SYS-CON

THE SOA DIET: BUILDING ON THE NEWEST RAGE PAGE 4

SEPT/OCT 2004

wldj™

THE LEADING INDEPENDENT MAGAZINE FOR WEBLOGIC™ PROFESSIONALS

WWW.WLDJ.COM

SOA-Based Applications

TIME TO GET ON THE BUS ...6

PLUS...

$9.99US $9.99CAN

09

0 74470 03424 7

TANGOSOL™

Ready for a stampede?

TANGOSOL COHERENCE™ provides even the most heavily burdened applications with predictable and linear scalability while shielding database systems from unnecessary load. Coherence powers some of the largest application server clusters, providing coherent data caching, stateful clustering and session management for clusters of up to hundreds of servers.

Try before you buy. Download free evaluation software at

www.tangosol.com

Coherence™ is pure Java software that reliably and cost-effectively scales J2EE applications across any number of servers in a cluster. Coherence provides reliable coordinated access to in-memory data that enables safe data caching and user session management for applications in a clustered environment.

Sales	+1.617.623.5782 or sales@tangosol.com
Support	support@tangosol.com or http://www.tangosol.net
Pricing	US$1,995 to US$4,995 per Coherence production CPU Development licenses are offered free of charge.

Managing your SOA apps can be slippery.

Remove the risks with Intersperse Manager.

Intersperse Manager™ is designed specifically to provide production management of applications deployed on BEA.

Leveraging the JMX standard, Intersperse Manager simultaneously manages both vertical apps and horizontal integrations—and market-moving capabilities enable applications to self-manage, self-heal, and self-protect.

Intersperse Manager gives you the tools to reap the rewards and reduce the risks of J2EE-based SOA application deployments, and that puts you in control.

For a free demonstration,
call Intersperse today at
866.499.2202
or visit www.intersperse.com.

intersperse™

Management solutions
for service-oriented
enterprise applications

626.535.2000
www.intersperse.com

The SOA Diet

BY JOE MITCHKO

It never ceases to amaze me how something can move from essential obscurity to mainstream hype in what seems to be only overnight. Take the low carbohydrate diets, which are all the rage now. For years, the Atkins diet was considered by most diet professionals to be pure nonsense – how can you lose weight with a diet rich in fat, with bacon and eggs in the morning and a pound of steak or two for dinner? Yet all it took was one investigative report from a prominent newspaper to give the diet credibility. Now, you can't seem to get away from low-carb food products – they're everywhere. Yet, for the most part, anyone who eats in moderation and has kept with the traditionally balanced diet stands a good chance of keeping their weight down without any fancy diet or trend. A lot of it is pure common sense and good old-fashioned self control.

By now you may be asking yourself whether the publisher has made a mistake and somehow mixed up editorials with some nutrition magazine. But wait; stay with me for a minute and you will see there is some relevance here to the WebLogic world.

Today, SOA (service-oriented architecture) is all the rage in the computing world. SOA promises to revolutionize the business of application integration so as to make enterprise applications, legacy and new, work together as one. Now, everyone seems to want it, and by golly, an increasing number of vendors out there have products that support various aspects of SOA development and deployment.

I do believe SOA holds a lot of promise for the future, especially when you begin to coordinate and orchestrate business workflow and processes using products such as WebLogic Integrator. But SOA, in and of itself, is not an instant "cure-all" for today's IT computing ills. The problems facing most enterprise systems are not going to go away overnight by moving towards an SOA-based architecture. As a matter of fact, the efforts I have seen so far moving towards SOA have only magnified the years of neglect and bad design decisions made by IT . And that is not to mention the host of hidden business rules, bad design, garbage data, and other gremlins typically found in most enterprise database systems.

SOA-based architecture does not replace enterprise applications, but instead builds upon them. If the applications lying beneath the shallow SOAP interface are shaky and are prone to constant problems, the Web services will reflect that. No matter how well you design your SOA-enabled business processes, there is no way it is all going to work correctly or reliably.

Now, for all of you WebLogic developers out there, it means this: more than ever, you need to be able to stay proficient in the underlying J2EE and WebLogic technologies and continue to design and develop high-quality applications using all of the "best practices" you can get your hands on. A well-designed application, along with a solid database design and proper configuration management practices, will make implementing SOA-based applications so much easier. You cannot ignore the basics.

And, this is where we come in here at ***WLDJ***. We will continue to provide you with the high-quality articles you have come to expect, as well as get you acquainted with the latest trends affecting the BEA platform, as you will see in this special SOA issue.

Some say SOA is really nothing more than rehashed ideas from the past – just CORBA revisited. Others say that it is revolutionary and bound to change everything. Whatever happens, we will continue to support you as a WebLogic developer.

Now, let me get back to that steak I was eating.

AUTHOR BIO...

Joe Mitchko is the editor-in-chief of *WLDJ* and a senior technical specialist for a leading consulting services company.

CONTACT: joe@sys-con.com

wldj THE LEADING INDEPENDENT MAGAZINE FOR WEBLOGIC PROFESSIONALS

EDITORIAL ADVISORY BOARD
LEWIS CIRNE, TYLER JEWELL, WAYNE LESLEY LUND, CHRIS PELTZ, SEAN RHODY

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,

ASSISTANT EDITOR
TORREY GAVER

RESEARCH EDITOR
BAHADIR KARUV, PhD

WRITERS IN THIS ISSUE
ANJALI ANAGOL SUBBARAO, ARUN CANDADAI, SATYA GHATTU, PETER HOLDITCH, COCO JAENICKE, ANBARASU KRISHNASWAMY, SAI KUMAR, VIJAY MANDAVA, JOE MITCHKO, FRANCO NEGRI, ERIC NEWCOMER, RAJUL RANA

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

PRODUCTION CONSULTANT
JIM MORGAN

ART DIRECTOR
ALEX BOTERO

LEAD DESIGNER
ABRAHAM ADDO

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

ADVERTISING SALES MANAGER
MEGAN MUSSA

ASSOCIATE SALES MANAGERS
KRISTIN KUHNLE • BETH JONES • DOROTHY GIL

PRESIDENT, SYS-CON EVENTS
GRISHA DAVIDA

NATIONAL SALES MANAGER
JIM HANCHROW

FINANCIAL ANALYST
JOAN LAROSE

ACCOUNTS RECEIVABLE
CHARLOTTE LOPEZ

ACCOUNTS PAYABLE
BETTY WHITE

VP, INFORMATION SYSTEMS
ROBERT DIAMOND

WEB DESIGNERS
STEPHEN KILMURRAY • MATTHEW POLLOTA

ONLINE EDITOR
MARTIN WEZDECKI

CIRCULATION SERVICE COORDINATORS
EDNA EARLE RUSSELL • LINDA LIPTON

SUBSCRIPTIONS:
888-123-4567
201-123-4566
subscribe@sys-con.com

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
WebLogic Journal (ISSN# 1535-9581)
is published monthly (12 times a year)
Postmaster: Send Address Changes to
WEBLOGIC 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@epostdirect.com

SYS-CON EVENTS

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
Application Management
1 888 GET WILY
www.wilytech.com

wily technology

A Dynamic Implementation Framework for SOA-Based Applications

THE KEY TO SOA SOFTWARE AND ENTERPRISE APPS OF THE FUTURE

BY **ARUN CANDADAI**

AUTHOR BIO...

Arun Candadai is the lead software architect at Soactive and drives the architecture and development of its SOA products. He has worked as a senior architect at BEA Systems, where he pioneered the use of Web services and SOA for its next-generation infrastructure. Prior to BEA, Arun held senior engineering positions for companies like Asera, Covad, and Lockheed Martin. He holds two patents in Internet software.

