

# BEA WebLogic

DEVELOPER'S JOURNAL

MARCH 2004 - Volume:3 Issue:3

weblogicdevelopersjournal.com

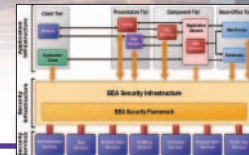
**SAVE UP TO \$400**

**Subscribe today and get up to 3 FREE CDs!**  
SEE PAGES 44 & 45 FOR DETAILS

# WEBLOGIC ENTERPRISE SECURITY

Paul Patrick 6

INTEGRATION: **ebXML and XML Digital Signature** WebLogic Integration takes advantage of a growing technology



12

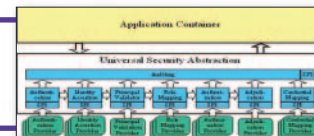
Sanjay Dalal

TRANSACTIONS: **Tricky Transactions** Understanding implicit transactions in WebLogic Integration

20

John Graves

SUPPORT: **Handling System Core Files** Before you debug



26

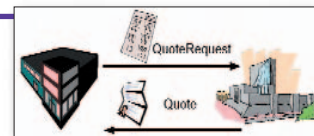
Steve Pozarycki

FROM THE OFFICE OF THE CTO: **Web Services Security Progress Report** Moving forward - and on schedule

30

Hal Lockhart

CASE STUDY: **A Safe Architecture Framework** Leverage the security features of BEA WebLogic 8.1



34

Ashley Byrd & Girish Gupte

ARCHITECT'S ANNEX: **Modernizing Legacy Systems, PART 2** Integration as the critical step



38

Anwar Ludin

JAVA: **Analyzing Java Application Problems** Using Java thread dumps to assess the dilemma

46

Sathish Santhanam

FROM THE EDITOR

Watch Your Security Hole  
by Joe Mitchko  
page 4

TRANSACTION MANAGEMENT

How Loose is Your Coupling?  
by Peter Holditch  
page 42

NEWS  
page 50

RETAILERS PLEASE DISPLAY UNTIL APRIL 30, 2004

\$8.99US \$9.99CAN



# BEA WEBLOGIC WORKSHOP 8.1 makes you more

# AMAZING.



Development Tools  
December 30, 2003  
BEA WebLogic Workshop 8.1  
BEA Systems, Inc.

"A shared, consistent  
development platform...for  
J2EE experts and business-  
oriented developers alike."



Java Pro Readers  
Choice Award

"WebLogic Workshop  
empowers all application  
developers...not just J2EE  
developers."



Software Development Jolt  
Product Excellence Award

"The integration of an IDE,  
controls and a deployment  
environment...greatly enhances  
developer productivity."



CRN Test Center  
Product of the Year

"By far the most innovative  
development tool reviewed  
this year...when compared  
with other tools, Workshop  
blew away the competition."

But don't take their word for it, find out for yourself now, for free.

BEA's WebLogic Workshop 8.1 has won every major award for software development tools in the past year. It's a full-featured Java development environment that lets you visually build and assemble enterprise-scale Web Services, Web Applications, JSPs, Portals, EJBs, and Business Process models based on the latest standards and open source technologies.

Get it now. For free. Visit <http://dev2dev.bea.com/workshop2>

dev2dev™







# Optimize J2EE.

The J2EE revolution is here to increase performance, value, and lower IT costs. It's a solution from Mercury Interactive that makes your whole J2EE applications ecosystem work right.

It's the fastest way to find the toughest J2EE problems from the development to the live applications. Even pinpoints root cause and the line of source code.

It's about more than delivering J2EE applications. It's about delivering applications that work and yield real business value.

It's the Business Technology Optimization revolution.

And Mercury is the only one bringing you a revolutionary solution for J2EE.

Download our free white paper, "**Diagnosing J2EE Performance Problems Throughout the Application Lifecycle,**"

at [www.mercuryinteractive.com/optimizej2ee](http://www.mercuryinteractive.com/optimizej2ee)

Get Optimized.™





# Watch Your Security Hole

BY JOE MITCHKO

Anyone who has recently been on the job hunting circuit, looking for a position as a developer, knows that employers are getting rather picky. With the over-supply of IT professionals, recruiters are not just looking for good people, they are looking for good people with an exact skill set to match their requirements. As such, the chances of getting the position you desire is not as guaranteed as it was back in the boom times four years ago. Besides having good looks and luck, one of the ways that you can get around this dilemma is to maintain a diverse set of skills on your résumé and hope that some combination will get you the job.

One of the traditional "must-have" items on your résumé used to be working knowledge of relational database systems (and associated APIs) or application servers. Job requisitions would be combinations of Java/Oracle, WebLogic Server/J2EE/Sybase, etc. The trend now, especially in the new millennium where the Internet can sometimes appear as a vast wasteland swarming with viruses, worms, and other types of low life, is that security issues have taken center stage – both in and out of cyberspace.

For example, a leading IT consulting company has recently upgraded their methodology to include security-related issues and design. What makes this particularly interesting is that this methodology addresses security-related issues during all phases of the project life cycle, starting with the envisioning phase and on through to the operational services phase. In another case, the Microsoft Solutions Framework includes security in a number of different phases in their process model, starting early in the planning phase. As is evident in these examples, security issues are no longer an afterthought and need to be addressed throughout application and system development, and rightly so.

Now, what this means for the average Joe Developer is this: you should start thinking more about further developing your knowledge and skills related to IT security and make sure your résumé reflects whatever experience you do have. To start, you will need some working knowledge regarding basic security concepts, including encryption, authentication, and authorization. That is just the basics! When you move into the Web services space, there are other security-related specifications that you should be aware of, including WS-Security and SAML. Therefore, if you haven't paid much attention to your security-related skill sets, or if you think that it's only for the security specialist or guru, it's time to think again.

To help you begin to sort through all this stuff, this month we will focus on some of the security issues and solutions that you may come across as a WebLogic developer. You will also get up to speed on the new Enterprise Security Initiative, announced by BEA, which will introduce a number of security management services to the BEA WebLogic Server platform, including single sign-on and other advanced security features.

Finally, if you just happen to be in San Francisco during the May 24–27 time frame, come and visit the Ninth Annual BEA eWorld Technology Conference at the Moscone West Convention Center in town. The conference promises to provide you with the latest on what is happening at BEA, and will be jam-packed with new and exciting hands-on sessions and a variety of keynote speakers. You will also find an exhibit floor full of BEA partners and vendors with plenty of giveaways to help fill up your "techy" bag. For more information and to sign up for the event, check the BEA Web site at [www.bea.com](http://www.bea.com). I look forward to meeting all of you there!

**EDITORIAL ADVISORY BOARD**  
TOM ANGELUCCI, LEWIS CIRNE, KEVIN JONES,  
TYLER JEWELL, WAYNE LESLEY LUND, SEAN RHODY,  
CARL SJOGREEN

**FOUNDING EDITOR**

PETER ZADROZNY

**EDITOR-IN-CHIEF**

JOE MITCHKO

**PRODUCT REVIEW EDITOR**

JASON SNYDER

**EXECUTIVE EDITOR**

GAIL SCHULTZ

**EDITOR**

NANCY VALENTINE

**ASSOCIATE EDITORS**

JAMIE MATUSOW, JEAN CASSIDY

**ASSISTANT EDITOR**

JENNIFER VAN WINCKEL

**WRITERS IN THIS ISSUE**

ASHLEY BYRD, SANJAY DALAL, JOE MITCHKO,  
JOHN R. GRAVES, GIRISH GUPTA, PETER HOLDITCH,  
HAL LOCKHART, ANWAR LUDIN, JOE MITCHKO, PAUL  
PATRICK, STEVE POZARYCKI, SATHISH SANTHANAM

**SUBSCRIPTIONS**

For subscriptions and requests for bulk orders,  
please send your letters to Subscription Department.  
SUBSCRIPTION HOTLINE:

888-303-5282

Cover Price: \$8.99/Issue

Domestic: \$149/YR (12 Issues)

Canada/Mexico: \$169/YR

Overseas: \$179/YR

(U.S. Banks or Money Orders)

**PRESIDENT AND CEO**

FUAT KIRCAALI

**VP, BUSINESS DEVELOPMENT**

GRISHA DAVIDA

**GROUP PUBLISHER**

JEREMY GEELAN

**TECHNICAL DIRECTOR**

ALAN WILLIAMSON

**PRODUCTION CONSULTANT**

JIM MORGAN

**ART DIRECTOR**

ALEX BOTERO

**ASSOCIATE ART DIRECTORS**

LOUIS F. CUFFARI • RICHARD SILVERBERG

**ASSISTANT ART DIRECTOR**

TAMI BEATTY

**SENIOR VP, SALES & MARKETING**

CARMEN GONZALEZ

**VP, SALES & MARKETING**

MILES SILVERMAN

**ADVERTISING SALES DIRECTOR**

ROBYN FORMA

**DIRECTOR OF SALES AND MARKETING**

MEGAN MUSSA

**ADVERTISING SALES MANAGERS**

ALISA CATALANO • CARRIE GEBERT

**ASSOCIATE SALES MANAGERS**

KRISTIN KUHNLE • BETH JONES

**ADVERTISING INTERN**

ALYMA HAWTHORNE

**PRESIDENT, SYS-CON EVENTS**

GRISHA DAVIDA

**NATIONAL SALES MANAGER**

SEAN RAMAN

**FINANCIAL ANALYST**

JOAN IAROSE

**ACCOUNTS RECEIVABLE**

CHARLOTTE LOPEZ

**ACCOUNTS PAYABLE**

BETTY WHITE

**VP, INFORMATION SYSTEMS**

ROBERT DIAMOND

**WEB DESIGNERS**

STEPHEN KILMURRAY • CHRISTOPHER CROCE

**ONLINE EDITOR**

LIN GOETZ

**CIRCULATION SERVICE COORDINATORS**

SHELIA DICKERSON • EDNA EARLE RUSSELL

**JDJ STORE MANAGER**

BRUNILDA STARAPOLI

**EDITORIAL OFFICES**

SYS-CON Publications, Inc.

135 Chestnut Ridge Road, Montvale, NJ 07645

Telephone: 201 802-3000 Fax: 201 782-9638

SUBSCRIBE@SYS-CON.COM

BEA WebLogic Developer's Journal (ISSN# 1535-9581)

is published monthly (12 times a year)

Postmaster: Send Address Changes to

BEA WEBLOGIC DEVELOPER'S JOURNAL,

SYS-CON Publications, Inc.

135 Chestnut Ridge Road, Montvale, NJ 07645

**WORLDWIDE NEWSSTAND DISTRIBUTION**

Curtis Circulation Company, New Milford, NJ

**FOR LIST RENTAL INFORMATION:**

Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com

Frank Cipolla: 845 731-3832, frank.cipolla@eposdirect.com

**SYS-CON MEDIA**

© COPYRIGHT 2003 BY SYS-CON PUBLICATIONS, INC. ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM, WITHOUT WRITTEN PERMISSION. FOR PROMOTIONAL REPRINTS, CONTACT REPRINT COORDINATOR, SYS-CON PUBLICATIONS, INC. RESERVES THE RIGHT TO REVISE, RE-PUBLISH AND AUTHORIZE THE READER 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 WEBLOGIC DEVELOPER'S JOURNAL.

**AUTHOR BIO...**  
Joe Mitchko is the editor-in-chief of *WebLogic Developer's Journal* and a senior technical specialist for a leading consulting services company.

**CONTACT:** joe@sys-con.com





©2003 Wily Technology, Inc. The Wily logo is a trademark of Wily Technology, Inc. Java is a trademark of Sun Microsystems in the U.S. and other countries.

Two years without a vacation.  
The application's up. It's down.  
It's up. It's down.

I'm to blame. Steve's to blame.  
Someone's always to blame.



Not any more.

Get Wily.™

*Enterprise Java  
Application Management*  
1 888 GET WILY  
[www.wilytech.com](http://www.wilytech.com)

**wily**  
technology



# WebLogic Enterprise Security



BY PAUL PATRICK

## AUTHOR BIO

As chief security architect for BEA Systems, Paul Patrick is responsible for the overall security product strategy at BEA. He plays a key role in driving the design and implementation of security functionality across all of BEA's products, and is the architect for BEA's new enterprise security infrastructure product WebLogic Enterprise Security. Prior to becoming chief security architect, Paul was the lead architect of BEA's ObjectBroker CORBA ORB and co-architect of WebLogic Enterprise (now Tuxedo). He is also the author of several patent applications as well as industry publications and a book on CORBA.

## CONTACT...

[paul.patrick@bea.com](mailto:paul.patrick@bea.com)

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

## AN INFRASTRUCTURE APPROACH TO ENTERPRISE APPLICATION SECURITY

**B**EA WebLogic Enterprise Security 4.1 offers a new, integrated approach to addressing the distributed application security problem found with enterprise applications.

With this new distributed, infrastructure-based approach, application security becomes a function of the application infrastructure and is separate from the application itself. Any distributed application deployed using BEA WebLogic Enterprise Security can be secured either through the security features included out of the box, or by plugging in other specialized security solutions from major security vendors that the customer's enterprise standardizes on.

This article defines the major requirements for a distributed application security solution, and explains how WebLogic Enterprise Security 4.1 delivers them to your application.

## Introduction

The introduction of Web-based applications, component-based architectures such as J2EE, and now service-based architectures, has brought about a change in how applications are created. Where once an application would be constructed as a single entity containing both business logic and a set of embedded security mechanisms, applications are now constructed by integrating a number of applications that provide services to other components in a distributed environment.

But as these highly distributed applications proliferate, the ability to secure these applications from malicious use from outsiders as well as control the actions of insiders continues to present a critical challenge. A notable effect of this style of application construction is that the number of potential entry points into the application that could be leveraged for malicious



activities increases significantly. With the various components of the application distributed throughout the enterprise and even perhaps across enterprise boundaries, the traditional approach of securing an application at only its perimeter is no longer effective. Security enforced only at the perimeter leaves gaps that can be easily exploited by malicious insiders and results in individual silos of security enforcement at almost every component of the application.

Taming this challenge requires a solution that flexibly stitches the existing application fabric to the existing security foundation, while enabling the efficient administration of policies that govern access to business functions. Application security is not static. Administrators need the power to respond to evolving computing technologies and ever-changing threat environments. They must be able to determine the security posture of every single component executing business functions for which they are responsible. They must be able to update this posture by altering the use of various security technologies or changing the policies governing access to resources. Only by addressing the needs for comprehensive security integration, encapsulated policy enforcement, and responsive administration can an application security solution meet both goals.

Reducing the onerous burden requires two separate innovations: service-based security and unified distributed administration. A service-based security layer offers a universal security abstraction for application containers on one side and pluggable provider interfaces for security solutions on the other side. Of course, such flexibility could create its own set of problems surrounding the configuration of service bindings and maintenance of consistent policies. Avoiding this issue with unified administration requires a robust paradigm for synchronizing, propagating, and analyzing administrative directives.

BEA WebLogic Enterprise Security is the first solution to deliver these two innovations in a single, comprehensive package. It doesn't require enterprises to replace existing application containers or existing security solutions. What it does is allow enterprises to weave these existing components into a seamless whole that is easy to manage, maintain, and extend. For the first time, an information technology organization can have complete visibility into and control over every aspect of security for every business function supported by its applications.

Designed as a security infrastructure for providing security services in a consistent and uniform approach to application containers throughout an enterprise, WebLogic Enterprise Security leverages many of the lessons learned from successful distributed systems while focusing on the reliability, availability, scalability, and performance. In addition, WebLogic Enterprise Security is well suited for environments where an application server decision has not been made. Unlike a number of other products, it does not require customers to utilize any of the components of the BEA WebLogic Platform suite and can be used in environments where these components don't exist (see Figure 1).

One major difference between BEA WebLogic Enterprise Security and other security solutions is the use of a distributed infrastructure that allows for decision points to be colocated with the resources that are being protected. Instead of a central security server where policy decisions are determined, WebLogic Enterprise Security uses a patented approach for distributing configuration and policy information to the decision points that are colocated with the resources that are to be protected. Doing this avoids the performance degradation associated with the latency of network

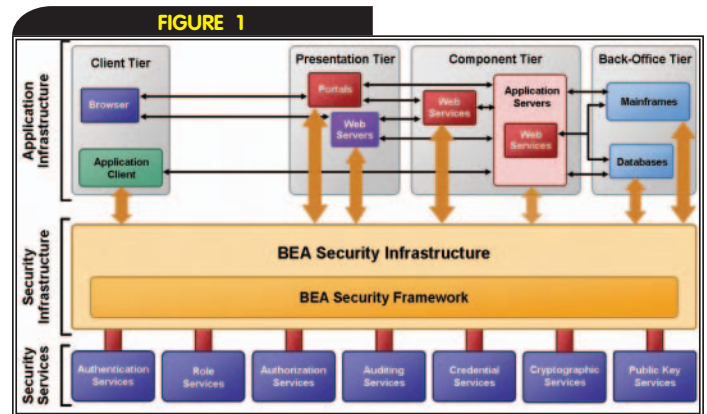
calls to a central decision point, and provides better reliability and availability since there is no runtime dependency on an external process that must be operational and responsive.

At the heart of the WebLogic Enterprise Security infrastructure is a sophisticated security framework known as the "BEA Security Framework", the same one found in BEA WebLogic Server. This allows security services developed for use with WebLogic Server to be utilized by WebLogic Enterprise Security throughout the enterprise. In addition, the use of a common security infrastructure provides customers with a single, unified approach to application security whether or not they use the BEA WebLogic Platform suite.

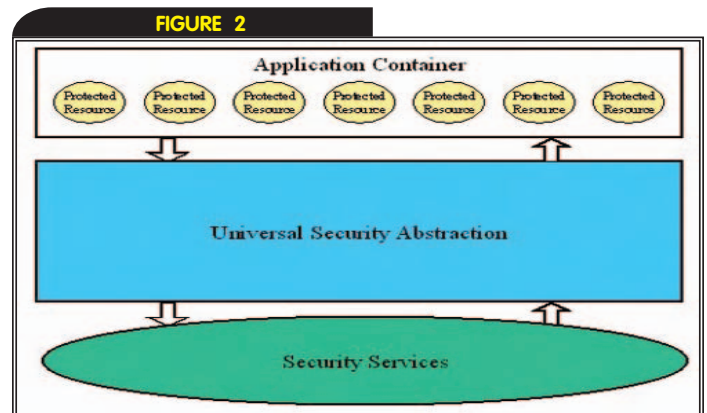
## Service-Oriented Security

The WebLogic Enterprise Security approach is to simplify the integration of application containers with security solutions. An application container is the runtime infrastructure that supports the execution of components. Web servers may act as containers for CGI, JSP, or ASP components. Application servers may act as containers for J2EE and .NET components. Packaged applications act as the containers for the business functions they provide. Stand-alone programs in languages such as Java or C must act as their own containers. Web services may run on top of frameworks, in which case the framework is the container, or as stand alone components, in which case they are like other stand-alone programs. Application components already delegate security functions to the container and WebLogic Enterprise Security takes this process one step further by having the container delegate security functions to it.

In principle, every instance of a particular type of container can



Enterprise security infrastructure



Shielding the application container from security details

use the same integration interface, saving a great deal of time and effort. In practice, the situation is actually even better because the model for this interface can be the same across all container types. There are three primary kinds of information any type of security function might need from a container: the security context of the request, such as the username and password or any embedded security tokens; the identity of the resource that is the target of the request, such as the “change address” method of the “Customer” object in the “Accounts Receivable” application; and optionally the context of the request, such as the request parameters that represent the particular address and the particular customer. These three categories of information are the same for all possible containers and all possible security functions. It’s simply a matter of encoding them according to the conventions of each type of container and dispatching the appropriate pieces of data to each security function in the correct order.

Figure 2 illustrates this approach. When a container receives a request on a protected resource, it makes a call to the universal security abstraction. This abstraction then invokes all the necessary individual security services, shielding the container and the component from the details. The container receives a decision indicating whether it should deny or fulfill the request.

The goal of BEA WebLogic Enterprise Security is to make integration with applications as easy as possible. In cases where applications already execute in a container-like abstraction, it may be

possible to provide shrink-wrapped integration. Containers that provide open mechanisms for extending the container where security decisions can be interposed in the normal flow of handling a business request, such as a Web server’s plug-in mechanism, can be used to integrate with WebLogic Enterprise Security. In its initial release, WebLogic Enterprise Security provides packaged integration for a number of containers, including BEA WebLogic Server and the Netscape/Sun ONE Web Server.

