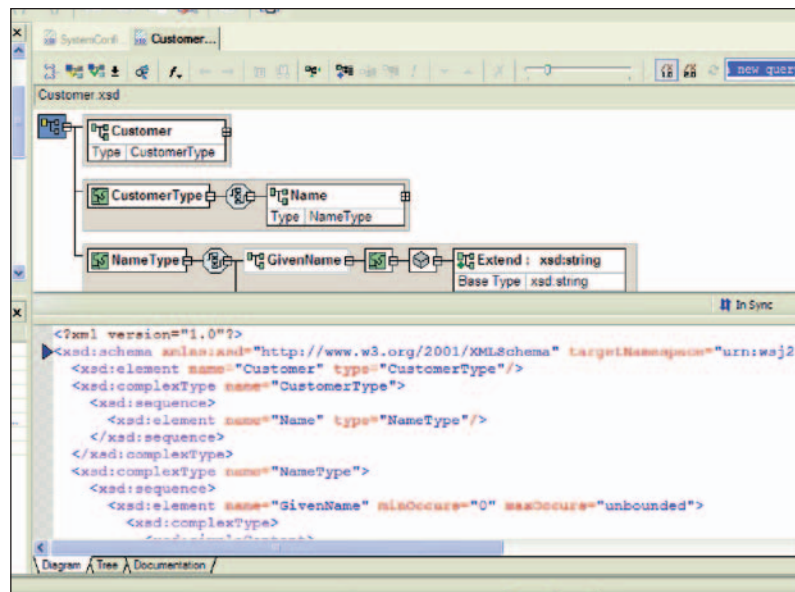


Figure 1 • XML Schema Editing

XML Schemas are also presented in the Text and Tree views, but the Diagram view is an added option for schemas (Figure 1). The Diagram view is a graphical representation of schema elements and structures. One simple but effective feature of the Diagram view is the ability to zoom in and out, which lets developers see the big picture of the schema documents they're working on. While the document is edited in any view, all other views are updated on-the-fly.

Another nice feature of the XML Schema editing environment is the ability to create schema documentation simply by clicking on the documentation tab at the bottom of the editing window. The result is an HTML document with clear

definitions of each element and type, the text of the definition as well as sample XML text for additional clarity.

XQuery and SQL/XML

Stylus Studio provides two ways to query data sources: XQuery for XML documents and SQL/XML for relational databases. XQuery documents can be created and edited using the Source editor and graphical Mapper editor. Stylus Studio creates a tree representation in the Source editor for each XML instance document referenced in the XQuery code. Nodes from these structures can be dragged into the query source to build any XPath statements quickly. While typing in the Source editor, a context-sensitive popup appears

AUTHOR BIO

Brian R. Barbash is the product review editor of *Web Services Journal*. He is a senior consultant and technical architect for the Envision Consulting Group, a management consulting company focusing on contracting, pricing and account management in the pharmaceutical industry.

containing functions and any previously defined variables.

Developers can also edit XQuery documents in the graphical Mapper view. As with the Source editor, referenced XML instance documents are presented as tree structures. The Mapper also displays the return document structure as a tree. Nodes from source documents can be dragged to the return structure to build the appropriate output. XQuery functions are applied by inserting graphical representations of the desired function into the Mapper and attaching the appropriate source and target nodes. Stylus Studio supports the July 2004 draft specification of XQuery 1.0.

SQL/XML statements can be easily created using the DB-to-XML Data Source editor. It should be noted that this editor requires a JVM version 1.4 or higher on the machine that Stylus Studio is running on so JDBC can manage the connections. Stylus Studio ships with support for IBM DB2, Oracle, SQL Server,

“Today there are so many facets to XML development that finding a single tool that effectively provides almost all of the functionality of the different areas is difficult.”

Informix and Sybase. It also provides ODBC connectivity options through the DataDirect SequeLink server. Once connected to the host database, schemas, tables and columns are presented to the developer in a tree structure. When a table is dragged into the text editor, the statement can be built as SQL/XML. From here, the source code can be edited to structure the resulting XML as needed. As in other editing environments in Stylus Studio, a popup window of relevant functions, tables and column names is available when typing. Once saved, the SQL/XML statement can be used as a source document for other operations in Stylus Studio.

Web Services

Stylus Studio assists Web Services developers in creating services by acting as a stripped-down Web Services client. It's easy to create

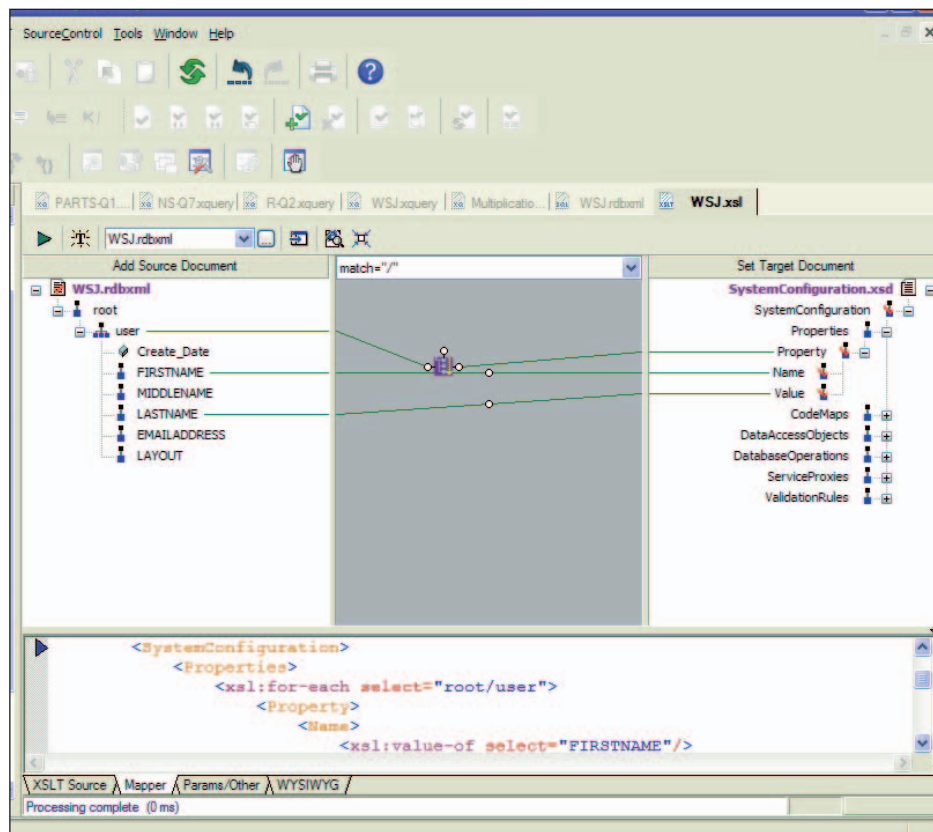


Figure 2 • XSLT Mapper

a call to a Web Service; simply identify the URL of the WSDL document for the desired service and Stylus Studio creates the requisite SOAP message. WSDL documents can be retrieved from either a file system or UDDI registry. The service call can be saved and reused as an XML source document for other operations in Stylus Studio; the call parameters retain the values set at the time saved.

XSLT and XML Mapping

XML document mappings are easily created in Stylus Studio using the XSLT editor. There are four views to this editor:

1. XSLT Source
2. Mapper
3. Parameters
4. WYSIWYG

The XSLT Source, Mapper and WYSIWYG are linked so that changes made in one are immediately reflected in the other. Figure 2 shows a simple XSLT mapping. The source for the transformation, which appears on the left side of the screen, reuses a previously created SQL/XML document. The target document, represented by the tree structure on the right,

is based on a previously defined schema. The center pane shows any transformation functions in graphical form.