CONTACT...

arun.candadai@soactive.com

Today's IT environments are increasingly characterized by heterogeneous and complex applications, tight schedules, budgetary constraints, and an ever-changing landscape of business requirements.

Few businesses have been agile enough to enhance their existing infrastructures to meet and overcome these challenges in an effective manner. Even so, to deal with a continuous flow of highly complex and dynamic set of business requirements rapidly and cost effectively, businesses need a flexible and dynamic approach to automate, build, and manage critical business processes.

Solution Approach

Service-oriented architecture (SOA) is often touted as a feasible solution approach to solving the business challenges mentioned above. SOA is a method of conceptualizing, designing, and building applications by using and assembling building blocks, each of which is usually represented as a reusable service. Many current approaches to SOA simply involve wrapping pieces of business functionality and using them within applications, often in ad hoc, static, and inflexible ways. The proposed approach to develop applications and business processes of the future is to employ a formal SOA implementation framework that is dynamic, flexible, and scalable enough to meet changing and complex business requirements. Regardless of whether you buy or build a framework for your SOA implementation, the functionality of this framework will hold good for your solution.

SOA Implementation Framework Overview

The SOA implementation framework is the enabling technology for efficiently building applications and business processes using SOA principles. It provides architects, developers, and administrators with an operational framework and tools to configure, use, and manage enterprise services that form the building blocks for applications and business processes. This framework uses a service-centric approach at all levels and stages of the implementation process and has the following generic characteristics:

- Ability to dynamically connect clients with services in a highly secure, protocol-independent manner
- Ability to reliably handle synchronous and asynchronous modes of service execution
- Ability to define and handle events in a declarative manner
- Ability to dynamically convert between data formats of clients and services
- Ability to manage distributed SOA resources (services, configurations, policies, etc.) in a centralized manner
- Ability to capture and handle exceptions in the service execution process
- Ability to log and monitor various events and metrics as they arise during the client-service transaction
- Provide a unified and reusable service invocation code library for use across client applications in the enterprise
- Support for Web services standards stack to promote wide-scale adoption and interoperability

Framework Components

Service Registry

The service registry is a foundational piece of the SOA-enabled solution for the enterprise. It is used to define, configure, and virtualize business services used by applications in a centralized manner. Most companies fail to realize the full potential of SOA due to the lack of a well-designed and managed service registry. It includes crucial information needed for defining and provisioning the services – services, providers, consumers, service interactions, policies, and all associated configurations. The service registry resides in a high performance data store and can be viewed and managed using the service manager, which is described later.

Service Bus

The service bus is a high-performance component that mediates between client applications and services and adds value by providing generic pieces of functionality on behalf of the clients, as well as the services aimed at promoting unified and reusable technical functionality – a key to real SOA implementations. It enables clients to focus mainly on business logic by providing a standard mechanism to connect to services and encapsulating service implementation details.

The service bus uses a pipeline approach, where the bus can be visualized as a sequence of components – a pipeline. Each stage in the pipeline is essentially a value-add component that takes in a set of inputs, where applicable, from the previous stage, processes the data, and passes on the outputs to the next stage.

The following are the features of the service bus:

- ***Dynamic connectivity and routing:*** Dynamic connectivity is the ability to connect to Web services dynamically without using a separate static API or proxy for each service. Most enterprise applications today operate on a static connectivity mode, requiring some static piece of code for each service. Dynamic service connectivity is key to enterprise agility. The dynamic connectivity API is the same regardless of the service implementation protocol (Web services, JMS, EJB/RMI, POJO, etc.). Client applications access services via URI interfaces that may either directly map to services or be routed based on the context or content of the service request.

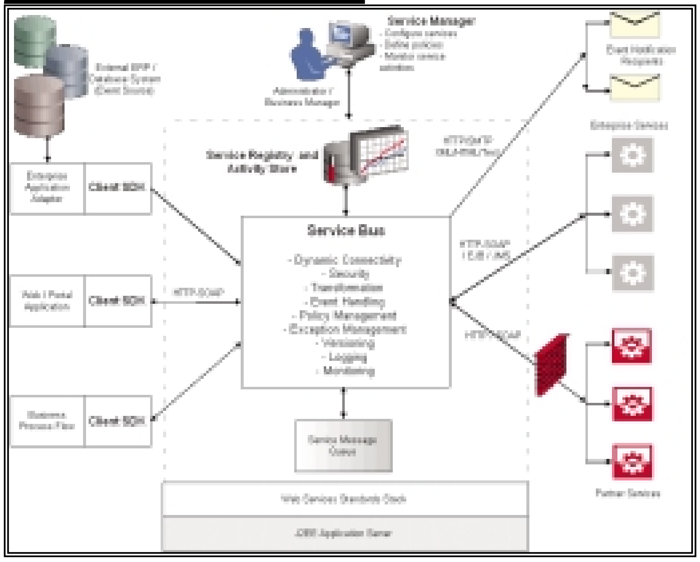
- ***Reliable messaging:*** Reliable messaging is the ability to queue service request messages and ensure guaranteed delivery of these messages to the destination. It also includes the ability to respond, if necessary, back to the requestor with response messages. Primarily used for handling events, this capability is crucial for responding to clients in an asynchronous manner, and for a successful SOA implementation. This is typically implemented using reliable JMS queues with store and forward as well as guaranteed delivery capabilities.

- ***Security:*** Generically handling and enforcing security is a key success factor for SOA implementations. The following are the main considerations:
  - *Federated authentication:* This feature intercepts service requests and adds the appropriate username and credentials. It is also capable of authenticating service requests before dispatching the request for service execution.
  - *Authorization:* Validates each service request and authorizes it to make sure that the sender has the appropriate privilege to access the service.
  - *Encryption/decryption:* Encrypts XML content at the element level for both request and response messages and performs decryption for the reverse scenario.

- ***Transformation:*** Needed when a client and a service use different data formats, it is the ability to transform a given source of data into a target format based on specified transformation rules.

- ***Caching and performance strategies:*** Important to improve the performance and quality of services and eventually to enhance overall customer service. It is performed at various levels, including service configurations, service response data services, and other SOA resources, depending on where optimization is needed to achieve high performance. Overall performance can also be improved by using compressed XML-based messages when transporting them across the nodes; the result is reduced

FIGURE 1



SOA implementation framework architecture bandwidth utilization. To accomplish this, compression and decompression agents are installed at the edges of the SOA grid.

- ***Logging:*** Needed to trace system execution and performance for auditing, troubleshooting, and monitoring purposes. Logging can be done for any of the services and can be done at various severity levels.

- ***Monitoring:*** The ability to track service activities that take place via the bus and provide visibility into various metrics and statistics. Of particular significance is the ability to be able to spot problems and exceptions in the business processes and move toward resolving them as soon as they occur.

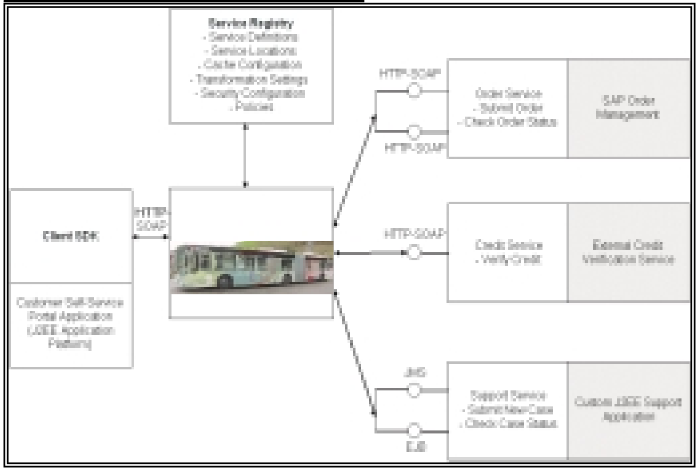
- ***Service-level agreement (SLA):*** SLAs specify performance guarantees associated with Web services operations and business processes –such as response times and service availability – that are essential for business-critical operations. Service providers can use SLAs to advertise and create service-level agreements with consumers. This helps them to provide the appropriate amount of resources and prioritize the servicing of requests.