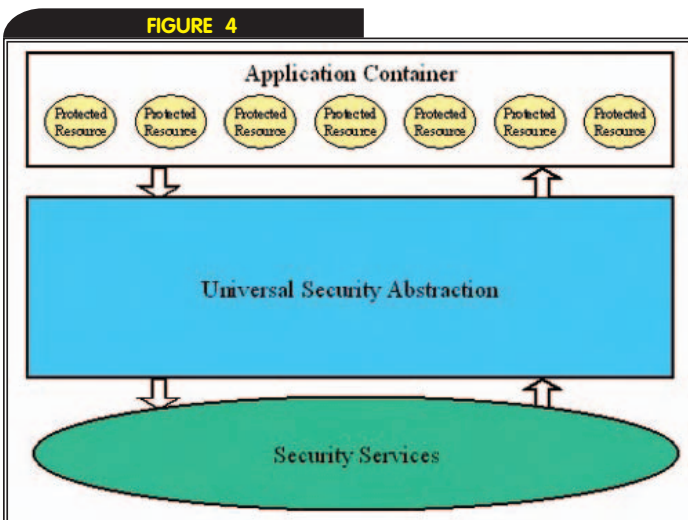
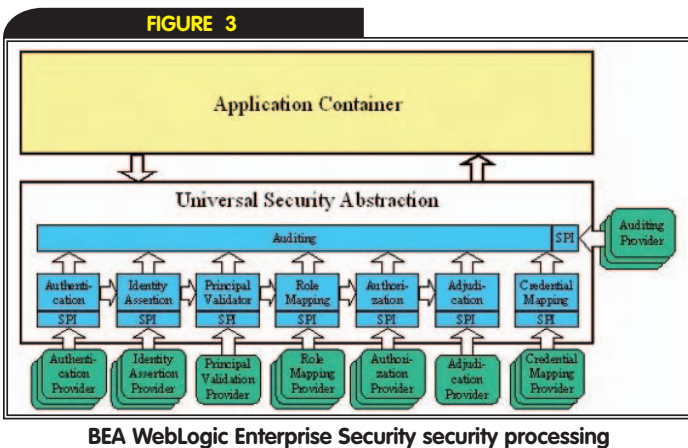
In the case of stand-alone applications, each application must individually call the WebLogic Enterprise Security API. For existing applications, there are a variety of straightforward techniques developers can use to add this delegation. Depending on the internal architecture, such techniques include using interceptors, changing the dispatch function, or creating proxy objects. For new applications, developers can create a mini-container abstraction that intercepts requests, calls WebLogic Enterprise Security and acts on the results. While these techniques all require some additional programming, this effort will be repaid many times over by eliminating the burden of maintaining all the embedded security code.

## Service Provider Integration

After BEA WebLogic Enterprise Security receives a request from an application container, it manages security processing through a sophisticated internal framework. This security framework is the same framework used in BEA WebLogic Server. The first important point to note about this framework is that every step must pass through an auditing phase that generates a comprehensive set of events for the execution of that step. By filtering and capturing these events, an auditing provider can create as fine grained a log as necessary to comply with enterprise policies. The second important point to note is that security processing is a pipeline. Security functions follow a natural order, with downstream steps requiring the results from upstream steps. The requester’s identity must be established before deciding whether to grant that identity access to a resource. Determining what roles an identity currently fulfills must occur before evaluating whether one of those roles authorizes it to perform a particular action on a resource. Within the logical processing order, this processing is very flexible. If a whole new category of security function emerges, WebLogic Enterprise Security can transparently enable it for all application containers by inserting it into its proper place in the pipeline.

For each step defined in the pipeline, WebLogic Enterprise Security invokes the service provider designated to handle that step. As shown in Figure 3, each security service has a corresponding Service Provider Interface (SPI) that defines the functions that security providers providing the service must support. To plug into WebLogic Enterprise Security, a security solution simply has to offer implementations of the SPI for services it knows how to provide. In many cases, these interfaces will consist simply of a wrapper around existing client libraries provided by the solution vendor. By taking advantage of WebLogic Enterprise Security’s universal security abstraction, enterprises can transparently and efficiently switch to alternative services providers, upgrade to new versions of existing providers, or even implement their own custom providers to handle special cases.

Out of the box, WebLogic Enterprise Security includes security service providers for a security service that simply use the framework SPIs. Other implementations of a security service can be created and integrated to the facilities of the underlying framework through the same SPIs. These clean SPIs make it possible to plug







## How Can You Sink Your Application Overhead?

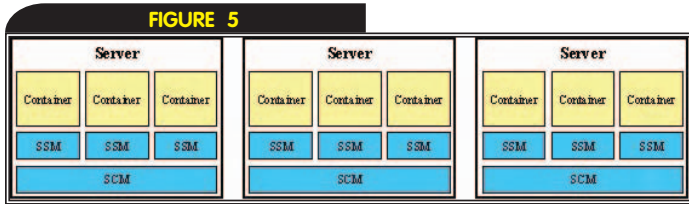
With Cyanea/One®, your company can ensure that the resource consumption of its J2EE applications stays below desired thresholds. Through Cyanea/One's Performance Analysis and Reporting (PAR) engine, you can quickly baseline an application and trend its performance and resource utilization over time. Automatically identify hotspots within the code and take corrective action. Be instantly notified when key threshold conditions are approaching. Improve service levels and customer satisfaction. Accomplish all these without touching your code or even requiring knowledge of your application.

*...Keeping* **Your Applications Under PAR.**



Find PAR at:  
[www.cyanea.com/wldj/underpar.html](http://www.cyanea.com/wldj/underpar.html)  
1-877-CYANEAS | [sales@cyanea.com](mailto:sales@cyanea.com)

Cyanea® and Cyanea/One® are registered trademarks of Cyanea Systems Corp.



**Service Control Modules**

and unplug different security providers as the security ecology evolves, benefiting everyone involved. Although BEA can individually upgrade the providers included with WebLogic Enterprise Security, security vendors can easily make their services available to all supported containers by coding their products to the appropriate SPIs. Moreover, enterprises can quickly implement customized security processing where necessary.

## BEA WebLogic Enterprise Security Architecture Security Service Modules

The pieces of WebLogic Enterprise Security that provide a universal security abstraction are called Security Service Modules (SSMs). An SSM instance includes the interface to the container, the security framework, and all the service providers configured for that instance. Each SSM instance supports a container instance (see Figure 4).

Every SSM requires configuration of its service providers and their corresponding policy information. An initial configuration occurs upon installation and enrollment of the SSM with the administration server, but updates then occur as service providers change and policies evolve. With perhaps a hundred different server machines involved in the execution of some applications, each with multiple instances of containers, the need for a sophisticated approach to administration is pretty clear.

### Service Control Modules

The first point of sophistication is the aggregation of administrative operations across multiple instances on the same machine. In most enterprise architectures, it is quite common to run multiple instances of a Web or an application server on the same machine. In some cases, particularly powerful servers may run instances of different types of containers on the same machine. Obviously, if every instance communicated directly with the administrative system there would be a lot of duplicative resource consumption on that machine. Moreover, many types of containers allow administrators to dynamically create and destroy instances so WebLogic Enterprise Security needs a means to control the creation and destruction of corresponding SSM instances. Therefore, every machine on which SSMs may run has a Service Control Module (SCM) as shown in Figure 5.

### Administration Server

BEA WebLogic Enterprise Security maintains named configura-

“Security enforced only at the perimeter leaves gaps that can be easily exploited”

tions in the administration server. In addition to the service providers assigned to the configuration, it maintains a hierarchy of all protected resources managed by that configuration. This hierarchy can include levels for groups of applications, applications, components, objects, and methods so policies can apply to any level of this tree. Resources inherit policies from their ancestors in this tree, though administrators can override this inheritance. All of this configuration, resource, and policy information resides in the policy store, which can be an Oracle or Sybase database. This policy store also maintains information about administrative roles and privileges. WebLogic Enterprise Security has an administrative resource tree that it protects just like application resources. The tree has four main branches: (1) operations on users and groups; (2) operations on policies for role assignment and authorization; (3) operations on protected resource definitions; and (4) operations on service provider configurations. Each of these branches is further divided. Branch 1 has subdivisions corresponding to the user and group hierarchy. Branches 2 and 3 have subdivisions corresponding to the resource tree. Branch 4 has subdivisions corresponding to the configuration tree. An individual administrator may be assigned create, read, update, or delete privileges for any set of branches of this resource hierarchy. In addition to this flexibility in compartmentalization, WebLogic Enterprise Security offers other features for administration.

## Conclusion

BEA WebLogic Enterprise Security doesn't impose a rigid security model on enterprises that hinders the integration of application components with security services and forces the costly workaround of mixing security code with business logic. Instead, it delivers an open framework, common throughout BEA's application platform suite, so that components running on existing application platforms can seamlessly cooperate with the existing security ecology. This framework eliminates dependencies between application components and security services – new application components can seamlessly utilize existing security services and new security services can seamlessly support existing application components. This capability reduces the life-cycle cost of securing existing application components with existing security services.

By embracing the principles of distributed computing, WebLogic Enterprise Security preserves flexibility without sacrificing control. Its innovative administrative model enables enterprises to have complete visibility into and control over the security configuration of every application component as well as the specific policies used to control access to business functions. They can administer security from a single location, propagating both configuration and policy changes throughout the distributed application fabric. This capability enables better assessment and mitigation of security risks.

In addition to supporting existing security services, WebLogic Enterprise Security offers groundbreaking role mapping and authorization services that make it easy to untangle security code from business logic. Because they offer an unprecedented level of flexibility in evaluating the context of a request, enterprises don't have to mix security code with business logic to achieve policy enforcement. This capability decreases the cost of maintaining applications and enables more responsive risk management. It is representative of BEA WebLogic Enterprise Security's overriding goal – to increase IT efficiency and improve system security while supporting business objectives by embracing business procedures rather than constraining them. 🍓



The world, by way of Memphis.

FedEx, the very model of corporate efficiency, always looks for new ways to improve service. HP helped FedEx IT managers deploy HP OpenView™, which lets them identify and correct potential issues quickly and simply. The result is a smoothly running operation that produces happy customers from Memphis to Monaco—not to mention Mexico, Morocco and Martinique. [www.hp.com/plus\\_fedex](http://www.hp.com/plus_fedex)

fedex + hp

= everything is possible

FedEx

# ebXML and XML Digital Signature



BY SANJAY DALAL

## AUTHOR BIO...

Sanjay Dalal is a staff software engineer at BEA Systems, Inc. He is one of the architects of WebLogic Integration, responsible for the Trading Partner Integration. Sanjay has worked on various business protocols, security, transactions and recovery, clustering, and trading partner management. He is also a co-author of the Business Transaction Protocol and participates in the development of the Web Service Coordination and Atomic Transaction Protocols.

## CONTACT...

sanjay.dalal@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

## WEBLOGIC INTEGRATION TAKES ADVANTAGE OF A GROWING TECHNOLOGY

The data exchanged in business-to-business (B2B) messages is often sensitive and requires protection. Secure Socket Layer (SSL) provides protection at the transport level through the confidentiality of data exchanged between two endpoints.

However, such business messages may travel through multiple endpoints. This makes the message authenticity (who sent it?) and integrity (is it changed on its way?) necessary. Also, it is important to have these facts recorded for non-repudiation purposes.

BEA WebLogic Integration <http://edocs.bea.com/wli/docs8/1> supports electronic business protocol using eXtensible Markup Language (ebXML) for B2B transactions. The ebXML protocol recommends using XML Digital Signature (XMLDSig) to provide message authenticity and integrity. This article provides an introduction to XMLDSig and describes how to use it with ebXML in WebLogic Integration 8.1.

## The Need for Signature in B2B Transactions

Businesses prefer to use the Internet to exchange business documents, as it is a cheap and effective medium compared to expensive, proprietary, and closed networks. The usage of an unreliable and insecure medium such as the Internet brings forth requirements for security of business transactions. Transport-level security that provides encryption, integrity, and authentication at the session level between the endpoints is not enough when messages travel through various intermediate servers. In such cases, confidentiality, authenticity, and integrity need to be provided at the message level. An interoperable and globally accepted message-signing technology is needed to answer questions, such as who sent the message, was the message modified in transit, and can the sender deny sending it, or the recipient deny receiving it?

The rest of this section provides an introduc-



tion to signature. It describes what is legally meant by signature, what a digital signature is, the processes of generating and verifying a signature, and, at the end, the introduction of XMLDSig in the context of ebXML.

The subsequent sections describe the anatomy of XMLDSig including requirements from ebXML, as well as the processes of signing and verifying an ebXML message, and provide information on support and configuration of XMLDSig in WebLogic Integration.

### Signature

According to the American Bar Association, a signature is not part of the substance of a transaction, but rather of its representation or form. A signature serves the following purposes:

- **Evidence:** A signature authenticates a writing by identifying the signer with the signed document.
- **Ceremony:** The act of signing a document calls to the signer's attention the legal significance of the signer's act, and thereby helps prevent inconsiderate engagements.
- **Approval:** In certain contexts defined by law or custom, a signature expresses the signer's approval or authorization of the writing, or the signer's intention that it has legal effect.
- **Efficiency and logistics:** A signature on a written document often imparts a sense of clarity and finality to the transaction and may lessen the subsequent need to inquire beyond the face of a document.

### Digital Signature

Digital signature is a value generated from the application of a private key to a message via a cryptographic algorithm such that it has the properties of integrity, message authentication, and signer authentication, where

- **Integrity:** The data has not been changed, destroyed, or lost in an unauthorized or accidental manner.
- **Message authentication:** A signature identifies what is signed, making it impracticable to falsify or alter either the signed matter or the signature without detection.
- **Signer authentication:** A signature indicates who signed a message. It should be difficult for another person to produce without authorization.

### GENERATING A DIGITAL SIGNATURE

Figure 1 describes the digital signature generation process performed by the signer.

1. A hash function is applied to the data to be signed to generate a digest.
2. The private key of the signer is applied to the digest of the data to be signed using a cryptographic signing function to produce a digital signature.
3. The data and digital signature are sent to a recipient in some associative form such as an envelope.

### VERIFYING A DIGITAL SIGNATURE

After receiving the data and associated digital signature, the recipient performs the process described in Figure 2 to verify the signature.

1. The same hash function that was used while signing the message is applied to the received data to compute a digest.
2. The verifying function is applied using the signer's public key to find out if the signature was signed using the signer's private key and to verify if the computed digest and the digest that was

- transformed into the received signature are the same.
3. Lastly, the signer's public key is verified.

### XML Digital Signature for ebXML Messages

ebXML is a modular suite of specifications that enables enterprises of any size, in any geographical location, to conduct business over the Internet. The ebXML Messaging Service (ebXML-MS) specification defines a communication protocol-neutral method for exchanging business messages electronically between trading partners under some business transaction. It defines specific enveloping constructs that support reliable and secure delivery of business information including attachments. It also defines a set of namespace-qualified SOAP Header and Body element extensions within the SOAP Envelope as shown in Figure 3. An ebXML message is packaged within a MIME multipart to allow attachments to be included as per the SOAP Messages with Attachment specification.

### Why Use XML Digital Signature with ebXML?

XML has become a lingua franca for business transactions conducted over the Internet as it is text based, Internet friendly, and provides a semistructured data format. Also, when a document is exchanged between several participants who may process only a portion of the document, which often is the case in business processes, a technology is needed to selectively sign only a portion of a document. The W3C/IETF joint XML Signature (XMLDSig) is the technology that leverages the power of XML and provides selective signing of documents. Therefore, the ebXML-MS specification mandates XMLDSig as the technology to use to sign the ebXML messages.

### Anatomy of an XML Signature in the ebXML SOAP Envelope

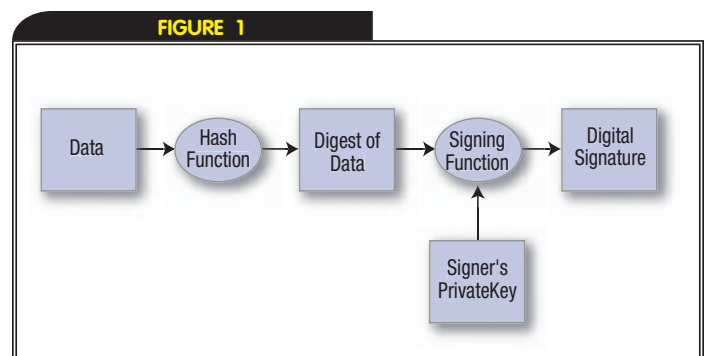
Figure 4 shows the anatomy of an XMLDSig and the subsequent paragraph provides a brief description of each element, including values recommended by the ebXML-MS for certain elements, as well as the relationship between these elements.

The SOAP Header of an ebXML message may contain zero or more Signature elements. The first such element must be signed by the From party of the ebXML message.

Each *Signature* element contains one *SignedInfo* element, which consists of the information signed.

The *CanonicalizationMethod* is the algorithm that is used to canonicalize the SignedInfo element before it is digested as part of the signature operation. For ebXML-MS, [www.w3.org/TR/2001/REC-xml-c14n-20010315](http://www.w3.org/TR/2001/REC-xml-c14n-20010315) (c14n) is the recommended algorithm.

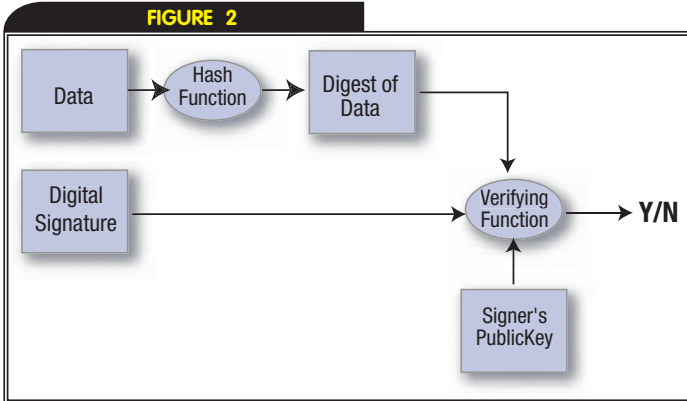
The *SignatureMethod* is the algorithm that is used to convert the



Generating a digital signature

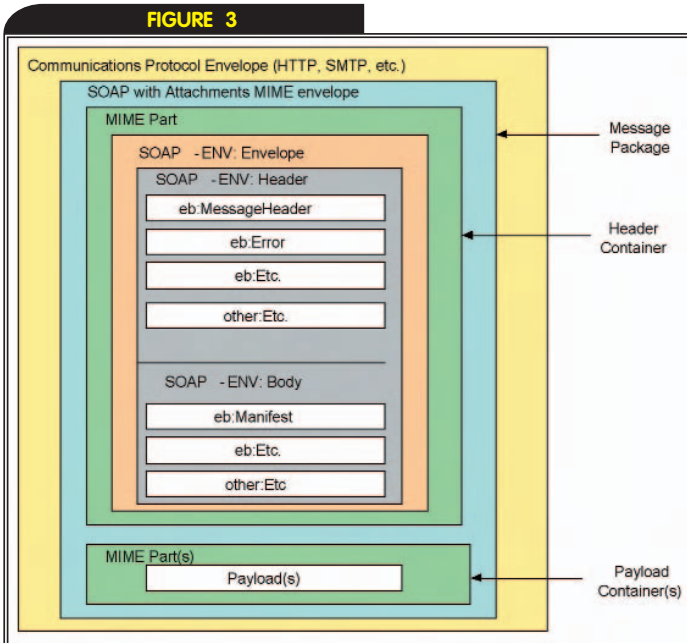


FIGURE 2



Verifying a digital signature

FIGURE 3



ebXML message structure

canonicalized SignedInfo into the *SignatureValue*. It is a combination of a digest algorithm, a key dependent algorithm, and possibly other algorithms such as padding. The ebXML-MS recommends [www.w3.org/2000/09/XMLESig#dsa-sha1](http://www.w3.org/2000/09/XMLESig#dsa-sha1).

The signature can contain references to one or more data (payload) objects in the ebXML message. Each *Reference* element includes the digest method and the digest value calculated over a data object. The optional URI attribute of the Reference identifies the data object to be signed. A URI with the value "" identifies the node-set (without comment nodes) of the XML resource containing the signature while the value of the MIME content-id identifies a payload part of the ebXML message.

*Transforms in Reference* is an optional ordered list of processing steps that are applied to the data object's content before it is digested. Transforms can include operations such as canonicalization, encoding/decoding (including compression/inflation), XSLT, XPath, XML Schema validation, etc. The three transforms required for signing ebXML messages are:

1. **Enveloped Signature** ([www.w3.org/2000/09/XMLESig#enveloped-signature](http://www.w3.org/2000/09/XMLESig#enveloped-signature)): Signature is over the SOAP Envelope that contains the signature(s) as an element.

2. **XPath** ([www.w3.org/TR/1999/REC-xpath-19991116](http://www.w3.org/TR/1999/REC-xpath-19991116)): With expression to exclude signing elements with attributes such as actor that could change as the message passes through intermediaries. It is

```

    not(ancestor-or-self::node() [@SOAP-ENV:actor='urn:oasis:names:tc:ebxml-msg:actor:nextMSH'] | ancestor-or-self::node() [@SOAP-ENV:actor='http://schemas.xmlsoap.org/soap/actor/next'])
  
```

3. **Canonicalization of the SOAP Envelope using** [www.w3.org/TR/2001/REC-xml-c14n-20010315](http://www.w3.org/TR/2001/REC-xml-c14n-20010315)

*DigestMethod* is the algorithm applied to the data after the Transforms are applied to yield the *DigestValue*. The signing of the *DigestValue* is what binds a data object's content to the signer's key. [www.w3.org/2000/09/XMLESig#sha1](http://www.w3.org/2000/09/XMLESig#sha1) is the mandatory digest algorithm for ebXML.

*KeyInfo* indicates the key to be used to validate the signature. Possible forms for identification include certificates, key names, and key agreement algorithms and information.

*Object* is the element where arbitrary data may be placed. An Object element is merely one type of digital data that can be signed via a *Reference*.

## The Process

This section describes the processes of generating a signature for an ebXML message to be sent, and validating the signature when that message is received.

Let's say that the company ABC, Inc., buys laptops from company XYZ, Inc. These companies exchange business messages using ebXML-MS v2.0. Quote is a signed business document sent by XYZ (*From party*) in response to the QuoteRequest message received from ABC (*To party*) as shown in Figure 5.

XYZ uses the following attributes to sign the Quote message.