In the example shown, the function is a simple `<xsl:for-each>` statement that copies all user records to the target document. Additional functions can be added to the transformation either by typing the appropriate XSL code or selecting the desired function from a popup list of options.

Summary

Today there are so many facets to XML development that finding a single tool that effectively provides almost all of the functionality of the different areas is difficult. Stylus Studio from Progress Software, however, makes a very good attempt and delivers on that idea. It has strong support for the core – XML and XML Schema editing, good tools for XML mapping and transformation via XSL. It adds capabilities for newer technologies such as XQuery and supports Web Services. Overall, Stylus Studio is an excellent item for the XML developer's toolbox. ☛

bbarbash@sys-con.com



HOME



ENTERPRISE SOLUTIONS



CONTENT MANAGEMENT



DATA MANAGEMENT



XML LABS



WRITTEN BY PG BARTLETT

An Easy Introduction to XML Publishing

Part 2 of a 4-Part Series:

Essentials of a Publishing System

In part 1 of this series, we discussed some of the key problems of capturing and sharing information – problems like delivering information to multiple types of media, making updates faster and easier, and reducing the cost and time to translate and publish.

Given those challenges, what should we do?

In this part, we'll describe the essentials for solving these problems, which include building a "single source" of information that eliminates redundancy, creating information in reusable modules, and automatically assembling and publishing information for multiple audiences and multiple media.

Let's take a trip through the Wayback Machine to 1960.

You find yourself responsible for publishing a technical manual. You begin the project by figuring out what information the book will contain and assigning various sections to subject matter experts (SMEs). The SMEs are mostly engineers, and they write out their content in longhand on foolscap. (If you don't know that "foolscap" is writing paper, congratulate yourself on your youth and vitality.)

The secretarial pool types out the handwritten notes from the engineers. An editor – that's your job too – reviews

and edits those typewritten notes and then sends them back to the secretarial pool to re-type.

You assemble all of the sections in order and proofread them again to make sure everything flows and that you mark all the typing errors. Then you add typesetting instructions (also known as "markup") in the margins to indicate the formatting of each section of the document. The markup specifies font, line measure, leading and other formatting instructions that you copy from a set of corporate guidelines.

"By reusing information instead of re-creating it, you can create a single source of information so that making just one change can update multiple documents"

You then send the typewritten documents with your markup to a printer. Their typesetter works on a Linotype machine to create typeset versions of your documents. The printer returns your copy in "galley," which are just long rolls of paper with your document printed out in a single column.

Then a designer or layout artist cuts up the galley and lays them out onto pages. An editor – you again – reviews the layouts both for typesetting accuracy and design. Then someone – poor you – scans through the layouts and builds a table of contents and an index, which you then send to the typing pool, review for accuracy, send to the typesetter, give to the designer to lay out on pages and then review for accuracy again.

After that, you send the entire book to the printer who makes all of the corrections, lays out the type to match your page layouts, and produces a copy of the book for someone – you know who – to check again. You send the book back, the printer makes any further corrections, and finally – finally! – he prints it.

Mercifully, your book contains no illustrations.

Back to the Future

Get back into the Wayback Machine and return to the present. Same job. Different millennium. Now you have desktop publishing tools that let you do everything yourself. The specialists are all gone except for the SMEs. No secretaries, no typesetters, no layout designers. There are other benefits: you no longer have to create a table of contents and index manually and you have much less proofreading to do.

What work remains? You create content, apply formatting and lay out pages. But there's more.

AUTHOR BIO

PG Bartlett is vice president of product marketing at Arbortext, where he is responsible for corporate positioning, marketing strategy and product direction, and driving an integrated communication strategy. Bartlett joined Arbortext in 1994, bringing more than 18 years of experience in both technical and marketing positions at leading-edge high technology companies. He is a frequent presenter at major industry events and has been invited to speak and chair sessions at Comdex, Seybold Seminars, XML conferences, AIM conferences, and others.

In the last 45 years, the demands on you have changed. It's no longer enough to produce a revision of the book every year or two; update cycles are now measured in months instead of years. You have to produce the book both in print and online. The book is translated into three additional languages and some day you'll have to produce it in eight. You're under pressure to produce customized versions for various types of readers, and some day you know that you will have to let your customers build their own versions. And you have to do all this while facing escalating pressure to deliver the book faster at lower cost with fewer people.

You need to find quicker, better ways to get your job done. So you take a page from the office automation projects of the past – the projects that eliminated adding machines, pencils and handwritten account ledgers. You need to eliminate the redundancy and rework that comes from copying and pasting the same content. You need to stop copying data from its original source into your documents. You need automation – lots of it, everywhere that it makes sense.

And you need to achieve all of that while gaining the flexibility to produce an ever-wider variety of information products.

So just as we are inspired by the benefits of office automation, we are also inspired by the approach. The key to successful automation, regardless of what process you're automating, is an absolutely consistent structure and data format. While the formatting of an individual business document suggests a structure that's obvious to anyone looking at it, authors freely alter structure and formatting to suit their circumstances and tastes. The result is that documents of the same type will have similar but not identical formatting and similar but not identical structures. These inconsistencies, however small, make automation impossible.

Another key to successful office automation is modularization. By splitting up data into pieces, it's not only easier to manage but also easier to incorporate into various reports. Translating that to documents, it means splitting up them into reusable components big enough to be worth managing separately but small enough to reuse in multiple instances.

By reusing information instead of re-creating it, you can create a single source of information so that making just one change can update multiple documents. A single source

also leads to the reduction or elimination of redundancy, which not only lets you reduce translation costs to only those parts that have changed, but also ensures the integrity and accuracy of your information. Can you imagine the problems your colleagues in accounting would face if data such as net income appeared separately in multiple locations, each with no connection to the other? How much work would it be to track them all down for updating? What are the risks of missing one?

Modularization is also critical for personalization. Creating information out of smaller components lets you set up a system to assemble those components dynamically to suit the needs of various audiences.

“Also key to automation is separating the information from its presentation”

For modularization to work the components must be interchangeable and they must fit correctly into their “parent” document. For example, you may choose to create two different sizes of modules such as warnings and topics, where warnings fit into topics and topics fit into books.

Also key to automation is separating the information from its presentation. In office automation, that means storing data in databases and extracting it for various purposes. Accountants can produce tabular reports, charts and graphs in virtually endless combinations, even though the data in its raw form would be unintelligible to them. Applying this principle to documents means that you can use the same information in different kinds of documents with different formatting, all without touching the information itself.

Let's review the essentials of your automated publishing system:

- Modularization – enables reuse, single-sourcing and personalization
- Structure and consistency – essential for automation
- Separation of content from presentation – required to deliver the same information to multiple different media types of media
- Automation – helps you assemble information for multiple audiences and publish

to multiple types of media without human intervention

- Single source – helps you eliminate the ongoing time and expense of maintaining redundant information while ensuring its integrity

You won't be surprised to learn that XML will play a key role in your automated publishing system. XML makes structure explicit and ensures the absolute consistency of your documents. In that respect, XML is unique. No other standard data format (except SGML, XML's predecessor) can represent any kind of information – text, data and graphics.


XML also enables the separation of content from its presentation. XML represents information in a “media neutral” form that's not constrained by the limitations and capabilities of any particular medium, so you can create information in its “pure” form and process it separately to produce information products.