- ***Versioning:*** The ability to version Web services and to proactively assist in the migration of the client applications to use new versions of the Web service as soon as they become available. Versioning support includes being able to register different versions of the same service, to provide the client with the necessary APIs and libraries for all versions, and to help in migrating clients to the new versions. The versioning component has the ability to transform requests to older versions of the service to the newer version using XLST-based transformation.

- ***Exception management:*** Ability to track and handle exceptions as they occur when clients access business services. Exception details such as the exception code, name, reason, and description are captured and logged. Exceptions are also handled appropriately by retrying the connection a predefined number of times, routing to an alternate service, or by simply returning a reason for execution failure.

- ***Custom business logic processing:*** Provides the flexibility to insert custom business logic during service processing. For instance, a specific business logic can be performed either on the service request, prior to submitting the service request for execution, or on the service response, prior to sending the service response back to the client.

FIGURE 2



Customer self-service portal application usage scenario

The service bus is itself implemented as a service, using Web service or EJB technologies. This service is deployed to a server pool or cluster and is designed to scale horizontally. Each of the components mentioned above must conform to the corresponding Web services standards.

“Businesses need a flexible and dynamic approach to the automation, building, and management of critical business processes”

Service Manager

The service manager is a UI-based administration tool that empowers administrators and business managers to define, configure, manage, and monitor business services and related SOA resources used within the applications. It can be thought of as the control center for the SOA implementation, used mainly to provision and monitor services. The following are the two key functions of the service manager:

- ***Service configuration management:*** Using this feature, administrators can onboard and configure services housed in the service registry. Users can define services, locations, binding information, and service configuration settings such as security, cache, transformation, logging, and monitoring. They can also define service providers and consumers and associated service contracts.
- ***Service activity monitoring:*** Provides a dashboard for business managers and administrators to view key metrics and statistics in relation to service activities that take place via the Bus. Users can view and monitor service usage, response times, service exceptions, activity logs, messages, etc.

The service manager is delivered as a Web-based application and can be accessed without any client-side installation requirements.

Client SDK

The Client SDK is software that is needed to connect with the service bus. The SDK – downloadable from the service manager – is packaged and provided in the form of an API. It is an easy-to-use library that application developers can use to discover, access, and utilize business services from their applications and business processes. This significantly enhances developer productivity as it offloads developers from message plumbing tasks that are, in turn, performed by the service bus. The client API does not use service location–dependent, hardwired proxies and stubs to access services. Instead, it uses a dynamic mechanism to connect to the services via the bus.

–continued on page 14

Need to solve performance problems in BEA WebLogic™ 8.1 applications?

Manage Complex Enterprise Applications
Acsera has the advanced technology needed to manage complex BEA WebLogic™ 8.1 applications in production.

100% Automatic Discovery and Blueprinting
Acsera discovers your WebLogic clusters, nodes, servers and deployed applications automatically in minutes.

Performance Management and Monitoring
Acsera is Performance Centric – it helps you deliver the high throughput and reliability your enterprise needs from WLI.

Complete Visibility into Business Processes
No more 'black box' applications in production. Acsera shows business processes and their complete hierarchy.

Root Cause Analysis on Running Applications
No more 'mystery performance problems' on WLI. Acsera helps you diagnose and isolate problems while in production.

Manages Across Clusters
Your WLI applications deploy across nodes and clusters – don't be trapped into managing with server-by-server tools.

Results in Minutes
You can't afford to wait hours or days to find WLI problems and correct them – Acsera delivers results in minutes.

Attend a WebLogic Integration performance webinar: www.acsera.com/register

ACSERA

Application Monitoring and Performance Management for BEA WebLogic™ 8.1

BEA WebLogic is a trademark of BEA Systems, Inc.
Acsera is a trademark of Acsera Corporation.

Performance Management Console
Shows application performance and health.

Application Blueprint Console
Gives 100% visibility into application hierarchy.

Root Cause Analysis Console
Isolates performance problems in minutes.

Canonical Message Formats

AVOIDING THE PITFALLS

BY **COCO JAENICKE**

AUTHOR BIO...

Coco Jaenicke has been an active evangelist for XML and Web services for over six years. She played key roles at both eXcelon Corporation and SilverStream Software as the director of product marketing, bringing next-generation, XML-based products to market, and she now consults for early stage companies in the greater Boston area. (Photo by Alexa Jaenicke)

CONTACT...

coco.jaenicke@pantero.com

As the scope of enterprise integration grows, IT organizations are demanding greater efficiency and agility from their architectures and are moving away from point-to-point integration,which is proving to be increasingly cumbersome to build and maintain.

They are migrating towards adaptive platforms such as BEA's 8.1 Platform and many-to-many architectures that supports linear growth costs as well as simplified maintenance. To connect systems, WebLogic developers are shifting away from creating individual adapters between every pair of systems in favor of Web services. For data, they are shifting away from individual mappings between data sources and targets in favor of Liquid Data enhanced with canonical messages. But trying to implement canonical messages can be rife with problems such as causing infighting between departments and creating rigid models that become obstacles to progress. Fortunately, these pitfalls can be avoided through proper architecture.

Introduction

As IT professionals engineer greater efficiency and flexibility into their enterprises, the industry has seen a shift from point-to-point connectivity to platforms such as BEA WebLogic that support many-to-many integration. The heart of this shift is a service-oriented architecture (SOA) that brings the world one step closer to IT infrastructure that is truly plug-and-play. However, any CIO who has implemented an SOA can tell you that the SOA still has some hurdles to clear before it delivers on the plug-and-play future. The most imposing of these hurdles is data interoperability.

Data documents are exchanged blindly because the SOA offers no systematic way to standardize the way in which inbound data is interpreted and validated. Even if an organization standardizes on an industry document-type definition (DTD), there is still no automated way to reconcile, for example, the strings "two," "2," and "2.00."

XML Schema Definitions (XSDs) and LiquidData help with data typing and transformations, but that's just the tip of the iceberg. What if the value of 2 is derived from multiple data sources? What if another system reports that the value is really 3? When should the value 2.001 be rounded to 2? What if there is a business requirement that dictates that the value must be greater than 5?

By most accounts, over 40% of the cost of integrating applications is spent writing point-to-point adapters for the transformation, aggregation, validation, and business operations needed to effectively exchange data – and much of the code is redundant, inconsistent, and costly to maintain. This approach results in a brittle architecture that is expensive to build and difficult to change without more hand coding.

Canonical models are a step in the right direction, but exchange models – a superset of canonical models – solve the entire problem with a many-to-many solution for modeling, automating, and managing the exchange of data across an SOA. They capture data reconciliation and validation operations as metadata, defined once but used consistently across the SOA. In this way, rules are enforced, reused, and easily modified without incurring additional integration costs. Without such infrastructure in place, data interoperability problems threaten the flexibility and reuse of the SOA – the incentives for implementing an SOA in the first place.