- **Signature algorithm:** [www.w3.org/2000/09/XMLESig#rsa-sha1](http://www.w3.org/2000/09/XMLESig#rsa-sha1)
- **Digest algorithm:** [www.w3.org/2000/09/XMLESig#sha1](http://www.w3.org/2000/09/XMLESig#sha1)
- **Canonicalization algorithm:** [www.w3.org/TR/2001/REC-xml-c14n-20010315](http://www.w3.org/TR/2001/REC-xml-c14n-20010315)
- **X509 certificate** of the signer party, XYZ, is encoded with the signature as X509Data of KeyInfo.

## Signing of a Message

This section, with the help of Figure 6, describes the steps of the signature generation and packaging process. The steps are divided into three main parts: reference generation, signature generation, and packaging.

1. **Reference Generation** for each data object to be signed:
  - a. Apply the three Transform(s) recommended by ebXML-MS.
  - b. Calculate the DigestValue for the transformed object using the SHA1 algorithm.
  - c. Create the Reference element including the URI for the referred object ("" for the SOAP Header containing the Signature or the value of the Content-id for the payload part with the Quote document), DigestValue, and Digest Method used.
2. **Signature Generation:**
  - a. Create SignedInfo with SignatureMethod, CanonicalizationMethod, and Reference(s).
  - b. Canonicalize the SignedInfo using the algorithm c14n.
  - c. Calculate SignatureValue for SignedInfo using the RSA-SHA1 algorithm and the private key of the Signer, XYZ.

# CRM Guru:



*Henry Holcombe*  
*CTO & SVP of Operations*  
*Globix*

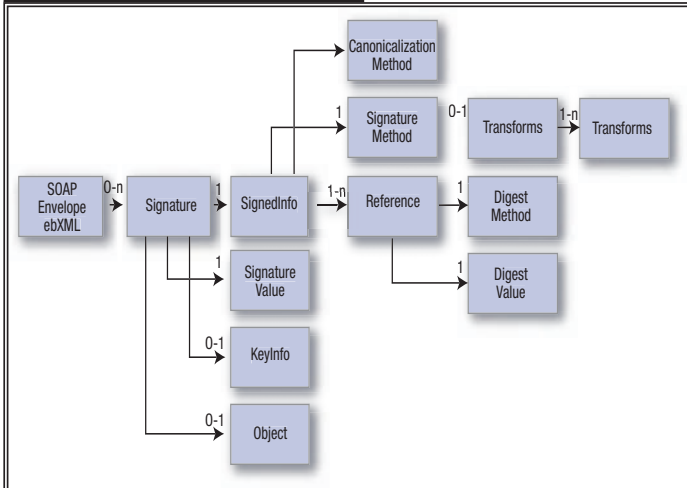
CALL 1•800•NO SOFTWARE

“By integrating with critical internal systems, salesforce.com has become invaluable. It's the easiest and most efficient way for us to get information to our sales force, and it makes it very easy for us to track sales activity and do forecasting.”

Experience the power of combining BEA WebLogic, Web services, and CRM. Download sample code for use with WebLogic Workshop and get your free salesforce.com Developer Edition today at [www.sforce.com](http://www.sforce.com)

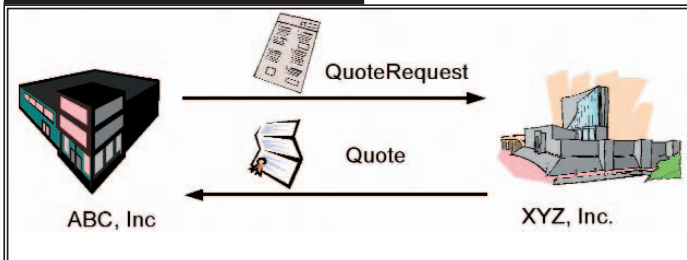
**salesforce.com** Success. Not Software.

FIGURE 4



Anatomy of an XML signature

FIGURE 5



Quote scenario

- d. Create the Signature element and include SignedInfo, SignatureValue and KeyInfo with the X509 certificate chain of XYZ.
3. **Packaging:**
  - a. Package Signature as a SOAP Header.
  - b. Package ebXML multipart message with a SOAP Envelope and a payload part containing the Quote document as MIME parts.

Listing 1 (the code for this article is online at [www.sys-con.com/weblogic/sourcecec.cfm](http://www.sys-con.com/weblogic/sourcecec.cfm)) shows an example of a signed ebXML message.

### Validation of a Signed Message

This section describes the signature validation process. The process consists of four main parts: unpacking the signature from message, signature validation, reference validation, and signer verification as shown in figure 7.

1. **Unpack Signature**
  - a. Obtain the Signature from the SOAP Envelope Header
  - b. Extract the KeyInfo, SignedInfo and SignatureValue from the Signature
  - c. Canonicalize SignedInfo using the c14n algorithm.
2. **Signature Validation**
  - a. Extract XYZ's public key from the KeyInfo
  - b. Apply XYZ's public key and SignatureMethod RSA-SHA1 to validate the SignatureValue of the SignedInfo

3. **Reference Validation** for each signed object
  - a. Get the data object to be digested. This should be obtained by de-referencing the URI of the Reference and applying Transform(s).
  - b. Calculate the digest of the data obtained above using SHA1 as the DigestMethod.
  - c. Compare this digest with the value in the DigestValue received for this Reference.
4. **Verifying the Signer**
  - a. Validate XYZ's certificate; this may include time validity, host name verification, key usage, basic constraint checks, etc.
  - b. Validate XYZ's certificate chain. Make sure none of the certificates in the chain are revoked.
  - c. If ABC and XYZ had exchanged their certificates off-band, make sure that the received certificate's fingerprint matches with the one stored in ABC's database.

### XMLDSig and ebXML in WebLogic Integration

BEA WebLogic Integration provides a J2EE and XML standard-based integration platform to enterprises to integrate business processes with various back-end systems and trading partners. The Trading Partner Integration module of WebLogic Integration helps enterprises automate and manage relationships with customers, suppliers, distributors, and other trading partners by providing support for business transactions conducted using ebXML and other protocols.

The ebXML protocol in WebLogic Integration uses XMLDSig for signing messages. This support consists of the following:

1. Signing of ebXML Message with attachment(s).
2. Signed acknowledgements for ebXML-MS 2.0
3. KeyInfo including X509Data with Signer's certificate chain
4. Signature algorithms
  - a. [www.w3.org/2000/09/xmlsig#rsa-sha1](http://www.w3.org/2000/09/xmlsig#rsa-sha1)
  - b. [www.w3.org/2000/09/xmlsig#dsa-sha1](http://www.w3.org/2000/09/xmlsig#dsa-sha1)
5. Digest algorithm: [www.w3.org/2000/09/xmlsig#sha1](http://www.w3.org/2000/09/xmlsig#sha1)
6. Canonicalization algorithm: [www.w3.org/TR/2001/REC-xml-c14n-20010315](http://www.w3.org/TR/2001/REC-xml-c14n-20010315)
7. Transform algorithms:
  - a. [www.w3.org/2000/09/xmlsig#enveloped-signature](http://www.w3.org/2000/09/xmlsig#enveloped-signature)
  - b. [www.w3.org/TR/2001/REC-xml-c14n-20010315](http://www.w3.org/TR/2001/REC-xml-c14n-20010315)
  - c. [www.w3.org/TR/1999/REC-xpath-19991116](http://www.w3.org/TR/1999/REC-xpath-19991116) (with default XPath expression mandated by ebXML-MS)

The following procedure configures a digital signature for trading partners XYZ and ABC in WebLogic Integration for the scenario shown in Figure 5. For more information on the terminology used and configuration, refer to "Introducing Trading Partner Integration" and tutorials in BEA WebLogic Integration.

### ASSUMPTIONS

- XYZ, Inc., uses WebLogic Integration. A WebLogic Integration domain is already created with custom keystores.

"The data exchanged in business-to-business (B2B) messages is often sensitive and requires protection"



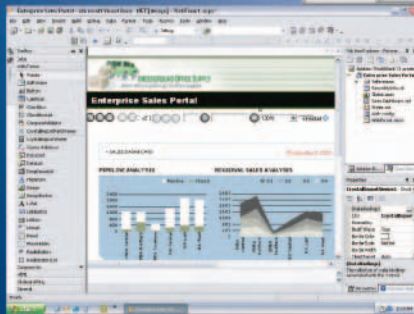
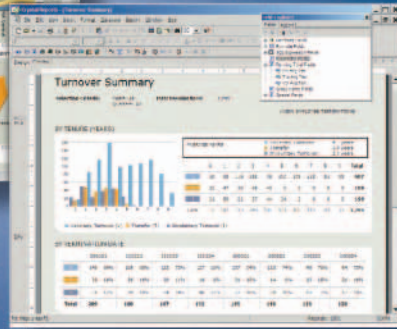
99.9% of the world won't find these  
 screen shots terribly exciting.  
 But if you're in the other 0.1%,  
 yeehaw.

Use Crystal Reports 10 with  
 your J2EE applications

New 100% Java reporting component. Deploy  
 reports across Unix, Linux and Windows  
 Extend Crystal Reports with Crystal Enterprise.  
 Get world-class web report publishing,  
 management, and scalability

Visual Designer simplifies  
 data connectivity  
 Deliver diverse data formatting  
 options within your presentation layer

Access data natively, or via  
 ODBC, JDBC and OLE DB



Expedite .NET and  
 J2EE report integration

Design and integrate reports  
 from within popular IDEs

Flexible Java, .NET and COM SDKs support the  
 tight integration of report interactivity including:  
 group tree navigation, exporting, printing, and  
 drill down



## New Crystal Reports 10.

The best in business intelligence now offers the best in business reporting.  
 New Crystal Reports® 10 is a faster and simpler way for developers to integrate dynamic  
 data into applications and implement high-quality viewing, printing, and exporting. Learn  
 more about Crystal Reports 10 and Crystal Enterprise™ 10, and access technical and evaluation  
 resources at [www.businessobjects.com/v10/059/](http://www.businessobjects.com/v10/059/) or contact us directly at 1-800-877-2340.



- Trading partners ABC and XYZ have obtained public key certificates from recognized Certificate Authorities. These certificates have RSA public key and necessary key usage for signing messages.
- XYZ offers a quote service using QuoteService.jpd as a BEA WebLogic Integration business process. This process receives the QuoteRequest document and responds with the Quote document. This process is already deployed in the WebLogic Integration domain.
- ABC's system is running and ready to send QuoteRequest to XYZ. It has XYZ's public key certificate in its database.

### Steps to Configure WebLogic Integration:

1. Start WebLogic Integration server and log in to the WebLogic Integration console.
2. Click Trading Partner Management->Profile Management.
3. Create a Partner Profile for XYZ.
  - a. Click Create New and fill all necessary information for XYZ. Make it a LOCAL trading partner. Click Add Profile.
  - b. Click Add Binding, select ebXML 2.0 and click Create Binding.
  - c. Change the Endpoint information. Click Add Certificate where Signature Certificate is required.
  - d. Choose Import certificate from file and click Next.
  - e. If the private key is password protected, you will have to create a password alias using Add alias.
  - f. Provide the location of the private key and certificate for XYZ and enable the Import Certificate In KeyStore checkbox. Click Create Certificate. This private key and certificate is used to send a signed Quote from XYZ to ABC.
  - g. Enable the Signature Required checkbox and click Add Binding.
4. Create a Partner Profile for ABC.
  - a. Follow steps 3a-3d above for ABC, but specify ABC as a REMOTE trading partner.
  - b. Provide the location of the file where the public key

## “Message-level authenticity and integrity are some of the basic requirements for B2B messages”

certificate of ABC is kept. Enable the Import Certificate In KeyStore checkbox. Click Create Certificate. This is needed to complete the signer verification process in case ABC sends signed messages to XYZ.

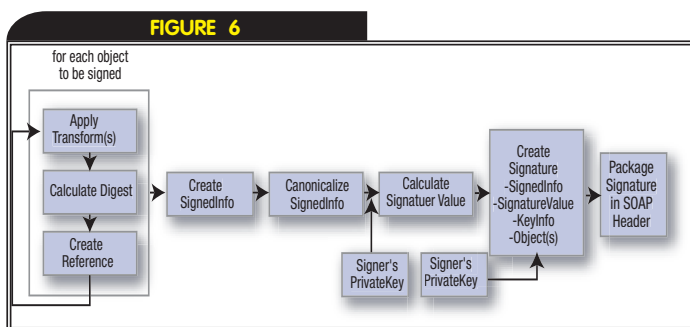
- c. Enable the Signature Required checkbox and click Add Binding. This indicates that ABC is expecting signed documents from XYZ.
5. Create a ServiceProfile between XYZ and ABC for service QuoteService.jpd by following the steps from Trading Partner Management->Service Management.
  6. A third-party plug-in to perform certificate verification can be configured in WebLogic Integration as described in Trading Partner Integration Security.
  7. To enable nonrepudiation, configure a Secure Audit Log provider. Follow the steps described in “Trading Partner Management” for more information.

### Conclusion

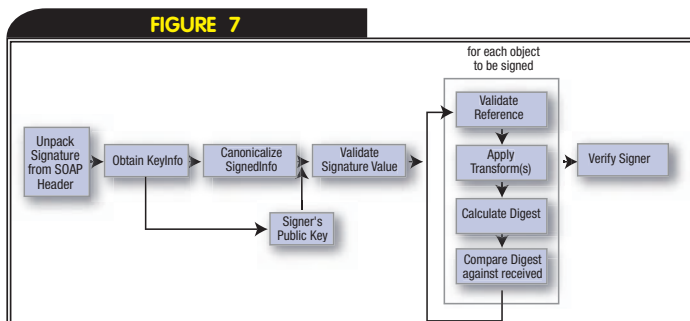
Message-level authenticity and integrity are some of the basic requirements for B2B messages that travel through various intermediate servers within and outside of enterprises. XMLDSig is best suited as the technology that achieves this for ebXML. WebLogic Integration supports this basic functionality for its ebXML protocol stack to help its customers take advantage of this technology.

### References

- *ebXML, Electronic Business using eXtensible Markup Language*: [www.ebxml.org](http://www.ebxml.org)
- *XML-Signature Syntax and Processing*. W3C Recommendation, February 2002: [www.w3.org/TR/xmlsig-core](http://www.w3.org/TR/xmlsig-core)
- *ebXML Messaging Service Specification, v2.0*: [www.oasis-open.org/committees/ebxml-msg/documents/ebMS\\_v2\\_0.pdf](http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf)
- *Digital Signatures in Trading Partner Integration Security, WebLogic Integration*: <http://edocs.bea.com/wli/docs81/tpintro/security.html#1097886>
- Simon, Ed et al. “An Introduction to XML Digital Signatures” August 2001, XML.com. [www.xml.com/pub/a/2001/08/08/xmlsig.html](http://www.xml.com/pub/a/2001/08/08/xmlsig.html)
- Shirey, Robert. “Internet Security Glossary”.
- *Digital Signature Guidelines*, American Bar Association. [www.abanet.org/scitech/ec/isc/dsgfree.html](http://www.abanet.org/scitech/ec/isc/dsgfree.html)
- Freed et al. (November 1996) “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, Part Two: Media Types”. [www.rfc-editor.org/rfc/rfc2045.txt](http://www.rfc-editor.org/rfc/rfc2045.txt). [www.rfc-editor.org/rfc/rfc2046.txt](http://www.rfc-editor.org/rfc/rfc2046.txt).
- Levinson, E. (August 1998) “The MIME Multipart/Related Content-type”. [www.rfc-editor.org/rfc/rfc2387.txt](http://www.rfc-editor.org/rfc/rfc2387.txt)
- Box, D. et al. W3C-Draft. Simple Object Access Protocol (SOAP) v1.1. Box D. et. al., W3C Note 08 May 2000, [www.w3.org/TR/2000/NOTE-SOAP-20000508/](http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)
- Barton, J. et al. “SOAP Messages with Attachments”. Oct 9, 2000. [www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211](http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211)



Generating XML Signature



Verifying XML Signature

As Java development evolves, so does JProbe®

## JProbe®

Find the cause of J2EE code performance, memory and threading problems faster than ever before with Quest JProbe. New investigative features for finding memory problems combined with dramatic performance improvements mean even quicker resolution of problems in your application, servlet, JSP and EJB code.

JProbe Suite

JProbe Profiler

JProbe Memory Debugger

JProbe Threadalyzer

JProbe Coverage

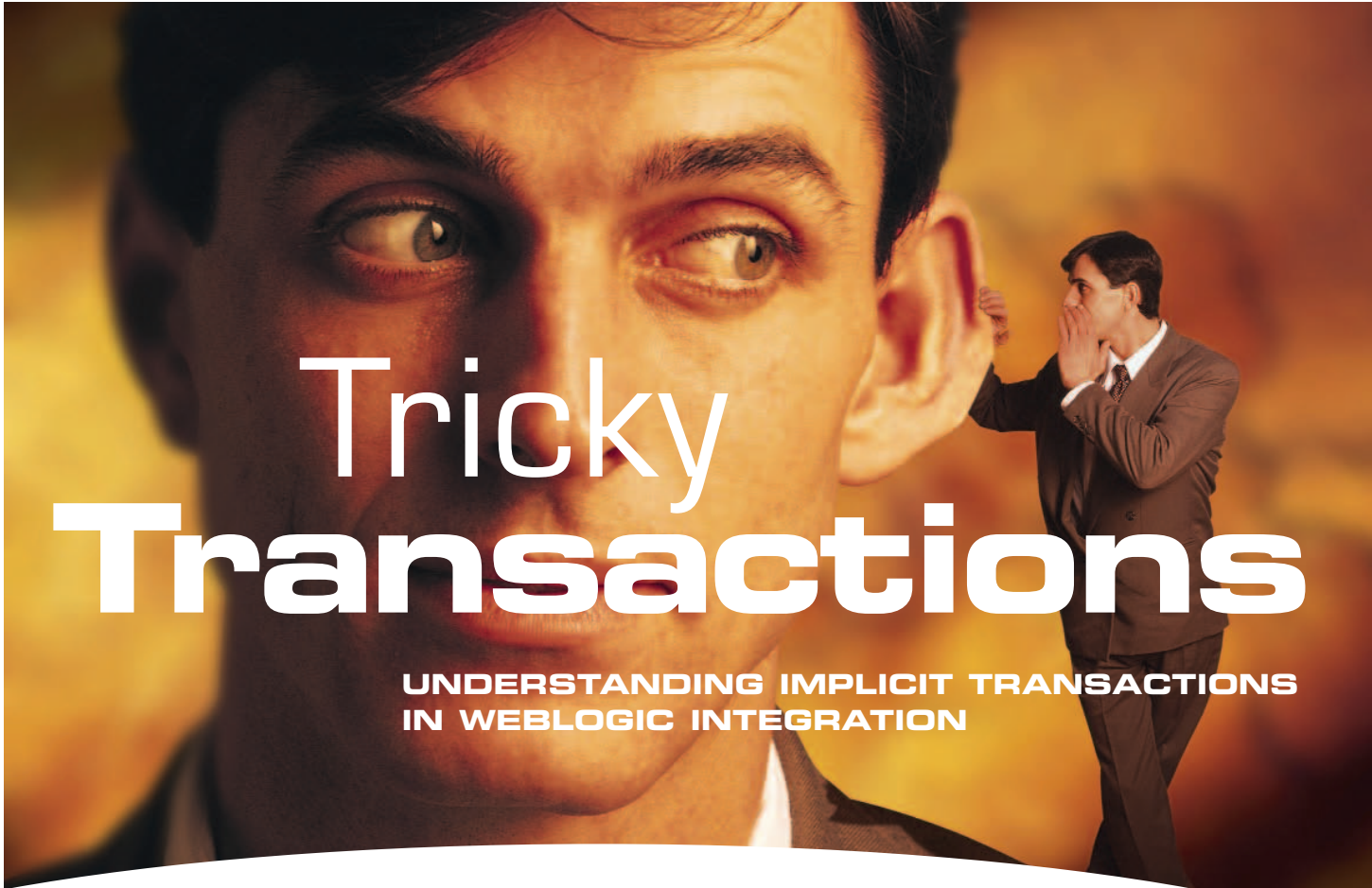
Part of the **Quest Performance Management Suite for the J2EE Platform**



For more info and a free eval, visit:

<http://java.quest.com/jprobe/wldj>





# Tricky Transactions

UNDERSTANDING IMPLICIT TRANSACTIONS IN WEBLOGIC INTEGRATION



BY JOHN GRAVES

## AUTHOR BIO

John Graves has been with BEA for more than five years, and works as an internal trainer for the World Wide Technical Training organization within BEA. He has also worked in the Advanced Services Group doing full life-cycle project work.

## CONTACT...

[john.graves@bea.com](mailto:john.graves@bea.com)

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

**B**EA WebLogic Workshop is fabulous at hiding many of the complexities of a J2EE application. This can be a problem when things don't behave this way you might expect. Tools are great, but sometimes do things "for us" that we really don't want.

Understanding what the tool is doing is key to knowing when a problem might arise. In this article I'll cover the details of how WebLogic Integration handles transactions, referred to as implicit transactions, within a BPM process. Truth be told, you really shouldn't need to know or care about these transactions. There are, however, some cases where they become important. Knowing how they work and where they exist is key to understanding the implications of process design.

The first and most obvious implication is performance. Transactions are inherently expensive. If you have unnecessary implicit transactions, it may be possible to modify the process to reduce these

and improve performance. However, the biggest performance impact is transitioning from one implicit transaction to two. This is due to how WebLogic implements a given process. If a process has only one implicit transaction, then WebLogic Workshop can deploy this process as a stateless session bean. If it has more than one, it must use an entity bean. As you might guess, an entity bean adds overhead to the process and reduces performance.