One of the advantages of this separate process is that through automation, the information can be presented with absolutely consistent formatting regardless of author, and it can take full advantage of the capabilities of each medium.

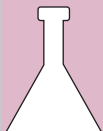
Through XML, you can also specify the size of your reusable information modules. And you can do this in a consistent way so that you can easily interchange one module for another.

Another key ingredient to an automated publishing system is the software that makes it all work – software that lets you create information in reusable modules of XML, assemble those modules for specific audiences and publish the result for multiple types of media.

The final piece of the puzzle involves changing the roles and responsibilities of the people involved in creating and improving your information. This article went into great detail about how roles changed between 1960 and now, and in a future installment you will see that roles will change yet again..

In part three, we'll look at the process of developing a solution: developing data models (DTDs, schemas), designing stylesheets and integrating various tools and technologies. 

 pgb@arborbortext.com



WRITTEN BY Deepak Vohra and Ajay Vohra

Transforming XML-to-XML or -Text or -HTML

XSLT morphs XML documents into other XML documents

The Extensible Stylesheet Language Transformations (XSLT) specification provides for morphing XML documents into other XML documents. An XML document can also be transformed into a format other than XML such as HTML or text. An XSLT processor is required for an XSLT transformation. Some of the commonly used XSLT processors include Xalan-Java, Oracle XSLT Processor for Java and the JAXP XSLT transformer. A stylesheet is used to transform an XML document. The elements of a stylesheet are in the XSLT namespace <http://www.w3.org/1999/XSL/Transform>.

With XSLT an XML document may be converted to another XML/HTML/text document.

The elements and attributes in an XML document get modified with an XSLT transformation.

In this tutorial some of the commonly required transformations are discussed. Having a Java application to transform an XML document with XSLT is a prerequisite. An example XML document will be used as the basis for the XSLT transformations: `catalog.xml`, the example XML document, is shown in Listing 1.

Listing 1 `catalog.xml`

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

```
xmlns="http://www.w3.org/2001/
XMLSchema-Instance">
  <journal title="Java Technology"
publisher="IBM developerWorks">
    <article level="Intermediate"
      date="January-2004"
      section="Java Technology">
      <title>Service Oriented
Architecture Frameworks</title>
      <author>Naveen Balani</
author>    </article>

    <article level="Advanced"
      date="October-2003"
      section="Java Technology">
      <title>Advance DAO
Programming</title>
      <author>Sean Sullivan</author>
      </article>

    <article level="Advanced"
      date="May-2002" section="Java
Technology">
      <title>Best Practices in EJB
Exception Handling</title>
      <author>Srikanth Shenoy </
author>    </article>

    <article level="Advanced" date="
May-2002" section="Java
Technology">
      <title>Best Practices in EJB
Exception Handling</title>
      <author>Srikanth Shenoy </
author>    </article>
  </journal>
</catalog>
```

The tutorial is structured into the

following sections:

- Identity Transformation
- Removing Duplicates
- Sorting Elements
- Converting to HTML
- Merging Documents
- Obtaining Element/Attribute Values with XPath
- Filtering Elements
- Copying Nodes
- Creating Elements and Attributes
- Outputting an XML Document

The different sections explain only the XSLTs required to transform the example XML to XML/text/HTML documents. The procedure is explained in the frame:

```
Import the javax.xml.transform
package.
import javax.xml.transform.*;
Import the javax.xml.parsers package.
import javax.xml.parsers.*;
Import the org.xml.sax and org.w3c.
dom package classes.
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.w3c.dom.Document;
import org.w3c.dom.DOMException;

Import the DOMSource, Stream-Source
and StreamResult classes.

import javax.xml.transform.dom.
DOMSource;
import javax.xml.transform.stream.
```

AUTHOR BIO

Deepak Vohra is a
Sun Certified Java 1.4
Programmer and a Web
developer.

Ajay Vohra is a senior
software engineer with
Compuware.

```

StreamSource;
import javax.xml.transform.stream.
    StreamResult;

Create a DocumentBuilderFactory object.

DocumentBuilderFactory factory =
    DocumentBuilderFactory.newInstance();

Copy the example XML document catalog.xml
to the c:/input directory.
Create a DocumentBuilder object and parse
the XML document to be transformed.

DocumentBuilder builder = factory.new
    DocumentBuilder();
    document = builder.parse(new
        File("c:/input/catalog.xml"));

Create a TransformerFactory object.

TransformerFactory tFactory =
    TransformerFactory.newInstance();

```

Copy the XSLT stylesheet to be used for transformation to the c:/input directory. Create a Transformer object from the XSLT to be used for transformation.

```

StreamSource stylesource = new StreamSource
(new File("c:/input/stylesheet.xslt"));
    Transformer transformer =
        tFactory.newTransformer(stylesource);

```

Create a DOMSource object for the example XML Document object.

```
DOMSource source = new DOMSource(document);
```

Create a StreamResult object for the XSLT transformation output.

```
StreamResult result = new
    StreamResult(System.out);
```

Transform the example XML document with an XSLT.

```
transformer.transform(source, result);
```

The output from the XSLT transformation is displayed in the System.out.

Identity Transformation

The Identity transformation in XSLT copies

the input XML document to the output document.

The structure and values of the elements and attributes in the XML document aren't modified.

An example XSLT for identity transformation is shown in Listing 2.

Listing 2 Identity Transformation

```

Stylesheet
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
    <xsl:output method="xml" version="1.0"
        indent="yes"/>
    <xsl:template match="@* | node()">
        <xsl:copy>
<xsl:apply-templates select="@* | node()"/>
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>

```

The XPath expression '@*[node()]' selects all the element and attribute nodes.

The identity transformation could be applied to modify the encoding, DOCTYPE or indentation.

Removing Duplicates

An XML document may have duplicate elements. The example XML document has duplicate "article" elements. The following XSLT outputs non-duplicate "article" titles.

Listing 3 Removing Duplicates

```

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
    <xsl:output method="xml" version="1.0"
        omit-xml-declaration
        ="yes"/>
    <xsl:template match="/">
        <xsl:variable name="unique-list"
            select="//title[not(.=following::
                title)]" />
        <xsl:for-each select="$unique-list">
<xsl:copy>
<xsl:apply-templates/>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>

```

The XPath expression '//title[not(.=following::title)]' selects non-

duplicate 'title' elements.

The output from the XSLT is the non-duplicate article titles as in Listing 4.

Listing 4 Title Elements with Duplicates Removed

```

<title>Service Oriented Architecture
Frameworks</title>
<title>Advance DAO Programming</title>
<title>Best Practices in EJB Exception
Handling</title>

```

In subsequent sections the XML document whose duplicate element have been removed will be used.

Sorting Elements

The XSLT xsl:sort is used to sort a group of elements. The attribute order of the xsl:sort element specifies the sorting order: ascending or descending. The data-type attribute (number or text) specifies the data type of the element to be sorted. For instance, the "title" elements in the example XML document are sorted with an XSLT, which is listed in Listing 5.

Listing 5 Sorting

```

<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/
Transform"
    version="1.0">
    <xsl:output method="xml" omit-xml-
        declaration="yes"/>
    <xsl:template match="/catalog/journal">
        <xsl:apply-templates>
            <xsl:sort select="title"
                order="ascending"/>
        </xsl:apply-templates>
    </xsl:template>
<xsl:template match="article">
    Title: <xsl:apply-templates
        select="title"/>
    </xsl:template>
</xsl:stylesheet>

```

The output from the XSLT is a sorted list of article titles in ascending order. The Sorting Elements section in XSLT is also an example of XML-to-text transformation.