SOAs Ignore Underlying Data

Web services provide a standards-based mechanism for applications to access business logic and ship documents, but they provide nothing for interpreting and validating the data in the documents within the context of the local application. In addition to the simple examples given above, IT departments are left with the sizable task of writing code to interpret and convert data every time a document is received. The operations performed include:

- ***Transformations:*** Schema mapping, such as converting credit bureau data into an application document
- ***Aggregation:*** Merging and reconciling disparate data, such as merging two customer profiles with different and potentially conflicting fields
- ***Validation:*** Ensuring data consistency, such as checking that the birth date precedes the death date
- ***Business operations:*** Enforcing business processes, such as ensuring that all orders fall within corporate risk guidelines

Consider the apparently simple concept of "order status". This concept is vital to the correct execution of many business processes within the enterprise, but what are the true semantics of the term? Is it the status of issuing a request for a quote, or placing a purchase order? What about the status of a credit check on the customer? What limitations should be placed on the order if the credit check reveals anomalies? Perhaps "order status" refers to the manufacturing, shipping, delivery, return, or payment status. In practice, it's probably all of these things as well as the interdependencies between them. Applications in different functional areas of the corporation will have different interpretations of "order status" and different databases will store different "order status" values.

Each database and application within a given area of responsibility, such as order management, manufacturing, or shipping, uses a locally consistent definition of "order status". However, when an enterprise begins using an SOA to integrate processes across these areas, source systems must convert their local definitions to the local definitions of the target systems. And the problem increases with the number of target systems – order management systems may send messages to both manufacturing systems and shipping systems, each with a different definition of "order status".

Converting these messages for the target system is labor intensive and goes beyond simple mappings. IT departments are often forced to alter applications and embed additional code in order to execute any of the operations shown in Table 1.

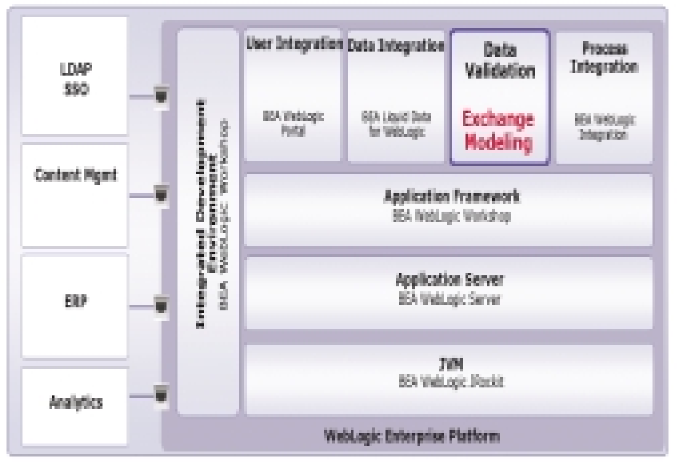
These tasks are often implemented with hand coding, which creates a brittle set of data services that scatter the semantics of reconciling data across the entire service network. What is worse is that this code is not reusable from project to project. Worse still is that it creates an architecture that is not amenable to change. Changing a single logical rule anywhere on the network sets off a domino effect in which every application service that has implemented this rule must also be changed – assuming each instance of the rule can even be found.

The interoperability of data plays just as big a role in SOAs as the interoperability of services, and if the mechanics of exchanging data are not addressed the two key reasons for implementing an SOA in the first place – flexibility and reuse – are severely threatened. Fortunately, this problem can be addressed through good architecture that includes an exchange model for data in the middle tier.

Migrating Point-to-Point to Many-to-Many

As the scope of enterprise integration grows, IT organizations are demanding greater efficiency and agility from their architectures. This demand has fueled a shift in the industry from point-to-

FIGURE 1



Exchange Modeling and BEA 8.1 Platform

TABLE 1

Comparative constraints	Two values share a comparative rule. For example, ship date must be earlier than received date.
Valid enumeration values	Mapping from the message format to internal systems. For example, "Fleet," "BOA" or "Wells Fargo" equals "bank."
Multifield validation checks	One field affects the rules on another. For example, if the "delivery locations" field has a value of 3, there must be 3 addresses elsewhere in the message.
Derived properties	A single value is calculated from data within the message and from other data in enterprise systems. For example, shipping costs are based on the order, selected delivery method, postal rates, and handling fees.
Merging content from multiple data sources	Rules for reconciling different values from multiple systems for the same field. For example, the systems list two different zip codes for the same client, or one lists "Main Street" and the other, "Main St."
Dynamic constraints	Constraints that are determined from an external source. For example, shipment must be denied to delinquent accounts.
Definition resolution	Mapping definitions to higher levels of abstraction. For example, "OEM" and "Distributor" equal "Customer."

Possible shared operations

point integration – which has proven to be increasingly cumbersome to build and maintain – to a many-to-many architecture that supports linear growth costs and simplified maintenance.

From EAI to SOA

Enterprise application integration (EAI) packages rely on point-to-point integration for interoperability between services and incorporate individual adapters between every pair of systems. The costs to build and maintain such an architecture become unmanageable as the number of systems grows; integrating n systems requires $n(n-1)$ adapters.

To overcome this problem, IT departments are turning towards a many-to-many approach for integrating diverse systems. This shift involves moving away from individual adaptors between every pair of services towards publishing Web services with standard interfaces on the enterprise message bus. This is the heart of an SOA.

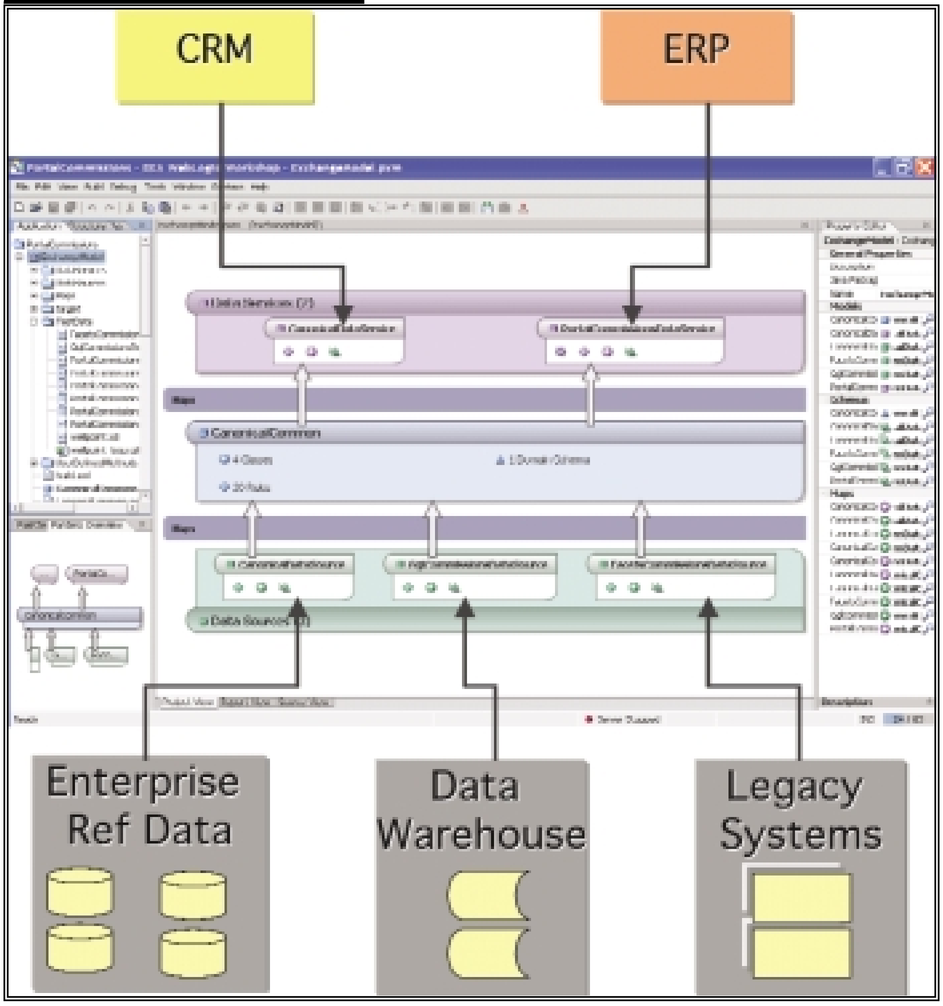
From Mapping Tools to Canonical Messages

IT organizations are making the same transition in how they architect for data interoperability. Rather than creating individual mappings between every pair of systems, organizations are implementing canonical messages for a move towards a many-to-many data architecture. Canonical messages involve publishing standard schemas and vocabularies, and all systems map to the agreed-upon models to communicate. This takes a major step in controlling the initial costs of data interoperability.