I won't go into detail about the architectural patterns associated with processes with a single transaction versus a process with multiple transactions. Suffice it to say that a single transaction process is usually a worker or utility process. It will take something in, do the work, and send it somewhere else. Obviously we want these processes to be fast, and we want to make sure we do not accidentally introduce an extra implicit transaction. Processes with multiple transactions

*"Understanding what the tool is doing is key to knowing when a problem might arise"*

are usually long running where performance is not a huge concern. The second implication of implicit transactions is behavior. Transaction boundaries are set for you, so it may not be obvious that a failure in one node of the process may actually cause the rollback of another node.

Consider the process segment shown in Figure 1. A developer may add a logging node above a JMS send node to try to figure out why the JMS is failing. Perhaps they are confused and believe that if the JMS fails, it will also roll back the entry in the database because they are in a single transaction.

Be aware that it is always possible to place an explicit transaction into the process. These are not confusing because the developer manually inserts them and they are clearly visible when the process is displayed.

So how do we know where the implicit transaction boundaries are located? The basic rule of thumb is: if the process stops and waits, then it must start a new transaction. There are a couple of exceptions to this that I will address later. This sounds easy but can get tricky. Here's a quick education.

I will use perform nodes as placeholders for discussion.

Consider the process segment shown in Figure 2. It is a very simple process that has a start node, perform, and end. It is easy to identify that this has a single implicit transaction, and that this process will be deployed as a session bean.

Consider the process segment shown in Figure 3. It adds a control receive into the process. As I said before, whenever the process has to wait, it adds a new implicit transaction. There will be two transactions then: one from the start node to the top of the Control Receive (Tx 1) and one from the Control Receive to the end (Tx 2).

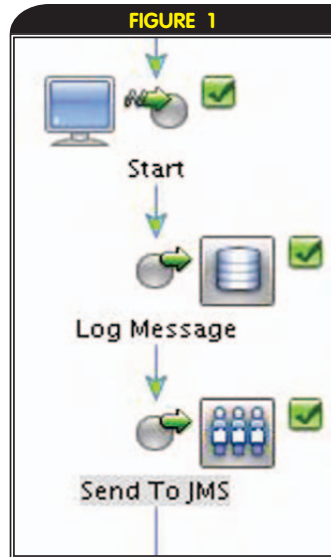
*Note:* A synchronous Control Send with Return does not add an implicit transaction.

Consider the process segment shown in Figure 4. It includes a parallel path. Whenever one is reached, a new transaction is started for each branch. Once the parallel paths come back together, another transaction is started. In this instance, there are four transactions. There are two points to be noted. If the parallel join node is an OR rather than an AND, one of the transactions (Tx 2 or Tx 3) will not execute. I'll cover this in more detail later. In the initial release of BEA WebLogic Integration 8.1 there was a problem with the final transaction. A new transaction was not started after the parallel join. Nodes after the join would be in the same transaction as the final parallel path executed. This has been fixed in Service Pack 2.

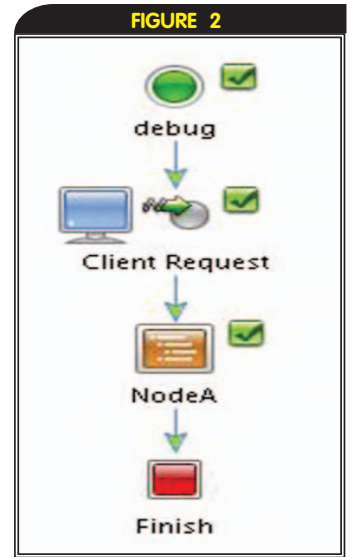
Consider the process segment shown in Figure 5. It includes an event choice. Much like a single Client or Control Receive, a new implicit transaction is started. Then, whichever event comes in first will be in the same transactions as the nodes below the event choice. Unlike the parallel node, there is *no* new transaction created when the paths visually come back together.

Consider the process segment shown in Figure 6. It is the most confusing case of transactions. The process never really stops, but a second transaction is started after the Client Response. For brevity's sake, I will not go into details here. Suffice it to say that in order to implement this process, it must save its state, return data back to the client (synchronous), and then restart the process at the node following the return.

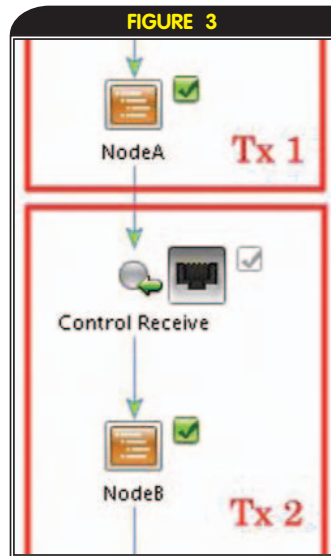
That's it! It appears easy but can be tricky. Now I'll go through a series of tests. For each figure, write down (or draw) where you think the transaction boundaries exist. The answers are provided to show you how well you did. The average for an "expert" is 80%.



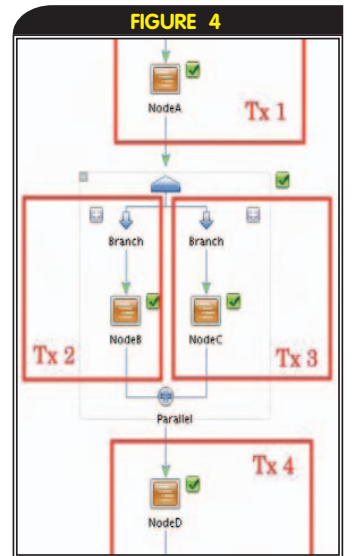
Process segment



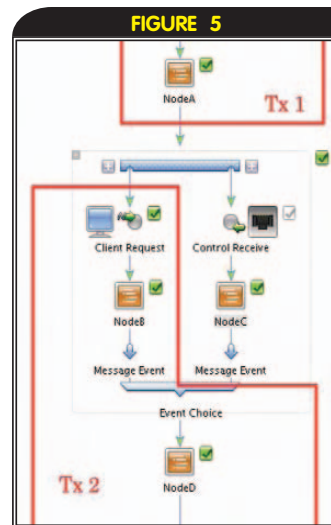
Process to deploy as session bean



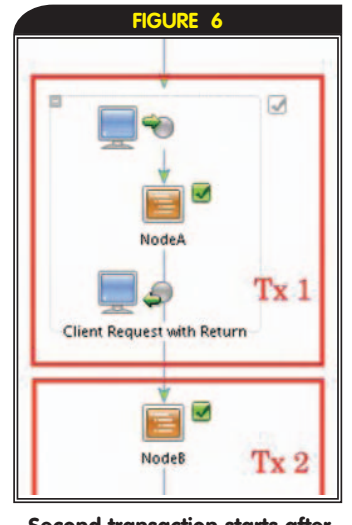
Add control receive into process



Parallel path

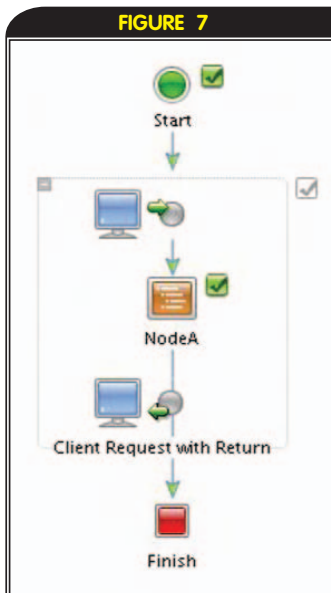


Event choice

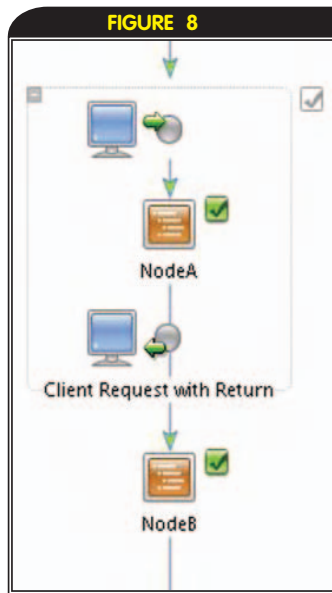


Second transaction starts after the Client Response

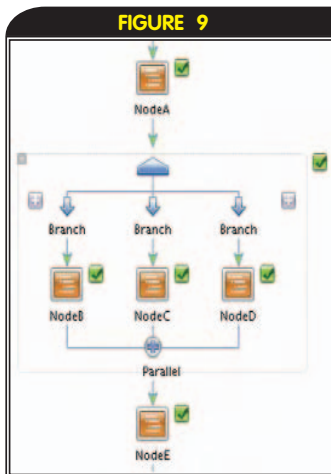




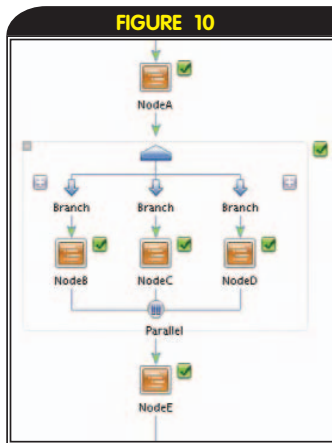
Simple synchronous client



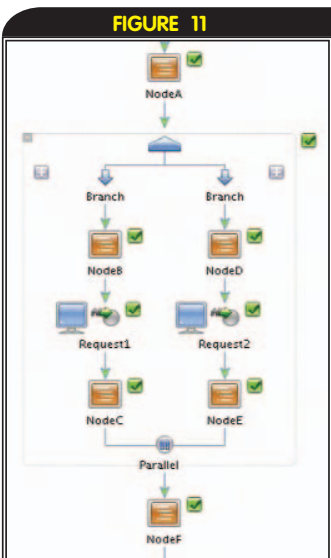
Add perform node



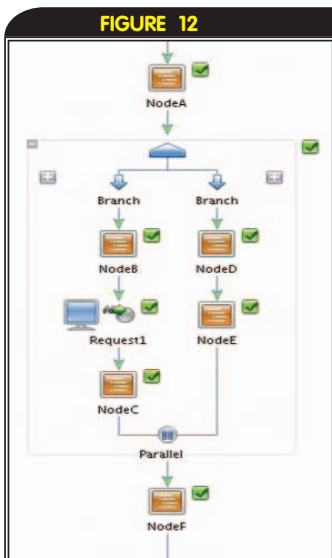
Branch with three paths



The end of the parallel paths is an OR



Two branches with client request node added



Client request node removed from second branch path

Note: These processes are for explaining implicit transactions only. They do not demonstrate best practices for building processes.

See if you can determine how many transactions each example contains.

Good luck!

**Question #1 (Figure 7)**

This is a simple synchronous client request with return.

**Question #2 (Figure 8)**

This adds a perform node after the client return.

**Question #3 (Figure 9)**

This has a branch with three paths. Assume there are no transactions before Node A or after node E.

**Question #4 (Figure 10)**

This is the same as Figure 9 except the join at the end of the parallel paths is an OR rather than an AND.

**Question #5 (Figure 11)**

This has two branches, but each branch has a client request node added.

**Question #6 (Figure 12)**

This has the client request node removed from the second branch path. Be careful, this one is tricky.

**Question #7 (Figure 13)**

This has a three-branch Decision node.

**Question #8 (Figure 14)**

This has a While Do loop.

**Question #9 (Figure 15)**

This has an Event Choice.

**Question #10 (Figure 16)**

This has a loop around a Client Receive. Assume the While Do returns back to the top twice (Node B and Node C Path executes three times).

**Final Exam (Figure 17)**

This one has a little bit of everything. If you answer this correctly, you are on your way to understanding implicit transactions!

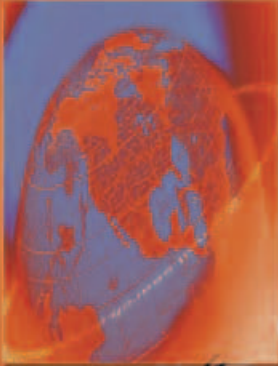
How did you do? It's not as easy as it looks.

Be sure to watch out for transactional boundaries when you are building your processes. If the behavior is not what you want, then put in an explicit transaction. If you have a utility process that is simply doing work, be sure not to introduce too many transactions. BEA WebLogic Workshop and BEA WebLogic Integration are powerful tools. Use them wisely and take advantage of their strengths. 🍷

**Answer #1 (Figure 7)**

There is one transaction. If a node were placed after the return (before the finish), then a second transaction would be created.

BEA's Workshop can be even  
more amazing for the phone.



One Application.  
Web. Voice. It's Easy.

AppDev™ VXML  
for BEA's WebLogic™ Workshop

Delivering Web applications to phone  
users is as easy as clicking a mouse.



For additional information:

SandCherry, Inc.  
1715 38th Street  
Boulder, CO 80301

+1 (720) 562-4500 Phone  
+1 (866) 383-4500 Toll Free  
+1 (720) 562-4501 Fax

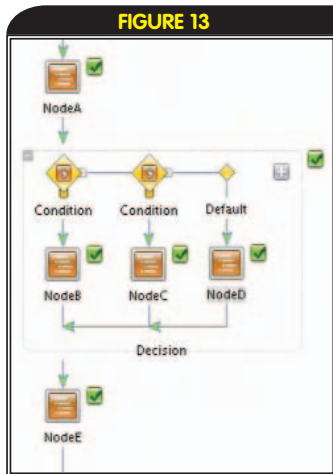
Info@sandcherry.com  
www.sandcherry.com

Copyright © 2004 SandCherry, Inc. All rights reserved.

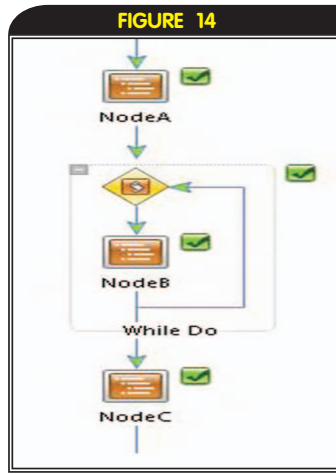
Download  
30 Day Free Trial  
www.sandcherry.com



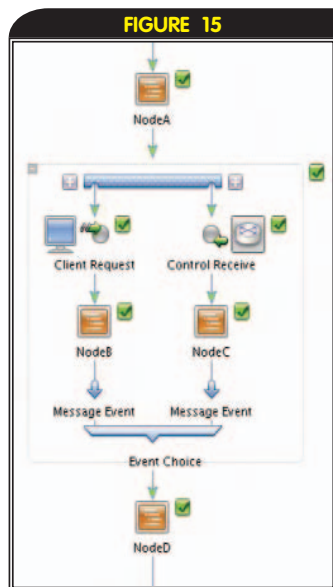




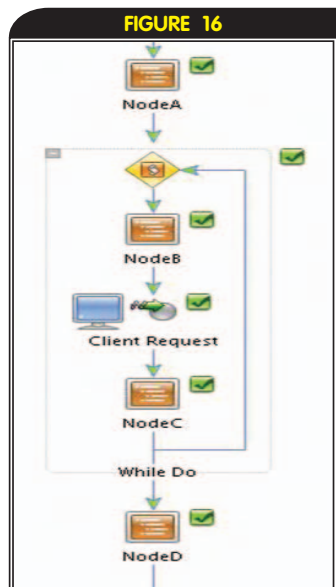
Three-branch Decision node



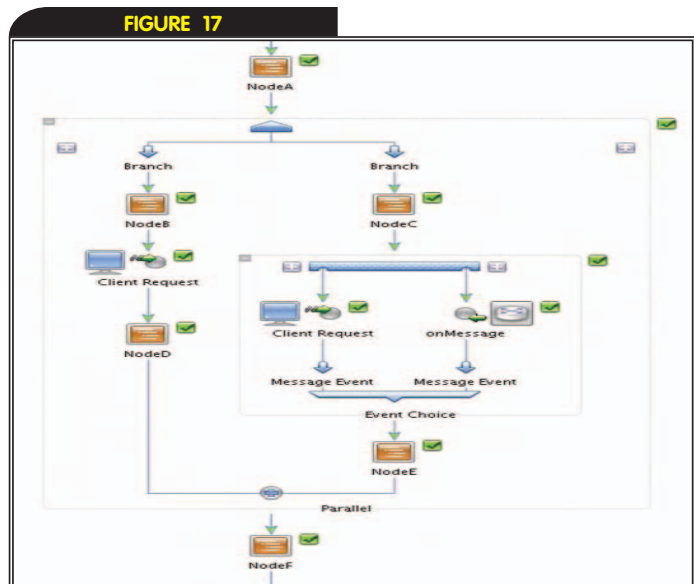
While Do loop



Event Choice



Loop around Client Receive



A little of everything

**Answer #2 (Figure 8)**

There are two transactions. Node A is in Tx 1, Node B is in Tx 2.

**Answer #3 (Figure 9)**

There are five transactions. Each of the perform nodes is in its own transaction. *Note:* This is only true because it is an AND join at the end. If this was an OR, then only one of the branches would be executed.

**Answer #4 (Figure 10)**

There are three transactions. Node A is in Tx 1, Node B (or C or D) is in Tx 2, and Node E is in Tx 3. Since the parallel node is an OR, it will not go down the other two paths, therefore, Node C and Node D will not run. *Note:* It is indeterminate as to which branch of a parallel branch is executed first. The sequence may change between executions.

**Answer #5 (Figure 11)**

There are five transactions. Node A is in Tx 1. Node B is in Tx 2. Node D is in Tx 3. Assuming client request 1 comes in before client request 2, Node C is in Tx 4; the parallel branch ends because it is an OR and Node F is in Tx 5.

**Answer #6 (Figure 12)**

There could be three or four transactions. Assume the left path is executed first. Node A is in Tx 1. Node B is in Tx 2. Nodes D and E are in Tx 3. Node F is in Tx 4. What happens if the right path is executed first? Node A is in Tx 1, Node D and E are in Tx 2 and Node F is in Tx 3. Node B is not executed in this case.

**Answer #7 (Figure 13)**

There is one transaction. I never mentioned decision nodes in the previous discussion. Based on the decision, Node A; Node E; and one of B, C, or D will all be in the same transaction.

**Answer #8 (Figure 14 )**

There is one transaction. Looping constructs have no effect on transactions.

**Answer #9 (Figure 15)**

There are two transactions. Node A is in Tx 1. Assume the Client Receive is first. Nodes B and D are in Tx 2. If the Control Receive is first, then Nodes C and D are in Tx 2.

**Answer #10 (Figure 16)**

There are four transactions. Nodes A and B are in Tx 1. Nodes C and B are in Tx 2. Nodes C and B are in Tx 3. Nodes C and D are in Tx 4. It is important to mention that nodes at the bottom of the loop are in the same transaction as those in the top of the loop in the next iteration. In this case, a rollback may be confusing.

**Answer Final Exam (Figure 17)**

There are six transactions. Node A is in Tx 1. Assume the left branch is first. It doesn't matter. Node B is in Tx 2. Node C is in Tx 3. Assume the far left Client Receive is first. Node D is in Tx 4. It waits because the join node on the Parallel path is an AND (+). Assume either Client Request or onMessage is received next. Node E is in Tx 5. Node F is in Tx 6.

# Generate **dynamic** rich content in your server-side Java applications

## JClass<sup>®</sup> ServerViews

Add professional, dynamic charts and documents to your Servlet, J2EE and Web Services applications.

Flexible and rock-solid, JClass ServerViews works with the latest JREs, web servers and application servers.

### JClass ServerChart

Describe and pass charts as XML, load XML-formatted data, and output to browsers as Flash, GIF, JPEG and more.

### JClass ServerReport

Generate high-quality Adobe PDF for a global audience.



Evaluate and experience JClass today – visit:  
<http://java.quest.com/jcsv/wldj>





# Handling System Core Files

## BEFORE YOU DEBUG

**T**his article will provide some useful tips for debugging your BEA WebLogic Server applications when a system core file occurs. It describes debugging tips, problem troubleshooting, and tools available to assist you in this process.

A system core file is usually indicative of an error in some native code. This could be from the application code of a user (if you are using native code [JNI] in your application), an error in the Java Virtual Machine version you are using, or in BEA WebLogic Server itself. There are a couple of places where native code could have caused the system core file to happen on your operating system. The following ideas and suggestions will help narrow the problem and stabilize the application until the exact cause of the system core can be determined.

### The Java Virtual Machine

The first place to look is at the JVM itself. The JVM is a native program and can cause such errors. When in doubt, you can try another certified JVM and/or a later release to determine if a JVM bug is at fault. Many JVM bugs involve the use of the JIT compiler and disabling this feature will often resolve this type of problem. This can be done by supplying the “-Djava.compiler=none” option to the Java command line. Most of the time the JVM will produce a small log file that may contain useful information as to which library the system core may have come from; however, this is not true all of the time. The file is produced in the directory where BEA WebLogic Server was started and is of the form “hs\_err\_pid<PID #>.log”, where “<PID #>” is the process ID of the BEA WebLogic Server process. On AIX this file would be “javacore<PID>.<ID Number>.txt”, where “<PID #>” is the process ID of the BEA WebLogic Server process and “<ID Number>” is a number generated by the operating system. You can go to <http://java.sun.com> and

search the Bug Database with the HotSpot “Error ID” number or the method that was happening when the error was reported.

### Native Code

The next place to look is at any native libraries that are accessed with JNI (native code) calls in your application, if applicable. This can also cause a system core file to be produced. If your application uses such libraries, they should be carefully examined. It may be difficult to rule out these libraries, as their functionality may not be easily removed from the application. Extensive logging may be needed to determine if a pattern of use can be correlated with the system core dump/Dr. Watson error.