Listing 6 Sorted List

```

Title: Advance DAO Programming
Title: Best Practices in EJB Exception
Handling

```


Title: Service Oriented Architecture Frameworks

Conversion to HTML

The data in an XML document may have to be presented as an HTML document.

The following XSLT converts the example XML document to an HTML document.

Listing 7 Conversion to HTML

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:output method="html"/>
<xsl:template match="/catalog/journal">
<html>
<head>
<title>Catalog</title>
</head>
<body>
<table border="1" cellspacing="0">
<tr>
<th>Level</th>
<th>Date</th>
<th>Section</th>
<th>Title</th>
<th>Author</th>
</tr>
<xsl:for-each select="article">
<tr>
<td><xsl:value-of select="@
level"/></td>
<td><xsl:value-of select="
@date"/></td>
<td><xsl:value-of select="@
section"/></td>
<td><xsl:value-of select="title"
/></td>
<td><xsl:value-of select="author"
/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

The output of an XSLT is set to HTML with the method="html" attribute of the xsl:output element. The output from the XSLT is an HTML document as illustrated in Figure 1.

With an XSLT transformer that supports

Level	Date	Section	Title	Author
Intermediate	January-2004	Java Technology	Service Oriented Architecture Frameworks	Naveen Balani
Advanced	October-2003	Java Technology	Advance DAO Programming	Sean Sullivan
Advanced	May-2002	Java Technology	Best Practices in EJB Exception Handling	Srikanth Shenoy

Figure 1 • Catalog HTML

XHTML output, the output from an XSLT transformation can be set to XHTML instead of HTML by setting the xsl:output element attribute method="xml". For XHTML output, set the default namespace declaration for the XHTML namespace in the xsl:stylesheet element. The default namespace declaration is set as:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/
Transform"
xmlns="http://www.w3.org/1999/xhtml">
```

For output to an XHTML document set the doctype-public and doctype-system attributes of the xsl:output element as illustrated:

```
<xsl:output method="xml" doctype- system
="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd"
doctype-public="-//W3C//DTD XHTML 1.0
Transitional//EN" />
```

Merging Documents

The document() function is used to refer to another XML document in an XML document.

With the document() function XML documents can be combined as illustrated in Listing 8.

Listing 8 Merging XML Documents

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:output method="xml" />
<xsl:template match="/">
<catalogs>
<xsl:copy-of select="*" />
<xsl:copy-of select="document('catalog2.
xml')"/>
</catalogs>
</xsl:template>
</xsl:stylesheet>
```

The XSLT combines the example XML doc-

ument catalog.xml and another XML docu- ment catalog2.xml listed in Listing 9.

Listing 9 catalog2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog
xmlns="http://www.w3.org/2001/XMLSchema-
Instance">
<journal title="Java Technology"
publisher="IBM developerWorks">
<article level="Intermediate"
date="February-2003">
<title>Design XML Schemas Using UML
</title>
<author>Ayesha Malik</author>
</article>
</journal>
</catalog>
```

The output from the XSLT is a combined XML document as shown in Listing 10.

Listing 10 Combined XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogs><catalog>
<journal title="Java Technology"
publisher="IBM developerWorks">
<article level="Intermediate"
date="January-2004" section="Java
Technology">
<title>Service Oriented Architecture
Frameworks</title>
<author>Naveen Balani</
author>
</article>
<article level="Advanced" date="
October-2003" section="Java
Technology">
<title>Advance DAO Programming</title>
<author>Sean Sullivan</author>
</article>
<article level="
Advanced" date="May-2002"
section="Java Technology">
<title>Best Practices in EJB
Exception Handling</title>
<author>Srikanth Shenoy </author>
</article>
```

```

</journal>
</catalog>
<catalog xmlns="http://www.w3.org/2001/
XMLSchema-Instance">
  <journal title="Java Technology"
    publisher="IBM developerWorks">

    <article level="Intermediate"
      date="February-2003">
      <title>Design XML Schemas Using UML
      </title>
      <author>Ayesha Malik</author>
    </article>
  </journal>
</catalog></catalogs>

```

Obtaining Element/Attribute Values with XPath

XSLT supports XPath, which is used to select elements and attributes. For example, select the value of the “date” attribute for the article with the title Advance DAO Programming, and select the value of the “title” element for the article by the

author Srikanth Shenoy. The XSLT illustrated in Listing 11 outputs the value of the “date” attribute and the “title” element.

Listing 11 XPath Expressions

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:
xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" omit-xml-
declaration="yes"/>
<xsl:template match="/catalog/journal">
Date: <xsl:value-of select="article[title='
Advance DAO Programming']/@date"/>
Title: <xsl:value-of select="article[author='S
rikanth Shenoy']/title"/>
</xsl:template>
</xsl:stylesheet>

```

The XPath expression ‘article[title=’Advance DAO Programming’]/@date’ selects the “date” attribute. The XPath expression ‘article[author=’Srikanth Shenoy’]/title’ selects the “title” element.

The output from the XSLT is shown in Listing 12.

Listing 12 XPath Expressions Output

Date: October-2003

Title: Best Practices in EJB Exception Handling

Filtering Elements

The elements in an XML document can be filtered by applying the xsl:apply-templates elements. For example, select the elements with the “level” attribute specified as “Intermediate.” The XSLT in Listing 13 selects the “article” elements that has the “level” attribute value “Intermediate.”

Listing 13 Selecting Elements

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:
xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" omit-xml-
declaration="yes"/>
<xsl:template match="/catalog/journal">
<xsl:apply-templates select="article[@level=
'Intermediate']"/>

```

FREE* CD! (\$198.00 VALUE!)

Secrets of the XML Masters

Every XML-J Article on One CD!

— The Complete Works —

CD is organized into 33 chapters containing more than 400 exclusive XML-J articles!

All in an easy-to-navigate HTML format! BONUS: Full source code included!

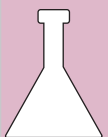
ORDER AT WWW.SYS-CON.COM/FREECD

*PLUS \$9.95 SHIPPING AND PROCESSING (U.S. ONLY)


©COPYRIGHT 2004 SYS-CON MEDIA. WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE. ALL BRAND AND PRODUCT NAMES ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES.

SYS-CON
EVENTS

Only from the World's Leading Technology Publisher



XSLT

```
</xsl:template>
<xsl:template match="article">
Title: <xsl:value-of select="title"/>
Author: <xsl:value-of select="author"/>
</xsl:template>
</xsl:stylesheet>
```

The XPath expression ‘article[@level=’Intermediate’]’ selects the “article” elements with “level” attributes set to “Intermediate.” The XSLT output contains only the “article” element with “level” value as “Intermediate.”

Listing 14 Selected Element

```
Title: Service Oriented Architecture
Frameworks
Author: Naveen Balani
```

Copying Nodes

The xsl:copy-of element copies the elements and attributes of the selected node. The XSLT in Listing 15 copies the “journal” node in the catalog.xml document to the output document.