Canonical message formats represent a many-to-many approach to sharing data across integrated systems, but they fall short of a complete data exchange solution. Early attempts to implement canonical messages formats have often been rife with problems, such as forcing agreement between departments and creating rigid models that become obstacles to future growth. While they do simplify some aspects of integration, shortcomings prevent them from being the silver bullet:

- ***Requires agreement between departments:*** One of the biggest drawbacks of imposing common message formats or vocabularies across the enterprise is accounting for diversity between users. Forcing groups with different needs and goals to agree on messages can cause infighting and usually results in the lowest common denominator that inevitably hinders progress.
- ***Inflexible:*** Because all systems write to the canonical model, it is impossible to change the model without considerable cost and disruption of service. A single change in a message schema requires code in all systems to be rewritten, tested, and redeployed.
- ***Ignores interpretation and validation operations:*** Canonical messages do nothing to address the semantic differences discussed in the previous section or to make sure that the information is valid within the context of the application. Even with all systems using the identical schema for "order status," the rules that govern the usage of each field are not defined – and this ambiguity has to be handled with custom code.

FIGURE 2



Exchange Model

From Canonical Messages to Data Exchange Models

Making a many-to-many SOA architecture deliver on the promise of simplified integration requires more than Web services and canonical message formats. The missing link is infrastructure that standardizes and administers all exchange operations including data transformation, aggregation, validation, and business operations. Canonical messages that are enriched with metadata defining such operations and constituting an exchange model complete the architecture and bridge the gap between diverse services that share information.

The Real Solution: Exchange Modeling

For an SOA to deliver on its promise, the architecture needs a mechanism for reconciling the inherent inconsistencies between data sources and data targets. Exchange modeling technology goes one step further than canonical message formats by creating an exchange model that captures the semantics and usage of data as metadata, in addition to the syntax and structure. Exchange modeling fits into the BEA 8.1 Platform (see Figure 1).

Businesses spend more than $80 billion annually on integration.

$19 billion of that cost is spent on manually reconciling data between systems.

Automating the Exchange of Data Across an SOA

PANTERO

Over 40% of IT development time is spent on manually reconciling and validating the data shared between systems - now there is a better way.

Pantero's unique Exchange Modeling™ technology captures data reconciliation and validation operations as metadata and generates Shared Data Services where operations can be enforced, reused, and dynamically redeployed without incurring additional integration costs. And, Pantero is seamlessly integrated into the BEA 8.1 Platform.

Contact us today!

781-890-2890

www.pantero.com

The metadata in an exchange model is used to deploy rich sets of shared data services that formalize and automate the interaction with data across an SOA – ensuring data interoperability and data quality.

Exchange models consist of three tiers (see Figure 2): one tier each for data sources, data targets, and the intervening canonical message formats. By having multiple tiers, each department can program to its own models, and changes can be made locally without disrupting other systems. Schemas are derived directly from existing systems, enterprise data formats (EDFs), existing canonical messages, or industry standard schemas such as RosettaNet, ACORD, or MISMO. They are then enriched with metadata that define the transformation, aggregation, validation, and business operations associated with each field, class, or the entire schema.

At runtime, services within the SOA need only to access the shared data services for data, rather than having to go to multiple individual data sources via tightly coupled integration channels. During each exchange, data is fully converted and documents are guaranteed to be valid before they are submitted to back end systems.

In the "order status" example discussed earlier, a developer would create a set of rules on canonical messages sent from order management, manufacturing, and shipping systems. These conditions would fire based on the source system, the target system, and the canonical message type. In cases where a transformation is necessary, the fired condition would apply a mapping that described how to translate among the various definitions of status. Exchange models are a practical part of any SOA because:

- ***Conditions and mapping are described as metadata:*** Unlike hand code in applications, the developer can change them dynamically at runtime.
- ***Exchange operations are defined centrally:*** The transformation, aggregation, validation, and business operations are defined once and take effect throughout the enterprise without exception to eliminate redundant coding and ensure consistency.
- ***Data services are deployed locally:*** There is no single point of failure or performance bottleneck.

Conclusion

IT organizations are architecting their infrastructures to support a greater level of integration and for a greater level of flexibility to respond to an ever-changing business climate. For many, this includes a shift to an SOA for service interoperability, but an SOA does nothing to address data interoperability. Many organizations resort to adding data interoperability code to every application, and others are implementing canonical data models but run into shortcomings.

Data exchange models go further by allowing data structures and message formats to be enriched with semantic information to fully describe how to interpret and convert the information that is shared between systems. Exchange models are also tiered so that organizations have control over their own data structures and are not forced to program to common models. As an added bonus, the change management facilities inherent in exchange models create infrastructure that can be responsive to the changing needs of business.

A Dynamic Implementation Framework for SOA-Based Applications

–continued from page 8

SOA Implementation Framework Usage

The SOA Implementation Framework can be used to develop robust applications and business processes. Figure 2 presents a usage scenario that uses the SOA implementation framework to build a customer self-service application consisting of order management and customer support functions. The application uses the client SDK to connect – via the bus – to core business services in the order and support process, such as purchase order submission, third-party credit verification, order status, support case entry, and support case status. These functions are exposed as either Web services or other Java API by wrapping business logic from existing applications – SAP Order Management and a custom-built customer support application. Further, the credit service is a Web service that is hosted at a partner site.

This example illustrates the ability for client applications to easily connect to heterogeneous services in a unified and service logistics-independent manner.

SOA Implementation Best Practices

Any successful SOA implementation – in addition to using a robust and flexible SOA implementation framework – will require a set of guidelines and best practices at various stages of the development cycle. A core set of guidelines is provided below:

- The process of onboarding new services must be configuration and discovery process-driven as opposed to having a lengthy and manual process cycle. This ensures that the overall system can scale to future requirements with minimal efforts.
- An SOA implementation is only as successful as the method used to design the business services. Each service must abstract a coarse-grained piece of business functionality and is designed in a way that is coherent and reusable across the enterprise.
- Services must be designed as asynchronous where applicable to improve response times and overall user experience.
- Client applications must use a unified mechanism to access services in a protocol-independent manner and regardless of whether the services are local or remote. Further, the clients must focus mainly on the business logic and any service connectivity logic must be abstracted in the client SDK. This ensures centralized plumbing logic, high levels of application developer productivity, and ease of maintenance.
- The generic components of the SOA implementation framework must be exposed in standard ways to promote code unification and reuse.
- Use configuration where possible, instead of customization and code-based business logic. This improves the ability to scale to future business requirements with minimal effort and time.
- Reuse and wrap existing legacy business application logic with more coarse-grained business-level services. Avoid rewriting legacy implementation logic.