Another place is any Type 2 JDBC driver that makes use of native DBMS libraries, which could also produce this type of error. An option is to switch to a pure Java (Type 4) JDBC driver in order to determine if that is the cause. You can also check with the Type 2 JDBC driver provider to see if there are any known issues or if there is an updated version of the native libraries available.

A final place that could have caused this could be in the BEA WebLogic Server performance pack. This is also native code and when enabled could potentially produce such an error. One option is to disable this feature to determine if that is the cause. This can be done by adding “-Dweblogic.NativeIOEnabled=false” to the command line to start your server. If you are on an older service-pack version you can also check the list of fixed change requests for your particular BEA WebLogic Server version at <http://edocs.bea.com>.

### System Core File

If after looking at the above items you cannot determine the cause of the error, then we can examine the system core file that was produced. If a system core file is produced it will be in the directory where BEA WebLogic Server was started. You will need to run an operating system-specific debugger on the resulting system core file to get a stacktrace that may help in pointing to the offending code.

BY **STEVE POZARYCKI**

#### AUTHOR BIO

Steven Pozarycki is a backline developer relations engineer with BEA Systems, Customer Support Group. He specializes in troubleshooting and solving complex customer issues with their mission-critical applications on BEA products. Steve holds a bachelor's degree in computer science.

#### CONTACT...

[spoz@bea.com](mailto:spoz@bea.com)

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

Now  
More  
Than  
Ever  
Before...

JAVA DEVELOPER'S JOURNAL

**JDJ**

Means

**Business**



Subscribe Now!

**\$69.99**  
1 YEAR/  
12 ISSUES

[www.SYS-CON.com/JDJ](http://www.SYS-CON.com/JDJ)

or call

**1-888-303-5282**

**The World's Leading i-Technology Magazine Just Got Better!**

Now with expanded business coverage, *JDJ* is the world's premier independent, vendor-neutral print resource for the ever-expanding international community of IT professionals.



**The Most Popular  
i-Technology Magazine  
in the World !**

**SYS-CON  
MEDIA**

The World's Leading i-Technology Publisher





If you are in development and the system core consistently happens, you can set the following flags to allow a thread dump to be taken of the server right before a core happens. This will get the state of the threads at that moment and may help narrow the problem to application code or point to a bug. The option is “-XX:+ShowMessageBoxOnError” on the Sun JVM (which is not officially documented on the Sun Web site). When the JVM crashes, the program will prompt: “Do you want to debug the problem?” You can then take a thread dump of the JVM. This option will be available on the 8.1 SP2 version of the BEA JRockit JVM when it becomes available. However, in that version the corresponding option will be “-Djrockit.waitonerror”.

The best option is to run a debugger on the resulting system core file to get a stacktrace of the native calls. The following information was also given in my last article (*WLDJ*, Vol. 3, issue 2) but it is being repeated here for consistency. This information may help point out the offending code to you; or, if you are unsure, then contact BEA Customer Support with this information so they can investigate the stacktrace more thoroughly. If you are on a Windows platform, then a “Dr. Watson” file may be produced. Please send this file to BEA Customer Support when opening a case. Otherwise, check the following “Unix” operating system values to make sure that they have already been properly set in order to generate a core file:

1. Check the “ulimit -c” (configured size of the core file) at a system and user level to make sure that it is set and that the value is not set too low to produce a meaningful core file.
2. Check the available disk space for the user (for example, is there a disk quota?).
3. Also check the following parameter, which on Solaris is usually in /etc/system file and can be used to disable core files: set sys:coredumpsizes=0.
4. On Linux, the core dump is turned off by default on all systems. In RedHat Advanced Server 2.1 it should be under “/etc/security”. There should be a self-explanatory file called limits.conf; within that file look for the word “core”. If it is set to “0”, then core dump files are disabled.
5. On HP-UX check the setting called “kernel parm maxdsiz” (max\_per\_proc\_data\_size, which increases the User Process Data Segment Size) from the old value of, say, 64M to something higher like 134M.

Please get a stacktrace (or backtrace) from your debugger. Here are the commands needed when using “dbx” or “gdb”, which will work on most platforms:

#### a. dbx:

- **\$ java -version:** Need to use right version of JDK
- **\$ ls /opt/bin/dbx:** Need to know dbx location, or “which dbx”
- **\$ export:** DEBUG\_PROG=/opt/bin/dbx (or wherever “dbx” is located)
- **\$ <path to java command>/java corefile**

Now you will be in the debugger. Execute the following commands:

- **(dbx) where:** Shows a summary of the stack
- **(dbx) threads:** Shows the state of the existing threads
- **(dbx) quit**

#### b. gdb

- **java -version:** Need to use right version of JDK
- **ls /usr/local/bin/gdb:** Need to know gdb location or “which gdb”

- **export DEBUG\_PROG=/usr/local/bin/gdb:** Or wherever “gdb” is located)
- **<path to java command>/java corefile**

Now you will be in the debugger. Execute the following commands:

- **(gdb) where:** Shows a summary of the stack
- **(gdb) thr:** Switch among threads or show the current thread
- **(gdb) info thr:** Inquire about existing threads
- **(gdb) thread apply 1 bt:** Apply a command to a list of threads, specifically the backtrace to thread #1)
- **(gdb) quit**

If you don't have access to a debugger you can check to see if you have the “pstack” and “pmap” utilities on your operating system. If you do have those utilities (on some operating systems you have to download these utilities separately); you can run them on the system core file to gather information for support. The syntax of the command would be something like this:

```
$ /usr/proc/bin/pstack core
$ /usr/proc/bin/pmap core
```

## Further Information

You can also go to <http://support.bea.com> and find some published solutions on core files. In the “Question” field type in “S-16147” to display some additional information about using pstack and pmap on Solaris.

### Enabling/Disabling Dr. Watson

The drwtsn32.log files are similar to system core files on Unix. On Windows 2000, these files are found in the following directory: C:\Documents and Settings\All Users\Documents\DrWatson. After entering “drwtsn32 ?” in a DOS command window and hitting enter, the “Dr. Watson for Windows 2000” box appears. The DrWatson log file overview option will display a screen that explains the format of the drwtsn32.log files.

By default, Dr. Watson will be enabled when Windows NT is installed. To disable Dr. Watson, the following Registry value must be changed from 1 to 0 (zero): “\HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug”. An entry called “Auto” corresponds to how Dr. Watson will start up. To enable Dr. Watson, change the “Auto” value from 0 (zero) to 1. This will then launch whatever debugger, or application, is under the Debugger registry value. For Dr. Watson, the Debugger value should contain:

```
drwtsn32 -p %ld -e %ld -g
```

### Links to On-Line Debugger Manuals

1. **gdb:** [www.gnu.org/manual/gdb-5.1.1/gdb.html](http://www.gnu.org/manual/gdb-5.1.1/gdb.html)
2. **dbx:**
  - a. Sun: <http://docs.sun.com/db/doc/805-4948?q=DBX>
  - b. IBM: [http://publib16.boulder.ibm.com/pseries/en\\_US/cmds/aixcmds2/dbx.htm](http://publib16.boulder.ibm.com/pseries/en_US/cmds/aixcmds2/dbx.htm)
3. **adb:** HP: <http://docs.hp.com/hpux/onlinedocs/B2355-90680/00/00/8-con.html>

If none of these hints help direct you towards a solution or an identifier in your application, then you should contact BEA Customer Support for further diagnosis. You can open a case with a valid support contract by logging in at <http://support.bea.com/login.jsp>.

A new tool for MX professional developers and designers...



### ADVERTISE

Contact: Robyn Forma  
robyn@sys-con.com  
(201) 802-3022  
for details on rates  
and programs

### SUBSCRIBE

[www.sys-con.com/  
mx/subscription.cfm](http://www.sys-con.com/mx/subscription.cfm)  
1 (888) 303-5282



**MX**  
developer's journal







FROM THE OFFICE OF THE CTO

# Web Services Security Progress Report

**MOVING FORWARD - AND ON SCHEDULE**

For the past several years there has been widespread agreement that the adoption of Web services for production applications will be limited, particularly for B2B transactions, until standardized security mechanisms, designed specifically for Web services, become available. While some applications can be adequately protected using the familiar SSL and TLS security protocols, their limitations make them unsatisfactory for many others.



BY HAL LOCKHART

For these reasons work began more than 18 months ago to standardize new security mechanisms specifically designed for the Web services environment. This article summarizes the work so far, describes what user organizations can expect in the near term, and discusses some of the missing pieces that are yet to come.

In September of 2002 the Web Services Security Technical Committee (WSS TC) was formed at OASIS to standardize mechanisms for protecting SOAP messages. The starting point for this work was a document previously published by IBM, Microsoft, RSA, and VeriSign entitled "WS-Security." The primary features of this specification are methods to digitally sign and encrypt portions of SOAP messages and methods to include in the messages the keys and identities associated with these signature and encryption operations.

The Web Services Security (WSS) specifications mostly define how existing XML security mechanisms should be applied in a SOAP message environment. For example, they reuse XML Digital Signature and XML Encryption (both W3C Recommendations). One of the key concepts in WSS is a token. A token is a data structure that associates an identity with a password or the key

used to digitally sign or encrypt data. WSS defines a specific type of token, called a Username token, which is used with password authentication. The Username token includes the user's name and optionally a password (in the clear or hashed) that can be authenticated by the receiver, as well as other features to prevent replay.

While WSS had to invent a Username token there are already standards for many other types of tokens. WSS defines a Binary token, which is just a wrapper for tokens that are not in XML format, such as X.509 Certificates and Kerberos Tickets. Tokens such as SAML Assertions and XrML Licenses, which are XML, are inserted as-is. The ability to support many different token types allows WSS to be used in a wide variety of environments.

WSS also defines a Security Token Reference that can be used in place of an actual token to refer to a token that appears elsewhere in the message or at some external location. There is also a Timestamp element that allows the sender to specify a date/time at which a message was created or should expire.

It is important to understand both the features and limitations of WSS. First, it deliberately does not define a secure protocol the way Kerberos and SSL/TLS do. It merely specifies a number of mechanisms that may be combined in various ways. The problem is that the security properties of any message exchange depend on every aspect of the messages. It is not uncommon for a small change to have a quite unexpected effect.

The flexibility of the specification also creates the possibility of lack of interoperability between different vendors' products. The initial testing has been largely successful, but some incompatibili-

## AUTHOR BIO

Hal Lockhart works for BEA Systems in the Office of the CTO, representing BEA on committees such as Web Services Security, SAML and XACML (of which he is the cochair) relating to security and management standards.

## CONTACT...

hlockhar@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

# LOOK FOR YOUR FREE...

>Linux >Java >Web Services >.NET >XML >Wireless >Storage >Security

The Premier Resource for Today's Corporate & IT Decision Makers

VOL: 1 ISSUE: 1 SPRING 2004

# IT solutions

>GUIDE

WWW.SYS-CON.COM/IT

## TECHNOLOGIES YOU NEED NOW!

How to Manage Your Ideas Using Today's i-Technologies

- > Delivering Software as Service
- > Leveraging Linux/Open Source
- > Moving to a Service-Oriented Architecture
- > Desktop Software: Migrating from Server to Client
- > Using Developer Tools to Drive Cost Out of Software
- > Application Integration
- > Storage & Security

Reaching 135,000 IT Decision Makers

Coming this **SPRING!**





ties have been discovered and corrected. The WS-I Basic Security Profile (discussed later) will further strengthen interoperability by reducing some of the options and clarifying some of the processing rules.

WSS does not address the problem of the description of the security mechanisms and capabilities used by a particular Web service. There are no WSDL features defined. The intended solution to this issue is the use of Policy advertisement mechanisms. A draft set of WS-Policy specifications has been published by BEA Systems, IBM, Microsoft, and SAP; and includes security policy. These specifications are being worked on intensively and will be submitted to a standards organization later this year. However, at the moment information about required signatures and tokens, for example, must be exchanged by some unspecified method, such as a phone call.

Another limitation to WSS is that most of the more complex Web services scenarios are quite speculative today. Most authorities agree that the use of Intermediaries is an important and powerful feature of SOAP. However, there is little agreement on precisely how these Intermediaries will behave and what their security requirements will be. Until more complex scenarios have been constructed and their security properties analyzed, this will remain a murky area. This means there will be plenty of scope for both interoperability problems and security weaknesses.

The OASIS TC made numerous changes to the original specification, improving it overall. Some features were added and some were cut. Language was tightened and clarified. The most noticeable change was to split the specification into several documents. A core specification describes the various mechanisms and a series of Token Profiles describe each Token type. It was decided to focus on the core specification and the Username and X.509 Profiles as the first set of specifications to complete. In the summer of 2003, a dozen or so vendors participated in interoperability testing of software based on the then current specifications. In most cases this was prototype code built on existing products. The testing identified some areas that needed clarification as well as some common errors to be avoided.

In September 2003, the TC approved the three specifications as OASIS Committee Drafts. As specified by the OASIS Process, this was followed by a 30-day public review. The comments received from this review, including an extensive set from the W3C XML Protocol Working Group and the WS-I Basic Security Profile Working Group, led to further changes in the specifications. Finally

in January 2004 the modified specifications were re-approved as Committee Drafts and submitted to the OASIS membership for approval as an OASIS Standard. This process has not been completed at the time this is being written, but normally takes about 60 days.

Work has begun to complete the SAML Token Profile, the XrML Token Profile, and the Kerberos Token Profile. These will probably be completed, in that order, during 2004. Several other specifications have been proposed, including a Minimal Profile, intended for use in devices like PDAs and cell phones; a Receipt Profile; and a Biometric Token profile. It is unclear whether any or all of these will be approved by the TC.

In March of 2003 WS-I chartered a Basic Security Profile (BSP) Working Group that was given two deliverables. The first was a set of usage scenarios. The reason for this was that it was observed that many existing Web services usage scenarios contained information that was irrelevant to security considerations and at the same time lacked critical details needed to select security mechanisms. The intention is to review this document widely both within and outside WS-I to get general agreement that the proper set of security requirements is being addressed. The usage scenarios will also serve the purpose of providing background to non-specialists on the security objectives and mechanisms relevant to Web services.

The usage scenarios document was nearly complete at the time of this writing. It will most likely be approved for release in early February 2004. The document describes a series of Security Challenges, which are essentially requirements to be achieved, a set of Threats that might prevent the challenges from being met, and the Security Mechanisms available at both the SOAP layer (WSS) and the transport layer (SSL/TLS).

Finally, the document lists three scenarios, originally developed by the WS-I Sample Applications Group: one-way, synchronous request/response, and basic callback. These are extended to include zero or more intermediaries in each case. Each is associated with a set of Security Challenges, which in turn are cross-referenced to the Threats and Mechanisms. The result is a concise summary of the problem space and the solutions most likely to be used.


The other deliverable of the BSP is a profile of the WSS specifications as well as SSL/TLS. This work is not as far along, but so far it is following the model of the WS-I Basic Profile – of which it is an extension – in defining constraints, reducing or eliminating options available in the profiled specifications. The first version of this document, covering the initial three WSS specifications is due nine months after they were first voted as OASIS Committee Drafts – June 2004.

## Conclusion

In summary, considerable progress has been made in developing Web Services Security mechanisms, but we still have some way to go before the work is complete. On the positive side:

- A number of products will provide interoperability and effective security in simple, common situations in 2004.
- The WS-I BSP will provide useful guidance in applying them.
- Users will see benefits not available in SSL/TLS.

On the negative side:

- Complex scenarios will likely have security holes initially.
- Parts of Web services, such as intermediary behavior, are not yet well enough understood to secure.
- Advanced features, such as policy discovery, will not be standardized for some time. 

# SAVE 16% OFF

**COLDFUSION** Developer's Journal

12 Issues for **\$89<sup>99</sup>**

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

That's a savings of **\$17<sup>99</sup>** off the annual newsstand rate. Visit our site at [www.sys-con.com/coldfusion/](http://www.sys-con.com/coldfusion/) or call **1-800-303-5282** and subscribe today!

# THE WORLD'S LEADING INDEPENDENT WEBLOGIC DEVELOPER RESOURCE

Helping you enable intercompany collaboration on a global scale

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

Now in More Than 5,000 Bookstores Worldwide – Subscribe **NOW!**

Go Online & Subscribe **Today!**

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

**SPECIAL OFFER!**  
**SAVE \$31\***  
 OFFER SUBJECT TO CHANGE WITHOUT NOTICE

**Using an Implementation Model to Identify Packaging Issues**

**CLASSLOADER ARCHITECTURE IN WEBLOGIC SERVER**

BY ANDY WINNELL

Most Java developers are familiar with the concept of classloaders. In the context of WebLogic, an application is usually an EAR file, but it could also be a WAR or an EAR file. When WebLogic creates a classloader for an application, it provides an effective supervisor of the application – they can't access each other's classes due to the classloader delegation.

In WebLogic, the application classloader loads the classes explicitly associated with the EAR file and the classes in the EAR JAR files. WebLogic also uses classloaders to load classes for the WebLogic Server.

Classloader architecture in WebLogic Server is a hierarchy of classloaders. The bootstrap classloader is the first in a sequence – the ancestor of all other classloaders. It's responsible for loading all the Java standard classes as well as the classes in the JAR packages. The bootstrap classloader loads all the JARs in the EAR's classpath. The system classloader loads all the JARs in the classpath (bootstrap) load of the classes in the development environment.

Application framework dependencies – loading strategies.

This article explains how to use the Implementation Model to identify WebLogic Server (WLS) packaging issues.

**Packaging Issues**

The classloader architecture changed with WLS 6. This affected a number of new features, including support for EAR files and a much chosen deployment model that supports hot deployment and hot redeployment.

**CLASSLOADER BACKGROUND**

A classloader allows a Java application to load new classes at runtime. Within a Java application, there is a hierarchy of classloaders. The bootstrap classloader is the first in a sequence – the ancestor of all other classloaders. It's responsible for loading all the Java standard classes as well as the classes in the JAR packages. The bootstrap classloader loads all the JARs in the EAR's classpath. The system classloader loads all the JARs in the classpath (bootstrap) load of the classes in the development environment.

**Standard Java classloader hierarchy**

**Using the Implementation Model to Identify Packaging Dependencies**

In WebLogic, the Implementation Model is composed of the structure of the physical directories and files that make up the application for implementation purposes – a collection of components. This is an important part of any development process. It's critical to have a cohesive, structured Java package structure. Using the Implementation Model provides an excellent way to identify packaging issues. It can be used to identify additional issues, such as component development dependencies and dependencies. For example, the Implementation Model can be used to help identify the packaging of EARs.

If you use the Implementation Model as a packaging tool, you can use it to identify classloader issues (found in our case, Rational Rose helps with the automatic tracing of dependencies). If you use the Implementation Model as a packaging tool, you can use it to identify classloader issues (found in our case, Rational Rose helps with the automatic tracing of dependencies). If you use the Implementation Model as a packaging tool, you can use it to identify classloader issues (found in our case, Rational Rose helps with the automatic tracing of dependencies).

Given the dependency shown in Figure 2, we can produce an EAR packaging diagram that reflects the same dependency (see Figure 3). Now a design decision should be considered during the

**WebLogic Developer's Journal**

**A DYNAMIC SYSTEM FOR WEB APPLICATIONS**

Large-Scale Financial Apps & Service-Oriented Architecture

Advanced JMS Design Patterns for WebLogic Server Environments

Implementing WebLogic GL

**WebLogic Developer's Journal**

**A CONVERSATION WITH ADAM BOSWORTH**

Good News for WebLogic

Recovering from an Invalid System Password

Big Impact: App Server Deployment

**WebLogic Developer's Journal**

**BUILDING BETTER BRIDGES**

Transaction Management

System Administrator

WebLogic Server on the Mainframe

Power of EJB QL Subqueries

WebLogic Web Services Security

Understanding the Advisor Framework

WebLogic Web Services Security

Monitoring Web Applications

Performance Session Persistence





# A Safe Architecture Framework

## LEVERAGE THE SECURITY FEATURES OF BEA WEBLOGIC 8.1



BY ASHLEY BYRD & GIRISH GUPTA

### AUTHOR BIOS...

Ashley Byrd is a senior software engineer with the ICF Consulting software engineering team. He has 15 years of experience in application development, the last 7 in full life-cycle, object-oriented Internet systems primarily for the financial services industry.

Girish Gupte works for BEA Professional Services. He has extensive experience in architecting and implementing mission-critical, high-availability applications. He performs on-site mentoring, architecture, design, and troubleshooting for BEA customers.

### CONTACT...

byrd@scra.org,  
grgupte@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

**E**MALL is a procurement and collaborative commerce portal for the U.S. Department of Defense and based on J2EE and WebLogic Platform 8.1. One core component of EMALL is a rule engine based on WebLogic Platform 8.1.

EMALL integrates with Web applications, based on .NET and J2EE technology, through which DOD personnel submit and track orders. The rule engine validates these orders and submits requisitions to various defense suppliers. All of these systems interoperate, using ebXML 2.0, under stringent security requirements. In our previous article (*WLDJ*, Vol. 3, issue 1), we discussed the overall EMALL architecture. In this article, we will describe how we leveraged security aspects of the BEA WebLogic Platform to implement security solutions.

### Security Requirements for EMALL

We will review and discuss some critical security requirements and analyze the threats and risks that this system must mitigate. EMALL handles sensitive information. Only authorized personnel with proper security privileges can perform tasks such as browsing the catalogs of various suppliers, reading the technical specifications of various products, submitting orders, and tracking order status. Accountability is also a critical requirement for this system.