Listing 15 Copying Nodes

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:
xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:output method="xml"/>
<xsl:template match="/catalog">
<xsl:copy-of select="journal"/>
</xsl:template>
</xsl:stylesheet>
```

The output from the XSLT consists of the journal node from the input XML document.

Listing 16 Copied Node

```
<?xml version="1.0" encoding="UTF-8"?>
<journal title="Java Technology"
publisher="IBM developerWorks">
<article level="Intermediate"
date="January-2004" section="Java
Technology">
<title>Service Oriented Architecture
Frameworks</title>
author>Naveen Balani</author>
</article>

<article level="Advanced" date="
October-2003" section="Java
Technology">
<title>Advance DAO Programming</
```

```
title>
<author>Sean Sullivan</author>
</article> <article
level="Advanced" date="May-2002"
section="Java Technology">
<title>Best Practices in EJB
Exception Handling</title>
<author>Srikanth Shenoy</author>
</article>
</journal>
```

xsl:copy, a different version of the xsl:copy-of element, doesn’t copy the sub-elements and attributes of the selected node.

Creating Elements and Attributes

The xsl:element element is used to create an element in an XML document. The xsl:attribute element is used to create an attribute in an XML document. The XSLT in Listing 17 creates the element “journal” and adds the attribute “publisher” to the “journal” element.

Listing 17 Creating an Element and an Attribute

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:
xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:output method="xml" omit-xml-
declaration="yes"/>
<xsl:template match="/">
<xsl:element name="journal">
<xsl:attribute name="publisher"><xsl:
text>IBM developerWorks</xsl:text></xsl:
attribute>
</xsl:element>
</xsl:template>
</xsl:stylesheet>
```

The output from the XSLT consists of the “journal” element with the “publisher” attribute.

Listing 18 Element/Attribute Created

```
<journal publisher="IBM developerWorks"/>
```

Outputting XSLT

XSLT output can be formatted with the xsl:output element. The encoding, the DOCTYPE declaration, and indentation can be set in the xsl:output element. The XSLT output in the section above can be formatted by setting the “encoding,” “doctype-

public” and “doctype-system” attributes of the xsl:output element.

The “encoding” attribute sets the encoding of the output XML document, the “doctype-public” and “doctype-system” the DOCTYPE of the document, and the “omit-xml-declaration” attribute the inclusion of the XML declaration.

Listing 19 xsl:output

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:
xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:output method="xml" encoding="UTF-
8" omit-xml-declaration="no" doctype-
public = "-//Sun Microsystems, Inc.//
DTD Enterprise JavaBeans 2.0//EN"
doctype-system ="http://java.sun.com/
dtd/ejb-jar_2_0.dtd" indent="yes"/>
<xsl:template match="/">
<xsl:element name="journal">
<xsl:attribute name="publisher"><xsl:
text>IBM developerWorks</xsl:text></xsl:
attribute>
</xsl:element>
</xsl:template>
</xsl:stylesheet>
```

The XML document output in the example XSLT consists of a DOCTYPE entity and the xml declaration, whose encoding attribute is set to ‘utf-8’.

Listing 20 Output with Doctype Declaration

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE journal PUBLIC "-//Sun
Microsystems, Inc.//DTD Enterprise
JavaBeans 2.0//EN" "http://java.sun.com/
dtd/ejb-jar_2_0.dtd">
<journal publisher="IBM developerWorks"/>
```

Conclusion

An XML document can be transformed to another xml/text/html document by applying suitable XSLT transformations. In this tutorial the XSLTs required for transforming an XML-to-XML or -text or -HTML are discussed. 🌀

dvohra09@yahoo.com

ajay_vohra@yahoo.com

Reach Over 100,000

Enterprise Development Managers & Decision Makers with...



Offering leading software, services, and hardware vendors an opportunity to speak to over 100,000 purchasing decision makers about their products, the enterprise IT marketplace, and emerging trends critical to developers, programmers, and IT management

Don't Miss Your Opportunity to Be a Part of the Next Issue!

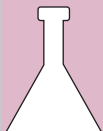
Get Listed as a Top 20^{*} Solutions Provider

For Advertising Details Call 201 802-3021 Today!

*ONLY 20 ADVERTISERS WILL BE DISPLAYED. FIRST COME FIRST SERVE.



The World's Leading i-Technology Publisher



WRITTEN BY JIMMY ZHANG

Process SOAP with VTD-XML

Discover the benefits

SOAP is an XML based data protocol standardized by W3C for the purpose of enabling inter-application data exchange over the Internet. In a typical Web Services scenario, a SOAP message delivered via HTTP needs to be parsed before anything else can happen. As two popular SOAP processing methods, DOM and SAX/Pull force application developers to choose between performance/memory efficiency and ease of use. VTD-XML is the latest open-source, “non-extractive” XML processing API written in Java that overcomes many problems and issues of the status quo. The combination of its high performance, low memory footprint, random access, incremental update, and inherent persistence simply means this: With VTD-XML, application developers can finally unleash to the fullest extent the power of SOAP.

AUTHOR BIO

Jimmy Zhang is a cofounder of XimpleWare, a provider of high performance XML processing solutions. He has working experience in the fields of electronic design automation and Voice over IP for a number of Silicon Valley high tech companies. He graduated from U.C. Berkeley with both an MS and a BS from the department of EECS.

To many application developers, Web Services are usually synonymous with SOAP over HTTP. While HTTP (hyper-text transfer protocol) has been around for over a decade, the real excitement of Web Services lies in the use of SOAP (Simple Object Access Protocol). Effectively a subset of XML, SOAP possesses some unique attributes that set Web Services apart from previous distributed computing technologies, such as DCOM and CORBA. For one, SOAP is open and human readable, meaning that programming SOAP is simpler and easier to grasp. And equally important is the fact that SOAP repre-

sentation of data is loosely encoded. Applications communicating using SOAP are no longer restrained by the rigidity of schema, making possible the true decoupling between application logic and wire format of data.

Current SOAP Processing Overview

Due to its textual nature, a SOAP message must be parsed into machine-readable form before it can be understood by software applications. There are two types of SOAP processing models widely in use today:

- DOM (Document Object Model) is a tree-based XML processing API specification. Because DOM creates in-memory data structure precisely modeling data represented in XML and allows random-access, it is generally considered an easy and natural way of working with XML. But building a DOM tree consumes 5x~10x the memory of the XML itself, and incurs a non-trivial amount of processing cost, making DOM ill-suited for most high performance XML applications.
- SAX/PULL are specifically created to tackle the memory and processing inefficiency of DOM, as both export low-level tokenizer interfaces and, by default, never keep the entire document in memory. As a result, SAX/Pull based XML processing incurs less memory overhead and can potentially process very large XML files. Unfortunately they are also more difficult to use than DOM

for precisely the same reason. Unless users build their own custom object model, SAX/Pull don't offer random access, and force users to scan the XML document multiple times, making performance improvements over DOM insignificant. What's more, SAX/Pull programming interweaves application logic with XML processing, resulting in awkward, bulky application code that is hard to maintain.

So in a way, with current XML processing methods, it is difficult to get both high processing/memory efficiency and ease of use. But there is more to think about.

Right now, parsing SOAP messages, whether the application uses DOM or SAX, is pretty much inevitable, even if it is done repetitively to the same messages. Would it be nice if there is a pre-parsed form of XML directly reusable without the overhead of parsing every time?

Also consider modifying the text content of the following XML file.

```
<color> red </color>
```

Using DOM, it would require at least the following three steps: build the DOM tree, navigate to and then update the text node, write the updated structure back into XML. So no matter how trivial the modification is, there is a round trip penalty of parsing the document and writing it back out. What if it is only a snippet buried within a big document? Would it be nice to be able to surgically remove then insert the update “in-place?”