Conclusion

This article presented a flexible, dynamic SOA implementation framework for building service-oriented applications and business processes of the future. This approach aims to provide enterprises with a highly scalable and dynamic framework for defining, configuring, accessing, and managing enterprise services.

diagnoSys

Are your Web applications open 24 hours? Get the tips and best practices you need to keep your apps running at peak efficiency 24 hours a day with WebApp News, the free newsletter from the performance experts at H&W.

Sign up today at:
www.hwcs.com/wldj08.asp

You might as well be closed for business if your Web applications aren't running at top speed 24 hours a day. With poor Web performance costing businesses $25 billion a year, you need to find problems, and you need to find them fast.

You need DiagnoSys.

DiagnoSys is the first intelligent performance management solution that analyzes your J2EE and .NET applications from end to end. DiagnoSys starts analyzing quickly, gathers data directly from all application resources, and proactively finds and helps fix the real cause of problems – before they cost your business big time.

So when it's absolutely essential that your Web applications say "Open 24 hours," trust DiagnoSys.

H&W
www.hwcs.com | 1.800.338.6692

Bridging the Enterprise Computing Gap For 25 Years

So You Want an SOA with Web Services

BEST PRACTICES FOR MIGRATING TOWARD SERVICE ORIENTATION IN THE ENTERPRISE

BY **ERIC NEWCOMER**

AUTHOR BIO...

In his role as CTO at IONA, Eric Newcomer is responsible for directing and communicating IONA's technology roadmap, as well as IONA's product strategy as it relates to standards adoption, architecture, and product design. Eric was a founding member of the XML Protocols Working Group at W3C, which produced SOAP 1.2. Eric is coauthor and editor of the Web Services Composite Application Framework (WS-CAF) and is the author of *Understanding Web Services* (Addison-Wesley), and coauthor of *Principles of Transaction Processing* (Morgan Kaufman).

CONTACT...

eric.newcomer@iona.com

Imagine, if you will, the following exchange:

CIO: You know, I've been reading a lot about this whole service-oriented architecture thing.

WebLogic Developer: And…?

CIO: It sounds pretty cool, I really think we need one.

WebLogic Developer: OK…sure, we'll get right on that.

There is no question that service-oriented architecture (SOA) is quickly becoming one of the hottest trends in enterprise computing. IT departments are inundated weekly, if not daily, with the claims and marketing messages of vendors announcing myriad technology and service offerings that will magically transform the way business gets done.

At the same time, the growing acceptance of service orientation and SOA as enterprise computing methodologies is driving a profound shift in how organizations view their IT infrastructures. Replacing complex, monolithic applications with nimble applications built from exposed services promises increased developer productivity, greater flexibility, and ultimately, reduced cost. The adoption of Web services and SOA can also remove a significant level of complexity and integration problems from enterprise application development projects.

But, as with any large-scale project with the potential for significant impact, IT departments must have the right plan and the right resources in place to ensure a successful transformation of their computing infrastructure. The biggest stumbling block facing most IT organizations is not whether to move toward an SOA, but determining the best approach and appropriate level of investment for doing so.

Defining the Service-Oriented Architecture

A service-oriented architecture (SOA) is a blueprint that guides all aspects of creating and using business services throughout their life cycle (from conception to retirement), as well as defining and provisioning the IT infrastructure that allows different applications to exchange data and participate in business processes regardless of the operating systems or programming languages underlying those applications. With the advent of Web services, it has become easier to accomplish this without becoming tied to any one specific approach (data-oriented, message-oriented, object-oriented, procedure-oriented, etc.)

The primary goal of any SOA is to help align IT capabilities with business goals. Migrating to an SOA helps organizations streamline the design and development of solutions, makes it easier for remote teams to collaborate, and enables reuse through module reconfiguration and repurposing.

Above all, SOA is an approach to building IT systems; it is not a single technology or a silver bullet solution. SOA is a methodology whereby business services (i.e., the services that an organization provides to clients, customers, citizens, partners, and other organizations) are the key organizing principles that help align IT systems with business services. Most importantly, the services are defined in a manner that is independent of the underlying technology. In contrast, earlier approaches to building IT systems that relied on approaches such as object orientation, procedure orientation, and message orientation resulted in systems that were tightly coupled to the capabilities of a particular technology such as CICS, IMS, CORBA, J2EE, COM/DCOM among others, and were therefore less generic to the business. Development following these methodologies and approaches often resulted in the monolithic, stove-piped applications found in the majority of today's IT systems.

Service orientation with Web services reduces enterprise integration project cost and improves project success rates by adapting technology more naturally to the people who need to use it, rather than focusing (as previous generations of IT systems have) on the technology itself, which forces people to adapt to the technology. The major difference between service-oriented integration and the other approaches is that service orientation lets you focus more on the description of the business problem, while the other approaches require you to focus more on the use of a specific technology to solve a particular integration problem.

Technology-oriented approaches are limited, by definition, to the capability of a single product or technology. The service-oriented approach with Web services is not based on the capabilities of a single technology, technology type, or product, but on designing and deploying services capable of executing on any technology type. Developers of service-oriented integration projects, therefore, can focus more on the business problem than on the technological problems, since Web services are widely available as interfaces to nearly every technology type. The right technology can still be chosen for the execution environment, but to the service consumer, the execution environment technology is irrelevant since they are dealing with the Web service description rather than the execution environment.

As illustrated in Figure 1, the eventual goal of designing, developing, and implementing an SOA is to provide new and better ways for applications to talk to each other, and to integrate with a business process engine, and multiple client types. The diamonds on a stick represent the Web service interfaces, which are either developed from scratch with, for example, new Java code, or retrofitted using a Web services infrastructure product. Once developed, the Web services are accessible using the common SOA service transport. Any new application or new client wishing to use the service obtains the description from the service repository. A business process engine (often called an orchestration engine) can also access the Web services after they are designed and developed to automate flows.

Businesses that successfully implement a service-oriented integration approach using Web services to obtain these benefits are likely to have a competitive advantage over those who do not, since those who have services aligned with strategic IT business goals can react more quickly to changing business requirements than those who have IT systems aligned to a particular technology orientation.

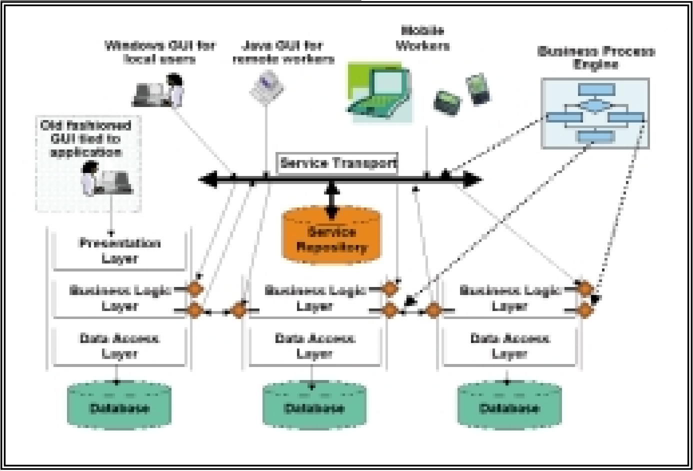
The concept of SOA isn't new, but what is new with Web services is the ability to mix and match execution environments, separating the service interface from the execution technology, allowing IT departments to choose the best execution environment for each job (whether it's a new or existing application), and tying them together using a consistent architectural approach.