Let's look at some of the important security threats:

1. **Man-in-middle attack:** The EMALL system spans a large geographic area, and several networks, including some public segments.
2. **Message tampering:** Attacker can try to modify the content of a message.
3. **Impersonation attack:** The EMALL system includes several business and government entities. An attacker can try to impersonate a legitimate entity.
4. **Trojan horse:** Several DOD personnel have access to EMALL. There is a threat from one or more individuals trying to compromise the system.

5. **Security breach at a subsystem:** EMALL includes several subsystems, some located at a DOD supplier's networks. There could be a security breach at one such subsystem that can affect the entire EMALL.

### Design Goals

We identified the following design goals to mitigate these risk factors:

#### Data Encryption

All traffic between various entities must be encrypted using strong encryption to prevent a man-in-middle attack.

#### Two-Way Authentication

Each entity participating in EMALL must identify itself using a digital certificate. Each data-exchange must enforce two-way-certificate verification to prevent an impersonation attack.

#### Control the Role of Each Subsystem within EMALL

Each subsystem entity provides specific services to other entities, and can invoke specific services from other entities. This would minimize the risk in case one of the subsystems is compromised.

#### Access Control

Each subsystem would implement role-based security for their internal users. These mechanisms would be different for each subsystem. The rule engine will receive the user ID information within each message that it receives from the external subsystem when it receives requests. It will in turn propagate proper credentials to the subsystems when it invokes services.

#### Digital Signatures

Each message will be digitally signed to prevent unauthorized modification.

#### Audit Trails

Each message will be tracked using audit trails to identify a security breach and improve

# The World's Leading Java Resource!

## JAVA DEVELOPER'S JOURNAL

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

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



ANNUAL COVER PRICE	
<del>\$71.88</del>	
<b>YOU PAY</b>	<b>\$49.99</b>
<b>YOU SAVE</b>	<b>30%</b> Off the annual cover price

Sign up **ONLINE** at [www.javadevelopersjournal.com](http://www.javadevelopersjournal.com)

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

## WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
BEA Systems	<a href="http://dev2dev.bea.com/medulla">http://dev2dev.bea.com/medulla</a>	800-817-4BEA	2
ColdFusion Developer's Journal	<a href="http://www.sys-con.com">www.sys-con.com</a>	888-303-5282	32
Crystal Decisions	<a href="http://www.businessobjects.com/v10/059">www.businessobjects.com/v10/059</a>	800-877-2340	17
Cyanea	<a href="http://www.cyanea.com/wldj/underpar.html">www.cyanea.com/wldj/underpar.html</a>	877-CYANEA8	9
H&W Computer	<a href="http://www.hwcs.com/wldj1.asp">www.hwcs.com/wldj1.asp</a>	800-338-6692	52
HP	<a href="http://www.hp.com/plus_fedex">http://www.hp.com/plus_fedex</a>	800-752-0900	11
IT Solutions Guide	<a href="http://www.sys-con.com">www.sys-con.com</a>	201-802-3021	31
Java Developer's Journal	<a href="http://www.sys-con.com">www.sys-con.com</a>	888-303-5282	27, 35
Mercury Interactive	<a href="http://www.mercuryinteractive.com/optimizej2ee">www.mercuryinteractive.com/optimizej2ee</a>	800-831-8911	3
MX Developer's Journal	<a href="http://www.sys-con.com/mx/subscription.cfm">www.sys-con.com/mx/subscription.cfm</a>	888-303-5282	29
NetIQ	<a href="http://www.netiq.com/solutions/web">www.netiq.com/solutions/web</a>	408-856-3000	51
Quest Software	<a href="http://java.quest.com/jcsv/wldj">http://java.quest.com/jcsv/wldj</a>	800-663-4723	25
Quest Software	<a href="http://java.quest.com/jprobe/wldj">http://java.quest.com/jprobe/wldj</a>	800-663-4723	19
SandCherry	<a href="http://www.sandcherry.com">www.sandcherry.com</a>	720-562-4500	23
SalesForce.com	<a href="http://www.sforce.com">www.sforce.com</a>	800-NO SOFTWARE	15
SYS-CON Publications	<a href="http://www.sys-con.com">www.sys-con.com</a>	888-303-5282	37, 44-45, 49
SYS-CON Reprints	<a href="http://www.sys-con.com">www.sys-con.com</a>	201-802-3026	43
WebLogic Developer's Journal	<a href="http://www.weblogicdevelopersjournal.com">www.weblogicdevelopersjournal.com</a>	888-303-5282	33
Web Services Journal	<a href="http://www.sys-con.com">www.sys-con.com</a>	888-303-5282	41
Wily Technology	<a href="http://www.wilytech.com">www.wilytech.com</a>	888-GET-WILY	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.

# LOOK WHAT'S COMING NEXT MONTH

## Considering MySQL? Read On...

The article we should have seen before: we explore using MySQL as the database engine, where the application is developed using BEA WebLogic Workshop 8.1 and deployed to WebLogic Server 8.1. Using a prototypical J2EE blueprint architecture the impact of using MySQL is evaluated from all aspects and various pitfalls are uncovered, logically approached, and methodically solved.

## XA: You Need It More Often Than You Think

Most developers have heard of XA and two-phase commit. Most developers also think to themselves, "I don't need to update two databases, so I don't need to use XA". Wrong! XA is for any time you need to update two separate resources simultaneously. This article will discuss this often overlooked requirement and provide specific tips on how to XA enable a small application on BEA WebLogic Server 8.1.

## An Architectural Blueprint for Merging SOA and BPM

SOA has become the single most important theme in software engineering. At the same time, BPM is making a strong comeback as a key enabler for modeling and operating the new agile enterprise. Although there is a clear indication of the emergence of both trends, no clear and prevailing thought exists about the convergence of the two. Are they complementary notations or do they overlap; how do I use them together; and if so, is there any additional benefit? Furthermore, why is the third wave of BPM poised to succeed when the BP re-engineering of the late 80's failed?

## "HTTP Session Replication Failure" Issues

There are several reasons for why session replication fails. We will look at ways to diagnose the issue, possible reasons for it to occur, and ways to address the problem.

## Entity EJB Development in WebLogic Server 8.1

EJB development in BEA WebLogic Server 8.1 consists of EJB design, creation, and deployment. This tutorial discusses the design and development of entity beans.





accountability. These trails would include the following information:

- Content of the message
- Digital signature of the message
- Timestamp
- User ID that initiated the message

Recipient must acknowledge each message.

## BEA WebLogic Platform Components

Figure 1 represents various parts of the WebLogic Platform related to security infrastructure.

### HTTPS and SSL Features

Some important aspects of SSL support provided by BEA WebLogic Server are relevant to EMALL.

### Strong Encryption

BEA WebLogic Server supports 1024-bit certificates and 128-bit bulk data encryption.

### Support for Asymmetric Key Cryptography:

WebLogic Server supports the RSA algorithm, including support for digital signatures.

### Two-Way Certificate Verification

With this feature enabled, both the client and BEA WebLogic Server must present SSL certificates before a connection is established. WebLogic Server checks the following before establishing a connection:

- The certificate is issued by a trusted cer-

tifying authority

- The IP address/Host-Name in the certificate matches the client's IP address

## WebLogic Security Provider Framework

BEA WebLogic Server supports J2EE declarative security including features such as user IDs, roles, access control lists, auditing, security events, etc. To implementing these features, WebLogic Server includes a security framework that can be extended using security provider modules. WebLogic Server comes with out-of-box implementations of these modules. You can plug your own custom implementations into the framework. Some of the modules relevant to EMALL include:

1. **Authentication providers:** Authenticate users by verifying credentials supplied by the user. This could be a user ID/password, or could be certificate-based.
2. **WLSUser:** Interface for the "Principal" object that holds security information representing a user.
3. **Identity assertion providers:** Can use security tokens such as CSIV2, SAML, Kerberos, and Microsoft Passport.
4. **Principal validation providers:** Handle signing, encrypting, and verification of security information across multiple WebLogic domains. It prevents malicious tampering of security information during RMI calls between WebLogic Servers.
5. **Authorization providers:** Enforce securi-

ty policies based on roles, user IDs, groups, etc., for all resources within BEA WebLogic Server.

6. **Role-mapping providers:** Allow dynamic role association.
7. **Auditing providers:** Tracks security events such as access to specific resources. It can record information such as user IDs, timestamps, credentials provided, and resources accessed. This information can be put into a database or a file. It can also trigger specific actions such as paging security personnel in response to security events.

Our planned architecture includes custom security provider modules to implement the following features of EMALL.

1. Create a custom implementation class for WLSUser to propagate additional information such as a user's rank, security-clearance level, digital signatures, department etc.
2. Develop a custom authentication provider that will interact with a central security database and authenticate the user. It will then extract all the necessary information and instantiate the custom WLSUser class.
3. Develop a custom identity-assertion provider to implement single sign-on.
4. Develop a custom authorization provider and custom role-mapping provider to enforce access control for secure information. This implementation will use the information (such as rank, clearance level, etc.) stored in a custom WLSUser implementation.
5. Implement a custom audit provider to record activities. The information in the custom WLSUser object will be tracked in the audit trails.

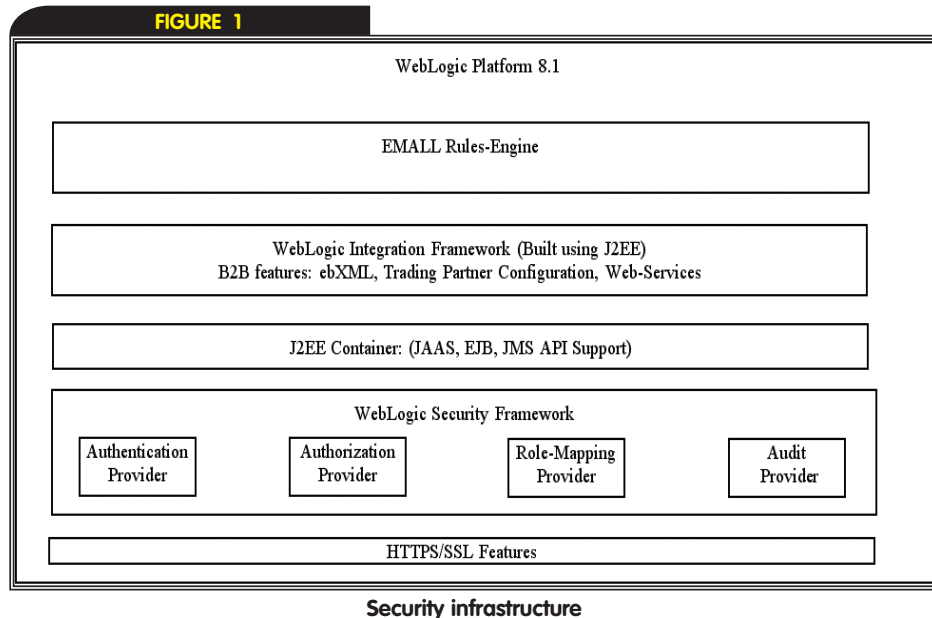
This framework will be used by the rule engine and all other J2EE applications deployed on the BEA WebLogic Platform.

Until now we've discussed security features from the core BEA WebLogic Server. Now let's discuss the B2B security features provided by WebLogic Integration.

## Using J2EE Declarative Security Constraints

The "rule engine" application includes several resources, such as URLs in Web applications and EJB methods that need to be protected by the Security Framework. We will use the J2EE security constraints

FIGURE 1



defined in web.xml and ejb-jar.xml files for this purpose.

## Trading Partner Configuration

BEA WebLogic Integration allows us to manage the relationships with other business entities by configuring trading partners. Some of the important features used in the EMALL security architecture include:

1. Associating a digital certificate with a trading partner. These certificates will be used for encryption and signing of ebXML messages.
2. Defining services offered by local and remote trading partners.
3. Defining the protocol bindings for each service profile. This includes business protocol ebXML 1.0, ebXML 2.0, or RosettaNet and Transport Protocol (HTTP 1.0, HTTP 1.1 or HTTPS 1.1).
4. Specifying roles for each "conversation" or Service invocation.
5. Easy-to-use, centralized admin console interface for managing the trading partner configuration.

Using these features, we can manage the trading-partner configurations with all vendors participating in EMALL from the "Rule-Engine" application.

## Implementation Notes

Several important points regarding implementation of the rule engine should be considered.

The ebXML participant process (acceptCart.jpd) receives the "Shopping Cart" document, which includes the user-metadata along with the other parameters. The code in this JPD file will extract this information from the XML payload and invoke a specific stateless EJB.

The BEA WebLogic Server Security framework will invoke the custom authentication provider, which instantiates a custom object implementing the weblogic.security.spi.WLSUser interface. The actual implementation class for this object also has additional fields to hold custom security data such as rank, digital signatures, etc.

In the stateless EJB method, we obtain the security context and the JAAS principal. This object is actually the custom implementation WLSUser created by our authentication provider. This object is cast to its original type, and then the fields containing

additional custom data are populated.

When the JPD code invokes other functionality in the rule engine, these data fields are checked by the role-mapping provider, authorization provider, and so on.

When each protected resource is accessed, the custom audit provider is invoked by the WebLogic Security Framework. This audit provider extracts these data fields and logs them in the audit repository.

## Nonrepudiation


The following architecture features support nonrepudiation in EMALL:

1. **Extensive audit trails:** The custom audit provider records critical information including user-metadata, digital signatures, timestamps, etc., in the audit trails.
2. **ebXML supports digital signatures:** Only the entity that owns the private-key for the digital certificate can sign the messages. This signature is irrefutable proof that the message was sent by the proper entity. In addition, once the message is received, ebXML allows for acknowledgement of the message, which is a proof that the recipient entity received the message. Together, these two (message signature and acknowledgment) are proof that the transaction is binding on both parties.

## Conclusion

The EMALL system has stringent security requirements. Using the security features provided by BEA WebLogic Platform 8.1, we were able to develop an architecture framework that meets these goals. The WebLogic Security Provider Framework, support for SSL, and use of ebXML are the most important factors in our security architecture.

## Acknowledgments

The authors would like to gratefully acknowledge the support of Dr. Bill Freeman, director of research and development for the Integrated Solutions Group of the South Carolina research Authority. Dr Freeman is also the project manager for SCRA's work on the DOD EMALL. BEA and ICF provide engineering services to SCRA. We would also like to thank Ricardo Valenzuela, Madhavan Rangarao, Dmitry Dimov, and Komal Mangtani from the BEA engineering team, who provided technical support to help us achieve our goals. 

# SUBSCRIBE TO THE FINEST TECHNICAL JOURNALS IN THE WORLD...



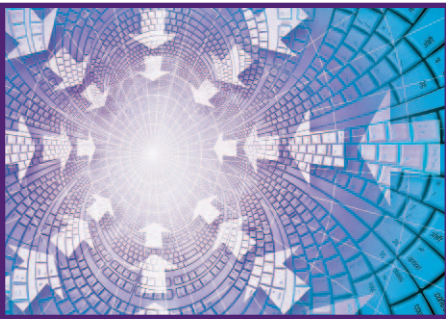
## IT'S JUST A CLICK AWAY!



 SYS-CON MEDIA

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





# Modernizing Legacy Systems, PART 2

## INTEGRATION AS THE CRITICAL STEP

**M**y previous article (WLDJ, Vol. 3, issue 2) introduced a “recipe” to modernize legacy systems with the BEA WebLogic Platform by using a step-by-step approach. In many ways, that article introduced a “modernization process”; in this one I will delve into the details of cross-platform integration and architectural design. I will also use the same recipe-based approach to illustrate how to build a scalable, robust, and maintainable modernized platform around WebLogic. In the process I will present some fundamental J2EE patterns that you will be able to reuse in your own design.



BY ANWAR LUDIN

### AUTHOR BIO

Anwar Ludin specializes in service-oriented architectures for the financial sector. He currently works as an independent consultant for financial institutions in Switzerland, where he helps design J2EE architectures based on the BEA WebLogic Platform.

### CONTACT...

[anwar.ludin@agilethinker.com](mailto:anwar.ludin@agilethinker.com)

As I mentioned in my previous article, integration is a critical step in the entire process. Its goal is to provide the bridge between the legacy world and the WebLogic modernized platform. In many ways, the legacy system can be seen as a resource that we want to integrate with the WebLogic Platform. One of the goals of the modernization process is to shield the platform as much as possible from the legacy system so that the design of the modernized system can evolve relatively independently of it. Let’s look at the architectural best practices used for integrating both platforms and implementing the integration tier.

### Design a Layered Architecture

A best practice is to build J2EE applications using logical layers. In particular, higher-level layers leverage the services of lower-level layers. The idea of a layered architecture is to organize the modernized platform according to generality. The topmost layer, the most application specific

in a layered architecture, contains application systems that can be considered as a coherent set of use cases available to end users. The layer directly under the application systems, the business-specific layer, contains reusable components specific to a business. For example, for a financial institution this layer could contain reusable components representing accounts, customers, financial statements and reports, etc. The application layer could then contain a Portfolio management package that leverages the underlying reusable financial components. Finally, the layer under the business-specific components consists of infrastructure components such as logging frameworks and utility classes. Some of the characteristics of a layered architecture are:

- The topmost layer is the most application specific and the bottom one the most generic.
- Each layer exposes its services to the layer directly on top through a set of interfaces and facades. The opposite is not true; a top layer does not expose its services to the underlying one. However, components located “horizontally” in the same layer can leverage each other’s services and public interfaces.
- Layering addresses module dependencies and therefore minimizes build and packaging time.
- Data is passed between layers by using Transfer Objects (see “Implement Transfer Objects”).

Layers should not be confused with tiers. A logical layer might span one or several tiers. Layering is really a logical way of organizing your architecture and exposing service interfaces through facades.

Figure 1, an example of a layered architecture, illustrates the case of a banking system as a set of package dependencies.



## Implement Transfer Objects

One of the first steps in designing the modernized platform is to identify transfer objects. As you might recall from my last article, the legacy system is most probably implemented by using a procedural language such as COBOL. Therefore, one of our initial tasks will be to encapsulate the legacy system in order to provide a Java object view of it to the BEA WebLogic Platform. Data access objects (see next section) will then encapsulate the legacy services and transfer objects will be used to transfer multiple data elements between the legacy system and the different tiers of the WebLogic platform. So what is a transfer object precisely? Basically, a transfer object is an instance of a plain Java class, used as a container for data elements in order to optimize data transfer between tiers. It must therefore be serializable and is passed by value to the client. A transfer object usually simply provides getter and setter methods to its members. For example, in a banking application we might have transfer objects such as AccountTO, CustomerTO, and CreditCardTO that contain relevant information about accounts, customers, and credit cards. An important point to emphasize is that transfer objects can be used effectively to build a domain model of the legacy system in Java. It is therefore very useful to spend some time identifying the legacy system's business entities. As a good rule of thumb you can map each identified entity in the legacy system to a corresponding transfer object. It is also worth mentioning that transfer objects are used in every tier of your application, from the resource to the presentation tier. It is therefore very important to design and implement your transfer objects in such a way that they are reusable and correspond to the business domain as much as possible. Finally, transfer objects were called value objects in the first edition of the book *Core J2EE Patterns*. However, the term value object was misleading. It was used well before the J2EE patterns group for "a small simple object, like money or a date range, whose equality isn't based on identity." Also, transfer objects are equivalently called data transfer objects but I have kept the J2EE patterns group terminology in this article for the sake of clarity.

## Implement Data Access Objects

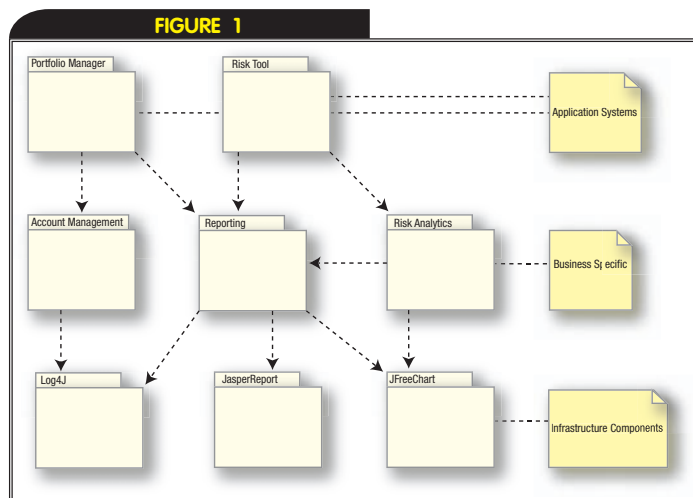
Data access objects (DAO) are used to completely encapsulate the legacy system and provide a simple and uniform interface to it. A DAO permits the following fundamental operations: create, read, update, delete (CRUD), and find. The parameters passed to and returned from DAOs are transfer objects. In our case the legacy system will play the role of a data source and we will use DAOs to retrieve or update data. Usually we will build one DAO per transfer object. Let's review the interface implemented by a DAO:

- **create(aTransferObject: Transfer Object): PrimaryKey.** The create/insert method effectively creates a new business entity in the legacy system and returns its primary key. For example, in a banking system the create method could be used to open a new account according to the information contained in the AccountTO and return the account's primary key, which should also be a serializable object.
- **read(aPrimaryKey: PrimaryKey):** Returns the transfer object corresponding to the primary key.
- **update(aTransferObject: Transfer Object):** Updates the legacy business entity corresponding to this transfer object's primary key with the new data contained in it.
- **delete(aPrimaryKey: PrimaryKey):** Deletes the business entity

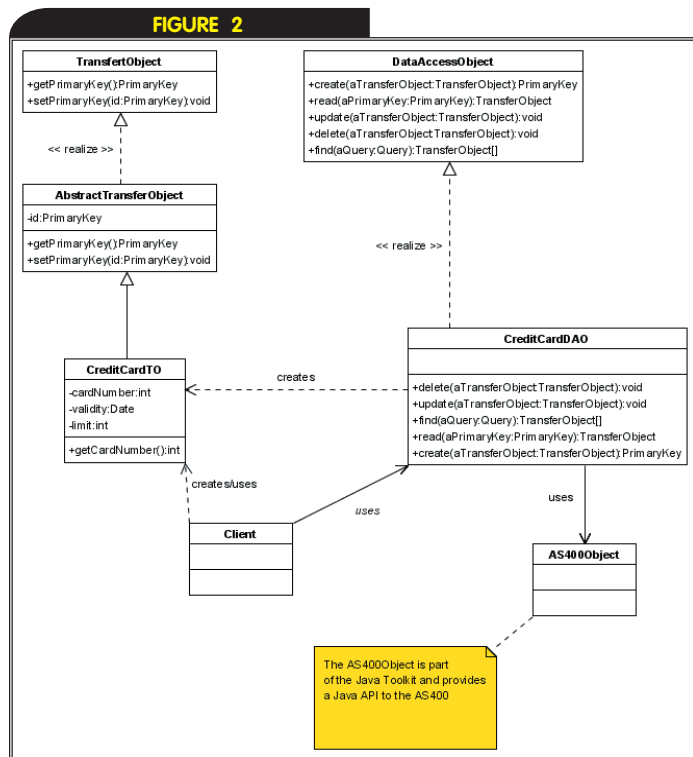
in the legacy system corresponding to this primary key.