VTD: A Simple Solution

Historically, the first step of text processing is usually to tokenize the input file into many little null-terminated strings. But there is another way to tokenize. Rather than extracting the token content out of the input, one can instead retain the original document intact in memory and use the offsets and lengths to describe tokens. In other words, tokenization can be done “non-extractively.” We can look at how this “non-extractive” tokenization approach works in practice and compare it with traditional “extractive” view of tokens in the context of some common usage scenarios.

1. **String comparison**- Under the traditional text-processing framework, C’s “strcmp” function (in <string.h>) compares an “extractive” token against a known string. In our new “non-extractive” approach, one can simply use C’s “strncmp” function in <string.h>.
2. **String to numerical data conversion**- C’s “atoi” and “atof” convert strings into numerical data types. One can introduce new functions or macros to convert “non-extractive” tokens into integers or floats. For example, the new “atof_ne” would have to take three inputs: the character pointer, the starting offset, and the length. Notice that the character pointer points at the memory buffer in which the entire document resides.
3. **Trim**- To remove leading and trailing white spaces of “non-extractive” tokens, we only need to recompute the offsets and lengths based on their older values. To do the same thing to extractive tokens usually involves creating new tokens.

How to store offsets and lengths is the next question to think about. The handy way is to store them as member variables of objects. In a way objects are nothing more than small memory blocks filled with bits also known as member variables. But in the strictest sense small memory blocks filled with bits aren’t necessarily objects. Consider a MIPS instruction that uses 32 bits to encode both op-code and operands. Also several segment registers in X86 architecture encode many parameters in 64 bits.

Above considerations have led to the design of a “non-extractive” token encoding specification called Virtual Token Descriptor (VTD). AVTD record is a 64-bit integer that encodes the length, the starting offset, the token type and nesting depth of a token in XML. For certain types of tokens, the length field further encodes the prefix length and qualified name length, since both share the identical offset.

One immediate benefit of VTD’s non-extractive tokenization is that, because the document is kept

intact, VTD allows applications to surgically insert and remove XML content similar to manipulating a byte array. For example, removing or changing the value of an attribute value or text content is the same as skipping the segment marked by the offset and length containing “unwanted” text. Also VTD makes possible the removal of entire element by simply skipping it according to its offset and length.

Introduce VTD-XML

Built on the concept of VTD, VTD-XML is the latest open source, “non-extractive,” Java-based XML processing API (VTD-XML) ideally suited for SOAP processing. Currently it supports only five built-in entities (& < > ' "). The latest VTD-XML is version 0.8, which can be download here (<http://vtd-xml.sf.net>). Aside from maintaining the XML file intact in memory and exclusively using VTD to describe tokens, VTD-XML also introduces the concept of location caches that provide efficient random access. Different from DOM, VTD-XML’s notion of hierarchy consists exclusively of elements, which essentially correspond to VTD records for starting tags. Resembling the index section of a book, location caches again make extensive use of 64-bit integers. The project web site (<http://vtd-xml.sf.net>) has an in-depth description on how VTD-XML achieves the purpose of random access with location caches.

VTD-XML should exhibit the following characteristics when used in a Web Services project. First, it parses SOAP messages at the performance level equivalent, if not faster, than SAX with the NULL content handler. On a 1.5 GHz Athlon processor, VTD-XML processes SOAP message at around 25~35 MB/sec. Second, unlike SAX, VTD-XML offers full random access as the entire parsed XML is resident in memory. Furthermore, if you are one of the developers that finds DOM’s node-based API verbose and difficult to use, you should find VTD-XML’s API clean and easy to comprehend. And VTD-XML’s memory requirement is about 1.3x to 1.5x the size of XML, with 1 being the document itself as it is part of the internal representation of VTD-XML. Plus, incremental, dynamic update to the XML content is much more efficient than either DOM or SAX.

Why does VTD-XML consume less memory than DOM? In many VM-based object-oriented programming languages, per-object allocation incurs a small amount of memory overhead. VTD records are immune to the overhead because they are not objects. Also VTD records are constant in length and can be stored in large memory blocks, which are more efficient to allocate and garbage

collect. For example, by allocating a large array for 4096 VTD records, one incurs the per-array overhead (16 bytes in JDK 1.4) only once across 4096 records, thus reducing per-record overhead to very little.

And more importantly, VTD’s efficient memory usage has strong implication on its performance. DOM is slow in a very large part because it is resource intensive. The spirit of VTD is this: one simply doesn’t have to, and has every incentive not to, create strings objects because they are slow to create. Even worse, they eventually need to be garbage collected. VTD-XML is able to achieve SAX’s performance level because VTD significantly reduces DOM’s memory usage, thus leading to savings on both object creation and garbage collection.

At the top level, VTD-XML provides three essential classes: VTDGen, VTDNav, and AutoPilot.

- VTDGen parses the XML/SOAP messages into VTD records and location caches.
- VTDNav is a cursor-based API allowing for DOM-like random access of the XML structure.
- AutoPilot works with VTDNav and emulates the behavior of DOM’s node iterator.

The rest of this article demonstrates how to use VTD-XML to process a sample SOAP message.

A Sample Project

To process SOAP with VTD-XML, the starting point is a memory buffer filled with the content of XML/SOAP message. The sample message containing the purchase order (shown below) in the body section of the SOAP envelope. For simplicity reasons, the project assumes the message resides on disk. In real life, one is more likely to read the message off HTTP. (See Listing 1.)

At the top level, this project has a single main method (shown below) that wraps all code with a single try catch block that takes care of various exception conditions for IO operation, parsing and navigation. (See Listing 2.)

The following code parses the SOAP message. It first allocates a byte array, and reads into it the byte content of the SOAP message. Then, it instantiates VTDGen and passes to it the byte array. Next, it calls VTDGen’s member method “parse()” to generate the internal, parsed representation of the SOAP message. Notice that “parse()” accepts the Boolean value of “true” to indicate the parsing is namespace-aware. (See Figure 3.)

After parsing, the sample code obtains an instance of VTDNav and uses the namespace aware “toElementNS()” to move the cursor to various

positions of the element hierarchy and prints out corresponding text values, or selectively pulls out the XML fragment at the cursor position. (See Listing 4)

The code above concerning VTDNav has several points worth mentioning.

1. There is one and only one cursor available, which can be moved using "toElement()" or "toElementNS()". Those methods return a boolean indicating the status of the movement. If true, the cursor is repositioned; otherwise, no movement on the cursor.
2. Several member methods, such as "getAttribute()" and "getText()", return an integer corresponding to the index value of the VTD record if there is one. -1 is returned if no such record is found.
3. VTDNav performs string to VTD record comparison directly, avoiding the round trip of creating and de-allocate string object.

4. VTDNav also performs VTD record to numerical data type directly for the same purpose.
5. There is a global stack available so one can save, then quickly store the the saved cursor location.
6. VTDNav also allows one to convert a VTD record into a string object. Use this carefully for reasons in 3.

The final part of the project composes an invoice for the purchase order (shown below with changes in bold). The invoice looks quite similar to the PO so VTD-XML allows cutting and pasting of XML. (See Listing 5.)

The code that composes the invoice is shown in Listing 6.

The Road Map and a Quick Recap

The other property of VTD-XML is that its internal representation is inherent persistent,

making it possible to avoid parsing for repetitive read-only XML processing. This also makes possible an XML upgrade path that improves XML processing performance without losing human readability.