Implementing the SOA

Once you have decided that service orientation and SOA with Web services is the direction in which you want to move your IT systems, you need to determine the best way to accomplish this. Even if you are a dedicated WebLogic/J2EE shop, Web services are still the best means for implementing an SOA because they are based on a set of technology standards that are independent of any particular software domain, development process, or design method. Web services, therefore, have the potential to significantly extend the reach of the WebLogic platform.

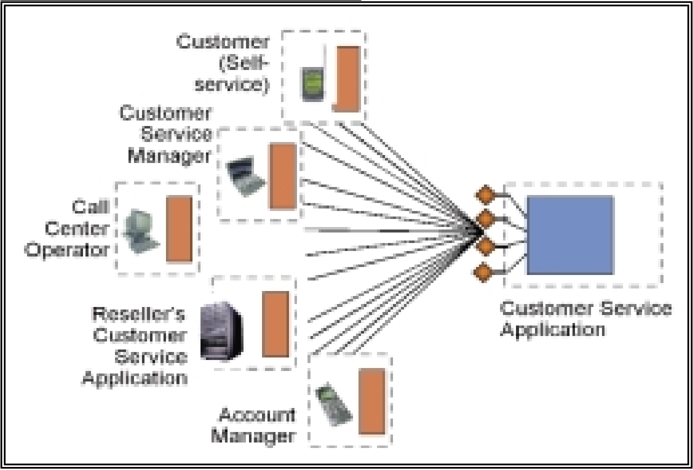
BEA WebLogic can easily be used as the design center and for the development of new code where it's needed. It can also be used to tie together a wide variety of existing systems using Web services. When existing systems are exposed through Web service interfaces,

FIGURE 1



Benefits of implementing an SOA

FIGURE 2

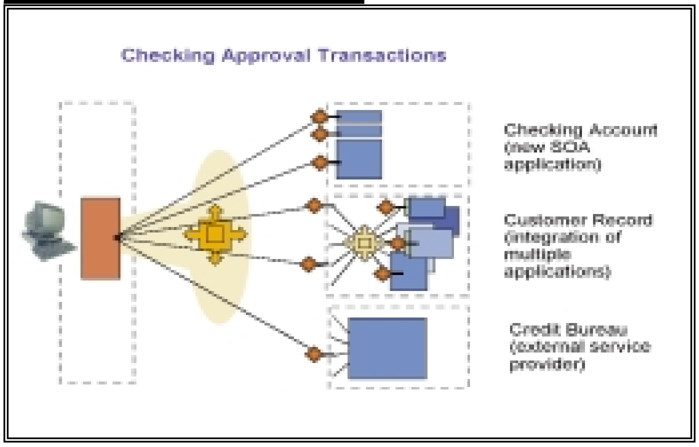


Using SOA with Web services to enable multi-channel access they can easily be combined into powerful business process flows and new applications.

Web services can be applied in a variety of ways to address a number of problems. SOA brings architectural vision, development guidelines, and design methods that increase the likelihood that Web services will be applied in a strategic manner to add long-term value to the organization though agility and reuse.

Figure 2 illustrates the distinct benefit of using SOA with Web services for multi-channel access to the same application. In this example, an existing customer service application is Web service enabled using an infrastructure product, and accessed as a reusable service from the account manager's cell phone, the customer's mobile device, the customer service manager's laptop, the call center operator's PC, and the reseller's service application. Using WebLogic in conjunction with a Web service enabler such as Artix extends the reach of existing applications to new applications and devices. Web services contribute to agility because they can be designed and reused across multiple business processes so that any user can access them at anytime from anywhere and using any device. Web services are reusable when they are designed and implemented according to enduring business themes rather than being tightly coupled to a particular implementation.

FIGURE 3


Checking Approval Transactions
Checking Account (new SOA application)
Customer Record (integration of multiple applications)
Credit Bureau (external service provider)

Varying services from coarse to fine grained

Long-Term Value

The real value of SOA comes from the later stages of deployment, when new applications can be developed entirely, or almost entirely, by composing existing services. When new applications can be assembled out of a collection of existing, reusable services, the best value for effort can be realized (that is, the lowest cost and fastest time to results). But it takes some time to reach this point, and significant investment in service development.

It's more likely that new applications will be constructed out of some number of reusable business services (such as order item, check credit card authorization, process invoice, etc.) and some number (hopefully smaller) of custom services developed specifically for the application (such as helping the customer decide on a purchase of remaindered books). <y company's customer field experience using SOA with CORBA indicates that a reuse level of about 70% is achievable, and the industry could expect the same, or possibly better, with Web services.

In general, it is easy to understand the benefit of reusing common business services such as customer name lookup, ZIP code validation, or credit check. In what you could call the pre-service-oriented development environment, these functions might be performed by reusable code libraries or objects loaded/linked into new applications and thus reused (although duplicated). In SOA-based applications, common functions such as these, as well as typical system functions such as security checks, transaction coordination, and auditing are instead implemented using external services. Using services external to the application not only reduces the amount of deployed code; it also reduces the management, maintenance, and support burden by centralizing the deployed code and managing access to it.

A New Development Approach

As many readers are aware, the software industry has widely adopted the paradigm of service-oriented development as the way in which to best take advantage of the power of Web services for application integration. However, it needs to be clear that service-oriented development truly implies a new, complementary approach to the object-oriented, procedure-oriented, message-oriented, and database-oriented paradigms that preceded it. While service-oriented architecture isn't new, its availability as an abstract Web services layer to any executable software system is, and therefore, service orientation is emerging as an important new development model that needs to be understood and adopted as it applies to a uniform service abstraction mapped to any or all of these prior technologies.

Developing a service is different from developing an object. A service is defined by the data it shares using messages exchanged with other services, not by the definition of a common method/argument signature. A service must be defined at a higher level of abstraction (some might say at the lowest common denominator) than an object since service definitions are mapped to a procedure-oriented language such as COBOL or PL/I, or to a message-queuing system such as JMS or MSMQ, in addition to an object-oriented system. Because a Web service needs to be able to operate across all of these technology domains, its development requires some study and new thinking.

On the development side, it's necessary to understand the granularity at which the service is to be defined. Web services normally require a larger-grained interface that accepts more data in a single invocation than an object. With Web services, however, it's possible to create an aggregation of Web services such that the published Web service encapsulates multiple other Web services. This allows a coarse-grained interface to be decomposed into a number of finer-grained services. The coarse-grained service may make more sense to publish while the finer-grained services may make more sense as "private" Web services that can be invoked only by the coarse-grained Web service.

Figure 3 illustrates another major benefit of using Web services for an SOA: the ability to create a composite application. The WebLogic platform shown on the left side of the diagram can be used to access the Web services shown on the right side, which represent a mixture of services developed using new Java code and services enabled using other technologies. The customer record service exposes two services that are composed of other services.

On the project level, it's necessary for an architect to oversee the development of reusable services, and identify a means to store, manage, and retrieve service descriptions when and where they are needed. The reusable services layer insulates business operations such as "get customer" or "place an order" from variations in the underlying software platform implementations, just as Web servers and browsers insulate the World Wide Web from variations in operating systems and programming languages. The ability of reusable

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 to learn how we can help you address the challenges of your operational monitoring and management.

netiQ® Work Smarter®

services to be composed into larger services easily and quickly provides the organization the benefits of process automation and agility to respond to changing conditions.