- **find(aQuery: QueryOperator): TransferObject[]:** Returns all business entities corresponding to the query operator as a list of transfer objects.

Now that we have specified the DAO's interface, how do we implement it? As we will see, every method in the DAO's interface will usually correspond to one or several COBOL service programs running on the legacy system. A COBOL Service Program (SP) can be considered a special kind of COBOL module that can be accessed from Java. For example, consider a banking system. Figure 2 illustrates the relationship between a CreditCardTO trans-



Layered architecture

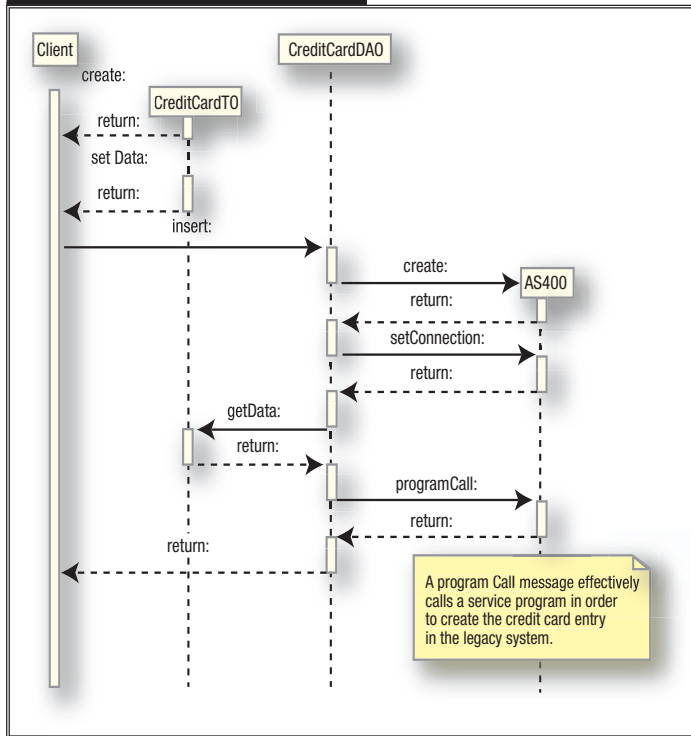


Banking system class diagram





FIGURE 3



Sequence diagram

fer object and its CreditCardDAO DAO in a class diagram.

Listing 1 shows a possible implementation of the CreditCardDAO DAO using the Java Toolbox for the AS/400. Note that I have shown only the create method.

Note that the DAO pattern has the advantage of completely shielding our components hosted on BEA WebLogic from the legacy system. In other words, we can modify the data access logic during the modernization effort and even completely replace it with entity beans without breaking any of our components relying on the DAOs. We could use a Factory method returning the appropriate DAO implementation.

Finally, Figure 3 shows the interactions between a client such as a session bean and a CreditCardDAO.

### A Note about Distributed Transactions

The details of distributed transactions between WebLogic and the legacy system are beyond the scope of this article. One of the major difficulties is to achieve a rollback once a transaction has been committed. In our case, the operations available on the legacy system are exposed as COBOL SPs and a rollback is no longer possible once an SP has been called. Also, the legacy system might not provide any transactional services at all and can only be considered as a simple data source. We could, however, achieve the same result as a rollback by doing a “compensating” data update. This simulates the effect of a rollback on the legacy system by doing, after an operation, a corresponding opposite operation in order to restore the system in the same state prior to the initial operation. Consider once again the legacy banking system:

- **Begin transaction:** T in the WebLogic business tier.
- **Update operation:** Withdraw amount A from customer account. This actually corresponds to a COBOL SP call with an immediate commit on the legacy system.

- **Commit:** Perform commit operation in the WebLogic business tier. The operation has already been performed in the legacy system.
- **Rollback:** For the legacy system, simulate a rollback by crediting customer account with amount A.

Note that the Web Services Transaction specification takes exactly the same approach for long-lived transactions or business activities. Finally, if we consider the most generic form of the operation, where we have a mix of JTA resources and the nontransactional legacy system, we could implement a compensating transaction as illustrated in the following code snippet:

```

updateLegacyBankingSystem();
try {

    UserTransaction.begin();
    updateJTASupportedResource1();
    updateJTASupportedResource2();
    updateJTASupportedResource3();
    UserTransaction.commit();

}
catch (RollbackException ex) {
    undoUpdateLegacyBankingSystem();
}

```

### Implementing Use Cases

The modernization recipe introduced in the first article mentioned that use cases with added value should be identified and considered as the initial candidates for the modernization effort and re-hosted on WebLogic. Another important task is to determine to which layer of our architecture the use case belongs. For example, if the identified use case can be reused by several different applications or other use cases, then it should be implemented in the business-specific layer and exposed through a facade interface. However, if the use case is very application specific, then it should be implemented in the topmost application layer of our architecture.

### Conclusion

In this article I have presented some of the essential patterns for designing the modernized BEA WebLogic Platform's architecture. In order to integrate both legacy and modernized systems, I used the transfer object and DAO patterns. Transfer objects have been used as a basis for building a domain model of our legacy system and transfer data between it and the modernized WebLogic platform. The legacy system can also be completely encapsulated as a data source by using the Data Access Object pattern. Usually we will build one DAO per transfer object.

A layered architecture minimizes dependencies between components of the modernized WebLogic platform and promotes greater reuse. An important aspect of modernization is to identify value-added use cases of the legacy system and reimplement them in one of the layers of the modernized platform. If the use case is business specific and can be leveraged by other use-case implementations, then it should be located in the business-specific layer of our architecture and its interface exposed as a service facade. However, if the use case is application specific it should be located at the topmost application-specific layer of our architecture. Now that we have built the integration infrastructure of our modernized platform, I will show how to build the application- and business-specific layers of

our architecture using BEA WebLogic Workshop in my next article. I will put all of the pieces of the puzzle together and modernize a small banking system with the WebLogic Platform. We will also see how WebLogic Workshop takes a RAD approach towards designing the application.

## Reference

- Fowler, Martin (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley. ●

### Listing 1: CreditCardDAO DAO

```
public class CreditCardDAO implements DataAccessObject {

private String as400HostId = null;
private AS400 host;

CreditCardDAO(String as400HostId) {
    this.as400HostId = as400HostId;
    this.host = new AS400(as400HostId);
}

public Object create(TransferObject to) throws DAOCreateException {
    String primaryKey = "";
    if (to instanceof CreditCardTO) {
        CreditCardTO creditCardTO = (CreditCardTO)to;
        ProgramCallDocument pcml;
        // Return code for program call
        boolean rc = false;
        String msgId, msgText;
        try {
            pcml = new ProgramCallDocument(as400Host,
                "CreditCardPCML");
            pcml.setValue("CreditCard.cardNumber",
                new Long(creditCardTO.getCardNumber()));
            rc = pcml.callProgram("CreateCreditCard");

            if (rc == false) {
                AS400Message[] msgs =
                    pcml.getMessageList("CreateCreditCard");

                for (int m = 0; m < msgs.length; m++) {
                    msgId = msgs[m].getID();
                    msgText = msgs[m].getText();
                    // Perform Logging.
                }
                throw new DAOCreateException();

            }
            else {
                // Process the returned Data
                primaryKey =
                    (String) pcml.getValue("CreateCreditCard.primaryKey");
            }
        } catch (PcmlException e) {
            throw new CreateDAOException();
        }
        else {
            throw new DAOCreateException();
        }
    }
    return primaryKey;
}
}
```

# WebServices

.NET J2EE XML JOURNAL

LEARN WEB SERVICES. GET A NEW JOB !

## SUBSCRIBE TODAY TO THE WORLD'S LEADING WEB SERVICES RESOURCE

Get Up to Speed with the Fourth Wave  
in Software Development

- **Real-World Web Services:** XML's Killer App!
- How to Use **SOAP** in the Enterprise
- Demystifying **ebXML** for success
- **Authentication, Authorization, and Auditing**
- **BPM - Business Process Management**
- Latest Information on **Evolving Standards**
- Vital technology **insights** from the nation's leading Technologists
- Industry **Case Studies** and **Success Stories**
- Making the Most of **.NET**
- **Web Services Security**
- How to Develop and Market Your Web Services
- **EAI** and Application Integration Tips
- **The Marketplace:** Tools, Engines, and Servers
- Integrating **XML** in a Web Services Environment
- **Wireless:** Enable Your **WAP** Projects and Build **Wireless Applications** with Web Services!
- **Real-World UDDI**
- **Swing-Compliant** Web Services
- *and much, much more!*

The Best  
.**NET**  
Coverage  
Guaranteed!

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

Subscribe online at  
[www.wsj2.com](http://www.wsj2.com) or  
call 888 303-5252

*\*Offer subject to change without notice*



**SYS-CON  
MEDIA**

SYS-CON Media, the world's leading *i*-technology publisher of developer magazines and journals, brings you the most comprehensive coverage of Web services.





# How Loose is Your Coupling?



BY PETER HOLDITCH

### AUTHOR BIO

Peter Holditch joined BEA as a consultant in the Northern European Professional Services organization in September 1996. He now works as a presales architect in the UK. Peter has a degree in electronic and computer engineering from the University of Birmingham.

### CONTACT...

[peter.holditch@bea.com](mailto:peter.holditch@bea.com)

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

Whatever your innermost feelings about the < and > symbols, and however fondly you remember debugging network infrastructures with nothing more than a LAN sniffer and an uncanny ability to interpret 4K blocks of hex, it is fairly safe to say that Web services are here to stay. With the industry-wide support for the concept, and corresponding legions of emerging and released standards, they aren't going anywhere soon.

One raft of Web service standards that is currently a little closer to the dock than many is that discussing the concept of cohesions, or business transactions. On this raft float such passengers as OASIS' BTP, WS-Transaction, WS-Coordination, and a few others, all of which are trying to address the notion that multiple operations can be conceptually related, even if all the operations affect different back-end systems. Or to put it another way; transactions.

Many people start to feel more than a little queasy when they think of two-phase commits flowing in glorious XML over an HTTP wire, and perhaps this isn't unreasonable – none of these elements is exactly synonymous with lightning speed and efficiency in the millisecond order of magnitude – but to have that thought and decide transactional behavior and Web services don't mix is to put the cart before the horse. The point here isn't the fancy plumbing (let's face it, from a business perspective the point is seldom fancy plumbing) but the value attached to the ability to coordinate multiple independent applications in such a way that their whole is greater than the sum of their parts. After all, if a set of linked applications is no more functional than all the

same applications standing alone, the whole notion of Web services is starting to hemorrhage credibility faster than a steam-powered rocket booster in the space shuttle.

It is only a short leap from saying we need to link applications together – which Web services does, implicitly – to saying that from a business perspective we need to know how much of some composite transaction has completed (and in the limit, that we can not only say “do it!” but “undo it!” to our composite system, with some expectation that the “it” that we do and undo is a meaningful business operation with some level of assurance about all-or-nothing results, and all that other good stuff that transactions have given us for so long).

### How to Design for the Web Service World

So, given that you're about to design a new application (a rare luxury in this day and age, I confess) what should you do to make sure it will play nicely in the world of Web services (and thus guarantee that it has the respectable shelf life of a useful system rather than being just another here today, gone tomorrow mirage). Much has been written about putting Web service interfaces onto systems that are coarse-grained, loosely coupled, and asynchronous. Some systems have even been engineered in ways that conform to the spirit of these commandments, rather than just some skewed version of the letter, but not much mention has been made of transactional semantics. No doubt this is in large part due to the relative immaturity of industry standards in this area. But just because an implementation of a standard isn't out there doesn't mean that there aren't a set of “right things” to do when thinking about possible future participation in a transactional context for your

application. After all, loose coupling and the other mantras aren't exactly standards themselves, just best practice aspirations.

To try and arrive at another mantra, let us think about an architecture use case where some notion of transactions might be useful. Let's take an imaginary banking system. Someone hits the bank Web site, registers, and requests that an account be set up. Now, cards need to be produced, dispatched, received, and acknowledged; welcome packs need to be sent; security credentials need to be gathered; local branches need to be notified; and so on and so on. It is simple to imagine that all the systems responsible for these piece parts of the single business transaction "create new account" are independent. It is also clear that there is a complex web of relationships between them – when is the account considered to be created? When the card is acknowledged, when it's dispatched, when a credit check succeeds? The answer depends on the business rules. And let's think about the process of closing an account. The action we will need to take in this event will depend on what we have done to set it up – maybe we just failed a credit check and nothing else happened. Maybe we issued the cards, so we need to stop them, maybe... I could go on, but I might get a

access multiple resources, each of which might fail. And like the transaction manager, implementations of the standards – as they become available - will need a set of hooks into the applications they are coordinating to allow them to move the composite application between business-meaningful states.

### Yes, No, and Maybe

At a minimum, I would propose that a best practice would be to design all systems to have their operations come in threes – doItTentatively, doIt, and undoIt. And associate an externally supplied identifier of some sort with the tentative actions. That way, the composite system can put its toe into your application's water by creating a pending account (or whatever) and then allow it to be activated at some later time when the rest of the world (that loose coupling states you should not know or care about) decides that all is well (or deleted, if everything turned out dust for reasons outside your control). This clearly implies that you need to think through the semantics of pending operations from a business perspective too, which can only aid robustness.

Let's face it, with service-oriented architecture we're building communities of components, not monolithic applications. Systems architects these days are nearly as

“Just because an implementation of a standard isn't out there doesn't mean that there aren't a set of 'right things' to do when thinking about possible future participation in a transactional context for your application”

cramp in my typing fingers. All of this detail underlies the simple business transaction “close account.”

What we are beginning to see is that each component of an aggregate system has some set of states in its life cycle – at its simplest, nonexistent, pending, active – and the composite system has a set of states that is some matrix multiplication of all the components' possible states. This complexity starts to explain why it's so easy to design and implement a “new account” process flow with a tool like BEA WebLogic Integration, but so complex to build all the correct exception-handling logic.

In essence, all the emerging choreography standards aim to do for this complexity what the transaction manager did for the complexity of applications that need to

likely as computers themselves to think in binary. Try to imagine a human society where everything had an instant yes or no answer; how would negotiations happen? How many times a day do you use maybes to soften the yeses and nos that you know, ultimately, you will need to settle on when dealing with groups of people? We are building components that will interoperate like a society – they will increasingly need this kind of interface subtlety.

I claim that if you build in this kind of pattern you have a flying chance of integrating your services with others, whatever the eventual choreography standard that emerges says. In the meantime, writing compensating transactions using a process engine becomes much easier, so the pragmatists and the crystal ball gazers will all be happy. ●

# Once you're in it...



## ...reprint it!

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

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

REprints

SYSCON  
MEDIA



# SUBSCRIBE TODAY TO MULTIPLE

Go To [www.sys-con.com/suboffer.cfm](http://www.sys-con.com/suboffer.cfm)

and receive your **FREE CD Gift** Package VIA **Priority Mail**



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

**Pick the CDs to go with your Multi-Pack order**

- ▶ Pick one CD with your 3-Pack order
- ▶ Pick two CDs with your 6-Pack order
- ▶ Pick three CDs with your 9-Pack order

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

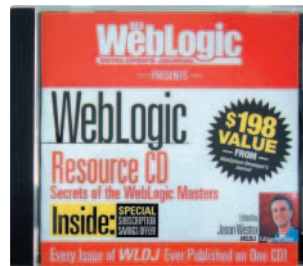
**Your order will be processed the same day!**



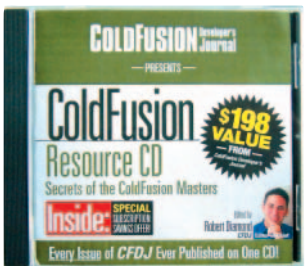
More than 1,400 Web services and Java articles on one CD! Edited by well-known editors-in-chief Sean Rhody and Alan Williamson, these articles are organized into more than 50 chapters on UDDI, distributed computing, e-business, applets, SOAP, and many other topics. Plus, editorials, interviews, tips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



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



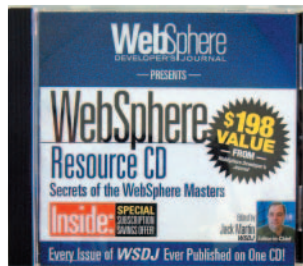
The most complete library of exclusive *WLDJ* articles ever assembled! More than 200 articles provide invaluable information on "everything WebLogic", including WebLogic Server, WebLogic Portal, WebLogic Platform, WebLogic Workshop, Web services, security, migration, integration, performance, training...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



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



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



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



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

**Subscribe Online Today**

# UPLE MAGAZINES ONLINE

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

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS



**3-Pack**  
Pick any 3 of our magazines and save up to **\$275<sup>00</sup>**  
Pay only \$175 for a 1 year subscription plus a **FREE CD**

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

**6-Pack**  
Pick any 6 of our magazines and save up to **\$350<sup>00</sup>**  
Pay only \$395 for a 1 year subscription plus **2 FREE CDs**

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

**9-Pack**  
Pick 9 of our magazines and save up to **\$400<sup>00</sup>**  
Pay only \$495 for a 1 year subscription plus **3 FREE CDs**

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

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

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

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

**Linux World Magazine**  
U.S. - Two Years (24) Cover: \$143 You Pay: \$79.99 / Save: \$63 + FREE \$198 CD  
U.S. - One Year (12) Cover: \$72 You Pay: \$39.99 / Save: \$32  
Can/Mex - Two Years (24) \$168 You Pay: \$119.99 / Save: \$48 + FREE \$198 CD  
Can/Mex - One Year (12) \$84 You Pay: \$79.99 / Save: \$4  
Intl - Two Years (24) \$216 You Pay: \$176 / Save: \$40 + FREE \$198 CD  
Intl - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

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

**Java Developer's Journal**  
U.S. - Two Years (24) Cover: \$144 You Pay: \$89 / Save: \$55 + FREE \$198 CD  
U.S. - One Year (12) Cover: \$72 You Pay: \$49.99 / Save: \$22  
Can/Mex - Two Years (24) \$168 You Pay: \$119.99 / Save: \$48 + FREE \$198 CD  
Can/Mex - One Year (12) \$84 You Pay: \$79.99 / Save: \$4  
Intl - Two Years (24) \$216 You Pay: \$176 / Save: \$40 + FREE \$198 CD  
Intl - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

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

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

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

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

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

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

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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE



# Analyzing Java Application Problems

## USING JAVA THREAD DUMPS TO ASSESS THE PROBLEM



BY **SATHISH SANTHANAM**

### AUTHOR BIO...

Sathish Santhanam is a developer relations engineer in the WebLogic/JRockit backline support team. He has more than six years of experience in software development, testing, and support; is a Sun-certified Java programmer and BEA certified developer; and is an expert in debugging server hang, performance, and JVM crash problems.

### CONTACT...

sathish.santhanam@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

This article discusses troubleshooting techniques for Java applications by analyzing Java thread dumps. We can use thread dumps to analyze situations like application hang, poor application response times, and application crash. Before getting into the details of analyzing the thread dumps, let's look briefly at the thread dump itself.

A Java thread dump is a snapshot of all the Java threads in a running Java application. It shows information like the current stack trace, state, and name of the threads. The list of threads includes the threads created by the JVM itself (to do housekeeping work like garbage collection, signal handling, etc.) and the threads created by the application.

You can get a thread dump by sending a SIGQUIT signal to the JVM. In case of Unix operating systems (Solaris/Linux/HP-Unix etc), this can be done by the “kill -3 <pid>” command. In the case of Windows, you can do a “ctrl-break” in the command window to get the thread dump. The thread dump is printed to the stdout or stderr of the JVM. The application will continue running normally after printing the dump. When you send the SIGQUIT signal to the JVM, the signal handler of the JVM responds to this signal by printing the thread dump. You can take thread dumps at any point while the application is running.