As readers can see, VTD-XML, the new, non-extractive, Java-based XML processing API based on VTD, offers a number of benefits not found with existing XML processing APIs. The most significant one is that it simultaneously offers high performance, low memory usage, user-friendliness. Also it introduces the notion of incremental update. As XML makes inroads into IT and becomes increasingly indispensable in our lives, VTD-XML should find its way in more places and hopefully enable new exciting XML applications. 🌐

jzhang@ximpleware.com

Listing 1

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
soap/envelope/">
  <SOAP-ENV:Body>
    <po xmlns="http://www.skatestown.com/ns/po"
      id="50383" submitted="2001-12-06">
      <billTo>
        <company>The Skateboard Warehouse</company>
        <street>One Warehouse Park</street>
        <street>Building 17</street>
        <city>Boston</city>
        <state>MA</state>
        <postalCode>01775</postalCode>
      </billTo>
      <shipTo>
        <company>The Skateboard Warehouse</company>
        <street>One Warehouse Park</street>
        <street>Building 17</street>
        <city>Boston</city>
        <state>MA</state>
        <postalCode>01775</postalCode>
      </shipTo>
      <order>
        <item sku="318-BP" quantity="5">
          <description>Skateboard backpack; five
            pockets</description>
        </item>
        <item sku="947-TI" quantity="12">
          <description>Street-style titanium
            skateboard.</description>
        </item>
        <item sku="008-PR" quantity="1000"/>
      </order>
    </po>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 2

```
import com.ximpleware.*;
import java.io.*;

class project{

  public void static main(String[] args){
    try{
```

```
// application stay here
}
catch(IOException e){
  System.out.println(" IO exception
    condition"+e);
}
catch(ParseException e){
  System.out.println(" XML file parsing error
    \n"+e);
}
catch(NavException e){
  System.out.println(
    " Exception during navigation "+e);
}
}
```

Listing 3

```
// parse SOAP message named po.xml from the diskFile
f = new File("po.xml");
FileInputStream fis = new FileInputStream(f);

// find out file length and allocate buffer
byte[] b = new byte[(int) f.length()];

// read file content into the buffer
fis.read(b);

//create an instance of VTDGen
VTDGen vg = new VTDGen();

// assign the buffer to VTDGen
vg.setDoc(b);

// parse with namespace awareness turned off
vg.parse(true);
```

Listing 4

```
// get VTDNav
VTDNav vn = vg.getNav();

// the cursor initially at root
if (vn.toElementNS(VTDNav.FIRST_CHILD,
  "http://schemas.xmlsoap.org/soap/envelope/",
  "Body"))
{
  if (vn.toElementNS(VTDNav.FIRST_CHILD,
```

```

        "http://www.skatestown.com/ns/po",
        "po"))
    {
        // print of the attribute value of "id"
        int id = vn.getAttrVal("id");
        if (id != -1)
            System.out.println("id value: "+vn.
toString(id));
        // print out the element element fragment for
        billTo
        // getElementFragment returns a long val
        // upper 32 bits are length
        // lower 32 bits are offset
        if (vn.toElement(VTDNav.FIRST_CHILD, "billTo"))
        {
            long l = vn.getElementFragment();
            int len = (int) (l>>32);
            int offset = (int)l;
            System.out.print("element fragment for billTo
==> \n");
            System.out.println(new String(
b,offset, len));
        }
        // print out the element element fragment for
        shipTo
        if (vn.toElement(VTDNav.NEXT_SIBLING, "shipTo"))
        {
            long l = vn.getElementFragment();
            int len = (int) (l>>32);
            int offset = (int)l;
            System.out.print("element fragment for shipTo
==> \n");
            System.out.println(new String(
b,offset, len));
        }
        // navigate across all elements of order
        // print out various attributes and text content
        // using getAttrVal, getText, toString
        // those functions return -1 if no value should
        be returned,
        // corresponding to null in DOM.
        if (vn.toElement(VTDNav.NEXT_SIBLING, "order"))
        {
            if (vn.toElement(VTDNav.FIRST_CHILD, "item")){
                do {
                    int temp = vn.getAttrVal("sku");

                    temp = vn.getAttrVal("quantity");
                    if (temp != -1)
                        System.out.println("quantity ==> "
+vn.toString(temp));
                    if (vn.toElement(VTDNav.FIRST_CHILD,
"description"))
                    {
                        temp = vn.getText();
                        if (temp != -1)
                            System.out.println(" text for
description ==> "
+vn.toString(temp));
                        vn.toElement(VTDNav.PARENT);
                    }
                } while (vn.toElement(VTDNav.NEXT_
SIBLING, "item"));
            }
        }
    }
}

```

Listing 5

```

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/">
<SOAP-ENV:Body>
<invoice xmlns="http://www.skatestown.com/ns/po"
id="50383" submitted="2001-12-06">
<billTo>
<company>The Skateboard Warehouse</company>
<street>One Warehouse Park</street>
<street>Building 17</street>

```

```

<city>Boston</city>
<state>MA</state>
<postalCode>01775</postalCode>
</billTo>
<shipTo>
<company>The Skateboard Warehouse</company>
<street>One Warehouse Park</street>
<street>Building 17</street>
<city>Boston</city>
<state>MA</state>
<postalCode>01775</postalCode>
</shipTo>
<order>
<item sku="318-BP" quantity="5" price="30.00">
<description>Skateboard backpack; five
pockets</description>
</item>
<item sku="947-TI" quantity="12" price="40.00">
<description>Street-style titanium
skateboard.</description>
</item>
<item sku="008-PR" quantity="1000" price="5.00"/>
</order>
<status> shipped </status>
<total> 5630.00 </total>
</invoice>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Listing 6

```

// compose invoice
// first set the cursor to root
// then navigates to po and get the fragment offset and
length using getElementFragment
// output writes to invoice.xml;
FileOutputStream fos = new FileOutputStream
(new File("invoice.xml"));
fos.write(
"<SOAP-ENV:Envelope xmlns:SOAP-ENV=\"http://schemas.xmlsoap.
org/soap/envelope/\">\n<SOAP-ENV:Body><invoice\".getBytes());
vn.toElement(VTDNav.ROOT);
vn.toElementNS(VTDNav.FIRST_CHILD,
"http://schemas.xmlsoap.org/soap/envelope/",
"Body");
vn.toElementNS(VTDNav.FIRST_CHILD,
"http://www.skatestown.com/ns/po",
"po");
long l = vn.getElementFragment(); // get the offset and
length of the fragment
int offset = (int) l + 3;
int len = (int) (l>>32);

vn.toElement(VTDNav.FIRST_CHILD, "order");
long l2 = vn.getElementFragment();
int offset2 = (int) l2;
int len2 = (int) (l2>>32);

vn.toElement(VTDNav.FIRST_CHILD, "item");
do{
    int temp = vn.getAttrVal("quantity");
    int ot = vn.getTokenOffset(temp)+vn.getTokenLength(temp)+1;

    fos.write(b, offset, ot-offset);
    if (vn.matchTokenString(vn.getAttrVal("sku"), "318-BP"))
        fos.write(" price=\"30.00\"".getBytes());
    if (vn.matchTokenString(vn.getAttrVal("sku"), "947-TI"))
        fos.write(" price=\"40.00\"".getBytes());
    if (vn.matchTokenString(vn.getAttrVal("sku"), "008-PR"))
        fos.write(" price=\"5.00\"".getBytes());
    offset = ot;
}while(vn.toElement(VTDNav.NEXT_SIBLING, "item"));


fos.write(b, offset, offset2+len2-offset);

fos.write("<status>shipped</status>\n<total>5630.00</total>\n"
.getBytes());
fos.write("</invoice>\n</SOAP-ENV:Body>\n</SOAP-ENV:
Envelope>".getBytes());

```

Mindreef Launches Collaborative Online Web Services Diagnostics, Updates SOAPscope

(Hollis, NH) – Mindreef, Inc., a provider of Web services diagnostics solutions, has

 announced the release of Mindreef Share-It, a new collaborative online Web services diagnostics solution that allows Web services professionals to publish and share evidence related to a Web services problem online via the Web site www.mindreef.net. Mindreef enables this new service with version 4.1 of Mindreef SOAPscope, the company's award-winning Web services diagnostics application and a primary component of the Mindreef System.

Using Mindreef SOAPscope 4.1, a Web services professional can capture complex problem data and its context and then publish that data to Mindreef.net where it can be shared with team members, partners, customers, and others.

Along with enabling Share-It, Mindreef SOAPscope 4.1 introduces a number of new features including SSL message capture, which allows for SOAP message capture over HTTPS, and support for proxy auto-config (PAC) scripts. www.mindreef.com

Eclipse Foundation Makes Available Web Services Tools

(Burlingame, CA) – The Eclipse Foundation has announced the first developer release of the Eclipse Web services tools. As part of the Eclipse Web Tools Platform (WTP) Project, the new Web services tools aim to help shorten development time, simplify the development of Java Web services, and automatically validate for conformance to industry standards. WTP 1.0 release is expected to be available in July 2005.

With this new milestone release, the WTP Project has added support for Web services standards from the World Wide Web Consortium (W3C) and the Web Services Interoperability (WS-I) organization and provides the reference implementation of the WS-I validation tools.

The new open source Web services tools offering includes authoring tools for the Web Service Description Language (WSDL), XML and XML schema standards, and wizards that simplify the Web service creation process. Used with the Web services validation tools, including the WS-I test tools, this enables developers to build, test and deploy Web services.

www.eclipse.org

Services Companies Switch to NetSuite From Salesforce.com

(San Mateo, CA) – NetSuite, Inc. has announced that a number of services-based companies have switched from salesforce.com to NetSuite. Some of the companies that have switched include DSLIndiana.com, Applicant Insight, GuildQuality and AD Systems. NetSuite offers a migration program for companies that want to standardize on NetSuite from salesforce.com.

These announcements were made in conjunction with today's introduction of NetCRM-Services Edition. Designed to meet the specific needs of services-based businesses, NetCRM-Services Edition offers features that cannot be found in any other stand-alone, on-demand CRM applications. These service-specific features consist of service item management, project/job tracking, client self-service center, advanced activity and time tracking, and document management in addition to traditional CRM functionality including marketing campaign management, client support and opportunity management. Additionally, with NetCRM's advanced customization capabilities, NetCRM-Services Edition gives services companies the flexibility and power to cater the feature-rich application to meet their exact needs. Finally, when deployed as part of the complete NetSuite ERP/CRM solution, services-based businesses can go from proposal to invoice in a single integrated solution.

NetSuite enables companies to manage all key business operations in a single, integrated system, which includes customer relationship management; order management and fulfillment; inventory management; finance; e-commerce and Web site management; and employee productivity. NetSuite is delivered as an on-demand service, so there is no hardware to procure, no large, up-front license fee, and no complex set-ups. Finally, NetSuite's patent-pending "real-time dash board" technology provides an easy-to-use view into role-specific business information that is always up-to-date.

www.netsuite.com

Versata Announces Versata 6 – A Comprehensive Business Rules Platform for Agile Application Development

(Oakland, CA) – Versata, Inc., an independent provider of rapid application development platforms to automate and manage complex, data-intensive business applications, has announced Versata 6, an integrated and comprehensive Business Rules Engine (BRE). Versata 6 significantly extends the capabilities of the Versata platform, and is

one of the first BREs to include wide-ranging business rules for decision support, process, transaction, data logic, Web services, and service-oriented architectures (SOA). Versata 6 provides customers with the power and flexibility to use an inclusive BRE for their applications needs, and to deliver IT systems that can be quickly delivered and modified to suit changing business conditions and regulatory requirements.

Versata 6 uses an open architecture and supports SOA and Web Services. Built to J2EE architectural standards, Versata 6 uses a model-driven approach using declarative programming and business rules to create Java applications with built-in support for SOA and Web Services. Applications developed with Versata 6 can expose transactions and services as standard Web services, or as stateless session beans, and can optionally use the built-in Service Data Objects (JSR235) support. This broad support for SOA continues Versata's tradition of creating high-performance, scalable, N-tier J2EE applications.

www.versata.com

Forum Systems and Caymas Systems Sign Technology Agreement Simplifying Web Services Security

(Salt Lake City, UT) – Forum Systems, Inc., a leading provider of Trust Management and Threat Protection Web Services Security solutions, has announced it has joined forces with Caymas Systems, the leader in Identity-Driven Access, to bring customers scalable and cost-effective access control and security for business applications that rely on Web services. The technology agreement is centered on integrating the Caymas 220, Caymas 318 and Caymas 525 Identity-Driven Access Gateways with the Forum XWall Web Services Firewall.

The market for optimized and dedicated Web services security solutions has grown exponentially as companies realize the need to prevent unauthorized access to external Web services as well as prevent against XML-related threats.

This partnership is continued recognition of the need to combine high-performance XML Firewall technology with identity-driven Web access. In the same manner that users are authenticated prior to granting admission to the corporate network, automated machine-to-machine interactions must also be authenticated and access restricted based on the identity of users, devices and resources involved, and based on the content of the information being exchanged.

www.forumsys.com, www.caymas.com

WebRenderer™



Standards compliant embeddable web browser component

WebRenderer is a cutting edge embeddable Java™ web content rendering component that provides Java applications and applets with a fast, standards compliant HTML and multimedia rendering engine. WebRenderer provides a feature-packed API including complete browser control, a full array of events, JavaScript interface, DOM access, document history and more.

Why WebRenderer?

- Standards Support (HTML 4.01, CSS 1 & 2, SSL, JavaScript, XSL, XSLT etc.)
- Exceptionally Fast Rendering
- Predictable Rendering
- Scalability (deploy in Applications or Applets)
- Security (based on industry standard components)
- Stability and Robustness

Embed WebRenderer to provide your Java™ application with standards compliant web content rendering support.

To download a 30 day trial of WebRenderer visit
www.webrenderer.com

**JadeLiquid™
Software**

Copyright JadeLiquid Software 2004. JadeLiquid and WebRenderer are trademarks of JadeLiquid Software in the United States and other countries. Sun, Sun Microsystems, the Sun Logo and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other trademarks and product names are property of their respective owners.



HostMySite.com

Built for ColdFusion Pros

by ColdFusion Pros

plans from

\$8.95 / mo.

FREE Domain Name*

FREE Setup

FREE 2 Months

- 24 / 7 / 365 Phone Support
- 99.9+% Uptime
- Macromedia Alliance Partner
- "Full Control" Panel
- CFMX 6.1 or CF 5.0
- SQL Server 2000 or 7.0
- Custom Tags Welcome

Visit www.HostMySite.com/mxdj for:

2 Months Free

and **FREE Setup on Any Hosting Plan***



**"When it comes to ColdFusion hosting,
HostMySite.com rules them all!"**

James Kennedy
mbateam.com

*offer applies to any annual shared hosting plan

call
today!

877•248•HOST
(4678)