### A Sample Thread Dump

Listing 1 is a sample thread dump from a single-threaded application that is using the Sun JVM

1.4.1. The thread named “main” is the main application thread. All other threads are created by the JVM to do housekeeping work. While analyzing application-level problems, we generally focus only on the application threads. Let's analyze the stack trace of the “main” thread from Listing 1

```
"main" prio=5 tid=0x002358B8 nid=0x7f8 runnable
[6f000..6fc40]
  at test.method1(test.java:10)
  at test.main(test.java:5)
```

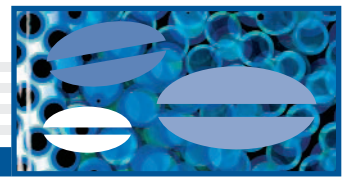
From this code snippet you can see that a thread stack trace has a name, thread priority (prio=5), state (runnable), source code line numbers, and the method calls. You can conclude from this stack trace that the thread “main” is executing some code in the method “method1” of class “test”. The call to this method came from the method “main” of the same class. You can also see the exact source code line numbers in those methods.

Before moving on to analyzing thread dumps from more complex scenarios, let's discuss the different state of threads that we normally see in the thread dumps and what they mean.

- **Runnable:** Either running or ready to run when it gets its CPU turn. JRockit thread dumps refer to this state as ACTIVE.
- **Waiting on monitor:** Either sleeping or waiting on an object for said period of time or waiting to be notified by another thread. This will happen when any of the sleep() methods on Thread object or wait() method on an Object is called.

For example, in BEA WebLogic Server the idle execute threads are in this condition and





they wait until a socket reader thread notifies them of some new work to be done. The stack trace will look like this:

```
"ExecuteThread: '2' for queue: 'weblogic.admin.RMI'" daemon prio=5
tid=0x1752f040 nid=0x180c in Object.wait() [1887f000..1887fd8c]
at java.lang.Object.wait(Native Method)
waiting on <04134D98> (a weblogic.kernel.ExecuteThread)
at java.lang.Object.wait(Object.java:426)
```

Some other versions of the JVM call this state CW. JRockit and refers to this state as WAITING.

- **Waiting for monitor entry:** Waiting to lock an object (some other thread may be holding the lock). This happens if two or more threads try to execute some synchronized code blocks/methods of an object at the same time. Please note that the lock is always for an object and not for individual methods. That is, if a thread has to execute a synchronized method of an object, it has to first lock that object.

The following is a sample stack trace of a thread in this condition:

```
"ExecuteThread: '24' for queue: 'DisplayExecuteQueue'" daemon prio=5
tid=0x5541b0 nid=0x3b waiting for monitor entry [49b7f000..49b7fc24]
at
weblogic.cluster.replication.ReplicationManager.createSecondary(Replication
Manager.java:908)
- waiting to lock <6c4b9130> (a java.lang.Object)
at
weblogic.cluster.replication.ReplicationManager.updateSecondary(Replication
Manager.java:715)
```

In this snippet you can see that this thread holds the lock on an object (6c408700) and is waiting to lock another object (6c4b9130).

Some other version of a JVM may not give the object IDs lock information in the stack trace. In those cases, we can guess that the thread is waiting to lock the object from the state of the thread. The same state may also be called as “MW”. JRockit refers to this state as LOCKED.

Now, let's see some thread dumps from different scenarios and analyze them to find the problem in the application.

## Thread Dump from a Deadlocked Application

In Listing 2 (only part of the thread dump is shown), you can see that the JVM has found the deadlock and it has given that information along with the thread dump. It clearly says that the “ExecuteThread: ‘47’ for queue: ‘default’” is waiting to lock an object that has been locked by “ExecuteThread: ‘57’ for queue: ‘default’”. (This thread dump is from a WebLogic Server. These threads are WebLogic Server’s worker threads, which process the client requests.) At the same time, ExecuteThread: ‘47’ has locked an object for which ExecuteThread: ‘57’ is waiting. This is a deadlock. Both the threads will never move from this state, waiting for the other thread to release the lock.

In Listing 2, Execute thread: 1 is waiting to lock the “ConnectionScavenger” object, but this has been locked by thread 47, which is in a deadlock with thread 57 as mentioned before. Therefore, Execute thread: 1 also won’t move further. This will eventually result in a hang because the other execute threads in the server will also try to get the lock for the same objects locked by Execute thread: 47 and 57 at some later time.

In some JVMs, the JVM will not give the deadlock information. It will just give the thread dump. In that case, we can arrive at this same conclusion by looking at the state and stack trace of the threads.

In this case the problem was due to a bug in WebLogic that was later fixed.

## Other Hang Scenarios

In some scenarios, you may see that most of the threads are “runnable” but the server may appear hung and won’t respond to client requests. If the threads are doing IO, they may be blocked in read() – the state will be “runnable” (database or some other network response may be poor). If this is the case, you need to check the health of the database and network.

Other reasons for a “runnable” thread to be stuck at the same place could be:

- An infinite looping in the code. This may also lead to high CPU usage.
- JVM garbage collection may be running for a long time, taking most of the CPU.
- The JVM process might have run out of file descriptors.

If the application code calls wait() or sleep(), the threads won’t move for the specified sleep time – the state of the threads in this case will be “Waiting on Monitor”. In this scenario, check the application code.

A lot of contention for a single object may also slow down the performance of the system, which may eventually lead to a hang-like situation. In this scenario, the state of most of the threads is “Waiting for monitor entry”. The application code needs to be investigated to reduce the contention on that particular object. The threads may also be waiting to get a response from another server and the other server may be hung for a different reason and it may not be able to send the response to this server.

In all of the above scenarios, it is also required that you take multiple thread dumps with a few seconds interval so that you can compare the state of a particular thread in all the thread dumps and you can decide whether the thread is moving or stuck.

## JVM Crash Scenarios

A JVM crash can happen due to:

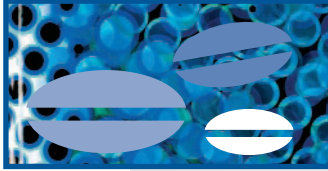
1. Bugs in the JVM library
2. Incorrect use of JNI APIs by the application
3. Bugs in the native modules used by the application – native database drivers, etc.

Normally, when the JVM crashes it gives the stack trace of the Java thread that caused the crash. Some JVMs give a complete thread dump (stack trace of all the threads) before exiting. Our interest in those thread dumps is to find which thread was running when the crash happened. This thread is called as a “current thread”. If the JVM gives a complete thread dump, it marks the “current thread”. Therefore, generally it’s easy to find the thread that caused the crash.

Note that the crash dump usually goes to stdout/stderr. The JVM may also generate a core file, which will be in binary format.

## Sample Crash Dump

If you look at the crash dump in Listing 3 you can see that the thread that caused the crash has the following stack trace:



Current Java thread:

```
at com.aaa.bbb.qqq.Direct.setString(Native Method)
at com.aaa.bbb.qqq.PreparedStatement.setString(PreparedStatement.java:51)
- locked <0x45029c60> (a com.aaa.bbb.qqq.PreparedStatement)
```

The last method executed (setString) was a “native method” in the class “com.aaa.bbb.qqq.Direct”. Therefore, we can guess that this crash is due to the native code in the implementation of the setString method. It could also be due to the JVM library code that has the implementation for the JNI API.

You can also see that the JVM gave the following in the dump:

```
Unexpected Signal : 11 occurred at PC=0x403AC9D1
Function=(null)+0x403AC9D1
Library=/opt/sunjdk/1.4.1_01-b01/jre/lib/i386/client/libjvm.so
```

```
# HotSpot Virtual Machine Error : 11
# Error ID : 4F530E43505002E6
# Please report this error at
# http://java.sun.com/cgi-bin/bugreport.cgi
```

This means that the code that caused the crash (PC=0x403AC9D1) is part of the JVM library: “/opt/sun-jdk/1.4.1\_01-b01/jre/lib/i386/client/libjvm.so”. This doesn’t mean that this is necessarily a JVM bug. Even though the instruction that caused the crash is in the JVM code, the crash could be due to a JNI call made by the application code (in this case, the implementation of the native method “setString”) that could have passed some incorrect arguments, which caused the crash in the JVM code.

If you don’t have the “current thread” info in the dump, then we need to analyze core dumps. This is out of the scope of this document. 🍷

### Listing 1: Sample thread dump

Full thread dump Java HotSpot(TM) Client VM (1.4.1\_02-ea-b01 mixed mode):

```
"Signal Dispatcher" daemon prio=10 tid=0x008F91D8 nid=0xb20 waiting on condition
"Finalizer" daemon prio=9 tid=0x008F77E8 nid=0x1800 in Object.wait()
"Reference Handler" daemon prio=10 tid=0x008F6360 nid=0xecc in Object.wait()

"main" prio=5 tid=0x002358B8 nid=0x7f8 runnable [6f000..6fc40]
  at test.method1(test.java:10)
  at test.main(test.java:5)

"VM Thread" prio=5 tid=0x008F5128 nid=0xa60 runnable
"VM Periodic Task Thread" prio=10 tid=0x0023FD68 nid=0xb58 waiting on condition
"Suspend Checker Thread" prio=10 tid=0x008F8808 nid=0xb94 runnable
```

### Listing 2: Java-level deadlock

```
"ExecuteThread: '1' for queue: 'default'" daemon prio=1 tid=0x6da4b380
nid=0x1667 waiting for monitor entry
  at
  weblogic.iiop.ConnectionManager$ConnectionScavenger.watch(ConnectionManager.java:390)
```

FOUND A JAVA LEVEL DEADLOCK:

```
-----
"ExecuteThread: '47' for queue: 'default'":
  waiting to lock monitor 0x808645c (object 0x43fa1fd0, a weblogic.socket.PosixSocketMuxer$FDRecord),
  which is locked by "ExecuteThread: '57' for queue: 'default'"
"ExecuteThread: '57' for queue: 'default'":
  waiting to lock monitor 0x80863b4 (object 0x6c84ff40, a java.lang.Class),
  which is locked by "ExecuteThread: '47' for queue: 'default'"
```

JAVA STACK INFORMATION FOR THREADS LISTED ABOVE:

```
-----
Java Stack for "ExecuteThread: '47' for queue: 'default'":
=====
  at
  weblogic.socket.PosixSocketMuxer.cleanup(PosixSocketMuxer.java:438)
  at
  weblogic.iiop.MuxableSocketIIOP.close(MuxableSocketIIOP.java:474)
  at
  weblogic.iiop.ConnectionManager.closeConnection(ConnectionManager.java:318)
  )
  at
```

```
weblogic.iiop.ConnectionManager.access$1(ConnectionManager.java:301)
  at weblogic.iiop.ConnectionManager$ConnectionScavenger.trigger(ConnectionManager.java:380)
  .....

Java Stack for "ExecuteThread: '57' for queue: 'default'":
=====
  at
  weblogic.iiop.EndPointManager.findOrCreateEndPoint(EndPointManager.java:42)
  )
  at
  weblogic.iiop.ConnectionManager.dispatch(ConnectionManager.java:246)
  at
  weblogic.iiop.MuxableSocketIIOP.dispatch(MuxableSocketIIOP.java:507)
  at
  weblogic.socket.PosixSocketMuxer.deliverGoodNews(PosixSocketMuxer.java:738)
  )

Found 1 deadlock.
```

### Listing 3: Sample crash dump

```
Unexpected Signal : 11 occurred at PC=0x403AC9D1
Function=(null)+0x403AC9D1
Library=/opt/sunjdk/1.4.1_01-b01/jre/lib/i386/client/libjvm.so
```

```
Current Java thread:
at com.aaa.bbb.qqq.Direct.setString(Native Method)
at com.aaa.bbb.qqq.PreparedStatement.setString(PreparedStatement.java:51)
- locked <0x45029c60> (a com.aaa.bbb.qqq.PreparedStatement)
at com.mmm.bbb.dao.jdbc.Request.executeUpdate(RequestDAO.java:11)
```

Dynamic libraries:

```
-----
40186000-4018b000 rw-p 00132000 08:01 1235494 /lib/i686/libc-2.2.4.so
4018f000-404a7000 r-xp 00000000 08:01 1544743 /opt/sunjdk/1.4.1_01-b01/jre/lib/i386/client/libjvm.so
404a7000-4065b000 rw-p 00317000 08:01 1544743 /opt/sunjdk/1.4.1_01-b01/jre/lib/i386/client/libjvm.so
4066b000-4067e000 r-xp 00000000 08:01 65081 /lib/libnsl-2.2.4.so
4067e000-4067f000 rw-p 00012000 08:01 65081 /lib/libnsl-2.2.4.so
-----

# HotSpot Virtual Machine Error : 11
# Error ID : 4F530E43505002E6
# Please report this error at
# http://java.sun.com/cgi-bin/bugreport.cgi
#
# Java VM: Java HotSpot(TM) Client VM (1.4.1_01-b01 mixed mode)
#
```

# SYS-CON MEDIA

304,187 of the World's Foremost IT Professionals

DIRECT MAIL, EMAIL OR MULTI-CHANNEL

Target CTOs, CIOs and CXO-level IT professionals and developers who subscribe to SYS-CON Media's industry leading publications

**Java Developer's Journal...**  
The leading publication aimed specifically at corporate and independent java development professionals



**PowerBuilder Developer's Journal...**  
The only PowerBuilder resource for corporate and independent enterprise client/server and web developers



**ColdFusion Developer's Journal...**  
The only publication dedicated to ColdFusion web development

**LinuxWorld Magazine...**  
The premier monthly resource of Linux news for executives with key buying influences



**WebLogic Developer's Journal...**  
The official magazine for BEA WebLogic application server software developers, IT management & users

**Web Services Journal...**  
The only Web Services magazine for CIOs, tech, marketing & product managers, VARs/ISVs, enterprise/app architects & developers



**.NET Developer's Journal...**  
The must read iTechnology publication for Windows developers & CXO management professionals

**XML-Journal...**  
The world's #1 leading XML resource for CEOs, CTOs, technology solution architects, product managers, programmers and developers



**WebSphere Developer's Journal...**  
The premier publication for those who design, build, customize, deploy, or administer IBM's WebSphere suite of Web Services software



**Wireless Business & Technology...**  
The wireless magazine for key corporate & engineering managers, and other executives who purchase communications products/services

Recommended for a variety of offers including Java, Internet, enterprise computing, e-business applications, training, software, hardware, data back up and storage, business applications, subscriptions, financial services, high ticket gifts and much more.

**NOW AVAILABLE!**  
The SYS-CON  
Media Database  
304,187 postal  
addresses



For email information:  
contact Frank at 845-731-3832  
frank.cipolla@epostdirect.com  
epostdirect.com 800-409-4443 fax 845-620-9035

For postal information:  
contact Kevin at 845-731-2684  
kevin.collopy@edithroman.com  
edithroman.com 800-223-2194 fax 845-620-9035





# News & Developments

## Norvergence Standardizes on BEA WebLogic Platform 8.1

(San Jose, CA) – BEA Systems, Inc., the world's leading application infrastructure software company, has announced that Norvergence, a provider of telecommunications services for growing companies, has adopted BEA WebLogic Platform 8.1 as its standard for enterprise application development, deployment, and integration.

Norvergence is a \$200 million business equipment and systems engineering company that provides "next-generation" communications equipment designed to help businesses reduce the cost of voice and data services by as much as 60%. The company is using the BEA WebLogic Platform to sim-



ply and accelerate the rollout of essential applications. Norvergence also uses the business process management capabilities in WebLogic Integration that are designed to automate many workflows and increase organizational productivity. [www.bea.com](http://www.bea.com), [www.norvergence.com](http://www.norvergence.com)

## BEA Helps NASA Seek Water, Perhaps Life, on Mars

(San Jose, CA) – The BEA WebLogic Enterprise Platform is delivering information to NASA scientists around the world about whether water, and perhaps life, once existed on Mars. Scientists at the Jet Propulsion Laboratory, and other NASA facilities and universities, are actively studying data beamed by the two Mars Exploration rovers from the

planet's surface. This data travels over NASA's Deep Space Network, an international network of antennae located in Australia, Spain, and the U.S., is processed by ground-based software and then stored on file servers at JPL. The processed data is available to scientists and researchers over the Internet through a middle-ware layer powered by BEA WebLogic Server.

It also provides event and personnel schedules and the current time in various Earth and Mars time zones, while the message service enables mission managers to broadcast messages. [www.bea.com](http://www.bea.com), [www.nasa.gov](http://www.nasa.gov)



## BEA Announces Expanded Offerings to Support Growing Linux Demand

(New York City) – BEA Systems has announced a new partnership and expanded products, services, and programs for Linux. Technologies like BEA WebLogic JRockit 1.4.2 and partnerships with SuSE LINUX can help BEA support customers building infrastructures based on the Linux platform.

Later this year, BEA WebLogic Enterprise Security is scheduled to support Red Hat Linux Advanced Server 2.1.

BEA has also strengthened its ties with SuSE LINUX, a product unit of Novell and HP. SuSE LINUX has agreed to codevelop Linux solutions and bundle BEA WebLogic JRockit as part of its product offering. <http://dev2dev.bea.com/subscriptions/index.jsp>

## Quest Software Releases Commercial Java Performance Tool

(Irvine, CA) – Quest Software,

Inc., a provider of application management solutions, has announced that its Quest JProbe performance tuning suite now supports 64-bit Windows and Linux Java environments. The first and only commercial toolset for 64-bit Java, Quest JProbe enables developers to analyze Java code running on 64-bit Java Virtual Machines (JVMs) on Intel Itanium 2-based operating systems including Microsoft Windows and Red Hat Linux.

New 64-bit architectures enable Java applications to access larger data sets and improve scalability. Quest JProbe is a complete performance tuning toolkit for Java code. Graphically depicting everything from memory usage to calling relationships, Quest JProbe helps developers understand precisely



what is causing problems in Java applications – right down to the offending line of source code.

Quest JProbe for 64-bit Java platforms is available now with pricing starting at \$2,000.

[www.quest.com/solutions/java\\_j2ee.asp](http://www.quest.com/solutions/java_j2ee.asp)

## New Study Finds Advantages for Integrated Platform Suite

(San Jose, CA) – BEA Systems, has announced the findings of a study that demonstrates potential for enterprise-wide cost savings by leveraging an integrated application platform suite (APS). The study, based on a detailed analysis of projects and data from systems integrators and Fortune 500 enterprises, highlights significant time-to-market and cost advantages – 22% faster, 25% less cost and assets – for IT initiatives implemented on an integrated platform when compared to projects implemented on a non-integrated platform.

This study compares the time-to-market and skills required for applications development, integration, deployment, and maintenance on an integrated versus non-integrated platform. It found that an integrated platform leads to savings during every phase of the application life cycle, from design to post-deployment operations. [www.bea.com](http://www.bea.com)

## E\*TRADE Japan Builds Linux-Based Trading System

(San Jose, CA) – E\*TRADE Japan, a wholly owned investment of Softbank Investment Corp., has selected BEA WebLogic Server and BEA WebLogic JRockit for its Linux-based online trading system. The new online securities trading system is one of the first to adopt a Linux and Intel-based architecture.

E\*TRADE Japan deployed an IA server, Linux OS, and Red Hat Advanced Server to its online trading system in order to improve site performance and online response times, enhance customer satisfaction,



and lower the total cost of ownership of the system. To maximize application performance and transaction processing of the new Linux-based online trading system, E\*TRADE Japan deployed BEA WebLogic Server and BEA WebLogic JRockit. The system, developed by BEA partners Nomura Research Institute, Ltd., Hitachi Ltd., and Hitachi Software Engineering Co., Ltd., is nearly 10 times more cost-effective. In addition, performance and transaction processing has nearly doubled, and hardware and maintenance costs were reduced by 20%.

[www.bea.com](http://www.bea.com),

[www.etrade.ne.jp](http://www.etrade.ne.jp)



# Forget something?

**Post-launch is NOT the time to be verifying web applications.**

**The wild blue yonder of operational monitoring and management is extremely unforgiving.** Which means that going live with the monitoring software you used in development is a great way to go dead—quickly! You simply can't support operations if your staff is drowning in details provided by development profiling tools and debuggers. **Let NetIQ cover your apps...with AppManager.**

AppManager—the industry's easiest-to-use Systems Management suite—is a proven management system for monitoring J2EE application servers, databases, operating systems and even end-user response time. NetIQ's AppManager monitors ALL application components—not just your server. **NetIQ. Nobody does UNIX better. Nobody.**

Visit us at [www.netiq.com/solutions/web](http://www.netiq.com/solutions/web) to learn how we can help you address the challenges of your operational monitoring and management.







## Keep the little problems from turning into big ones

diagnoSys

Learn how to improve your  
Web applications at H&W's  
free Web seminar.  
Visit [www.hwcs.com/wldj1.asp](http://www.hwcs.com/wldj1.asp)

You might not even know you have a leak, but that little drip can turn into a big deal if it's left unattended. Performance problems with your Web applications work the same way.

Hung threads, memory leaks, and faulty network connections can cause your application to perform poorly or even fail when it matters most – in production. Customers click away and profits plummet. But how do you find the problems if you don't even know they are there?

DiagnoSys is a complete software solution that allows you to analyze the performance of your Web applications through all phases of the life cycle. With true statistical correlation, in-depth analysis, and root cause identification, it helps your teams work together to find problems while they are still little.

The end result? With DiagnoSys, you build vital applications that optimize uptime and keep the bottom line healthy. No problem.

© 2003-2004 H&W Computer Systems, Inc.  
DiagnoSys is a trademark of H&W Computer Systems, Inc.

  
[www.hwcs.com](http://www.hwcs.com) | 1.800.338.6692

Bridging the Enterprise Computing Gap Since 1979