Developing the SOA

Two major phases of activity are required for developing a well-understood, managed, and deployed SOA project based on Web services. These phases center on identifying services that must be:

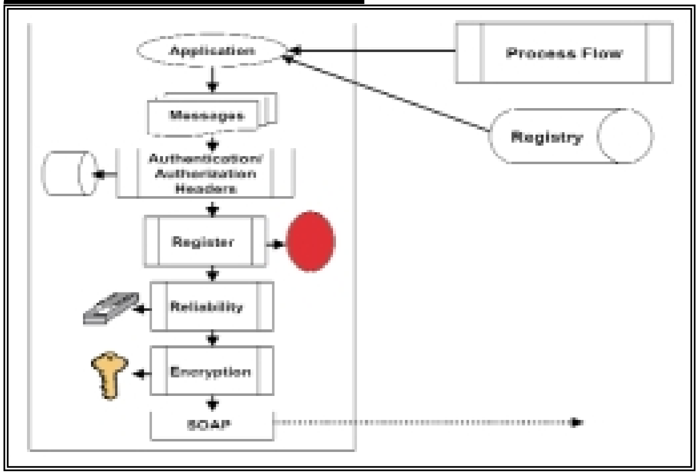
- ***Developed using new execution environments:*** Require the development of new application logic in Java classes and EJBs to run in WebLogic application server containers.
- ***Enabled from existing application logic:*** Require the use of a Web service–enabling toolkit adapted for use with the existing technology, and are integrated either directly with the new application code using Web services invocations, indirectly the WebLogic integration server, or both.

A combination of tools may be required to enable all the Web service endpoints to be included in the SOA. For example, IONA's Artix could be used to enable Web services for TIBCO, MQ Series, CICS, IMS, CORBA, and other existing systems, and integrated with new logic developed using WebLogic. Alternatively, Web services capabilities may exist in newer versions of these products, and it may make sense to expose or enable the existing applications by installing or upgrading the existing software to a newer version that supports Web services and use them, instead. However, this approach is vulnerable to potential incompatibilities among multiple Web services implementations, and the complexity of managing multiple Web services products.

A variety of Web service–enabling toolkits are available through independent vendors and open source projects. Therefore, one of the key decisions in your SOA implementation is dealing with the combination of technologies that will be involved. In an enterprise-wide SOA project, it's typical that several different, often widely varying, technologies will be involved, with varying qualities of service.

In developing services for an SOA, first you decide that you need a service, then design the service to be compatible with the other services you are developing. And finally, as illustrated in Figure 4, you decide among the possible additional quality of service requirements for reliability, security, and transactions you might need. It's necessary to check the Web service products you are using to ensure compatibility not only at the basic SOAP and WSDL level, but also at the level of these extended features.

FIGURE 4



Potential quality of service requirements

The starting point for service-oriented development is to identify the data to be shared. That becomes the XML Schema representation of the data types and structures to be contained in the message. Web services provide two basic styles of interaction:

- ***Document oriented:*** The message is structured as a "plain" XML document (in other words, the data is just laid out in a way that makes sense for XML).
- ***RPC oriented:*** The message is structured as a series of arguments to be passed to an object or procedure method, typically matching the data types and structures.

In the first case, messages are typically used to carry business documents such as purchase orders, invoices, and bills of lading. This style of interaction is very compatible with asynchronous message queuing systems such as MQ Series, MSMQ, JMS, TIBCO, IMS, and others. It is often used for business-to-business transactions that depend upon the exchange of large amounts of data to be processed in an execution flow that might take hours or days to complete, resulting in an executed document or a new document to be passed back to the originator. This style is also the most abstract since it says nothing about the nature of the signature on which the message is to be mapped within the execution environment. The RPC-oriented style, on the other hand, assumes that the data in the message is to be executed by a procedure or object with a defined interface, and the RPC-oriented formatting of the XML in the message is designed to make it easier to map into and out of such existing constructions.

However, like everything else in computing, there is more than one way to accomplish the same task. Most of the interoperability issues arising from varying Web services toolkits and products stem from incompatibilities in mapping the complex data types and structures such as arrays, records, and structures. In other words, the more simple the data type, the more likely you are to achieve interoperability.

As a rule, Web services products are more compatible the more general their use – for example, the WS-Interoperability Basic Profile recommends using the doc literal encoding style (i.e., the plain XML schema data types and structures) rather than the SOAP encoding style (which basically defines new data types and structures that are more compatible with existing RPC-oriented technologies). The doc-oriented interaction style is more abstract than the RPC-oriented style, since the SOAP processor and XML parser pull out the data of interest to the application and pass it to the queue or object. When it's data formatted for an RPC, knowledge about the types in the arguments is necessary to be able to decode and understand the message.

A necessary step in the design and development of any SOA is the identification of requirements for extended technologies. WS-Security, WS-ReliableMessaging, and WS-Transactions, among other specifications, have been defined to address these requirements, but their availability in WebLogic and other Web services tools, and the Web services interfaces to existing products, is limited, as is the interoperability across the extended feature set. The more extended the features you use, the harder interoperability is to achieve.

Facing Challenges

SOA-based integration provides a consistent way to access all applications within a company, and potentially outside the company. The challenge represented by implementing an SOA is that some applications may need to be modified in order to participate in the SOA. In addition, the definition of a reusable service is very difficult to get right the first time.

The largest barriers to adoption of SOA tend to be ensuring adequate staff training and the fact that implementing an SOA requires a disciplined level of investment for the future. Any technology, no matter how promising, can be abused and improperly used. Services have to be developed, not simply for immediate benefit, but also for long-term benefit. Unlike objects or databases, a service is developed for use by its consumer, which may not be known at the time. To put it another way, the existence of an individual service isn't of much value unless it is part of a larger collection of services that can be consumed by multiple applications, and out of which multiple new applications can be assembled. Any collection of services needs control, design, and architectural purpose since they are typically not all developed at the same time.

The main challenge of moving to an SOA is managing short-term costs. Building an SOA isn't cheap; reengineering existing systems costs money, and the payback becomes larger over time. It requires including business analysts to define the business processes, systems architects to turn processes into specifications, software engineers to develop the new code, and project managers to track it all.

Of course, incrementally adopting SOA and leveraging it where it will have the greatest business impact can amortize these costs when services can be used to solve tactical problems first. Part of adopting Web services and SOA, therefore, is to identify those projects which return immediate value by solving an immediate problem (such as integrating J2EE and .NET Framework applications), but also lay the foundation for strategic value in the future through reuse, such as creating a new order entry application more quickly because it can reuse services for credit checking and billing.

Another challenge of realizing the SOA vision is that current applications are tightly coupled while the underlying SOA technology is loosely coupled. Therefore, it may not be possible – at least not in a cost-effective manner – to immediately achieve loosely coupled perfection given the existing environment of tightly coupled, business-critical applications. It may be necessary to start with fine-grained, tightly coupled Web services to meet immediate project requirements, but this should only be done within the context of an overall long-term plan to develop coarse-grained services that encapsulate the fine-grained services.

Conclusion

So, you want an SOA built on Web services in a WebLogic environment. If this is a task you and your IT department are facing, the good news is that it can be done. With the proper expectations, strategy, planning, and execution, you can start now to migrate your enterprise to take advantage of this computing methodology today and well into the future. It's important to take into account the difference between Web services that require new Java code, and those that expose the capabilities of existing applications. Using WebLogic in conjunction with enterprise Web services infrastructure products can help extend the reach of WebLogic-based SOA projects by more easily handling the Web services based on existing application code, whether on the mainframe, Unix, or an established middleware system.

FORMULA ONE
Enterprise Reporting Toolsets for Java Projects & Portals
ReportingEngines
JAVA REPORTING TOOLS FROM ACTUATE

TROUBLE SEEING BEA PORTAL DATA?

NEW!

BEA PORTAL REPORTING SOLUTIONS

POWERED BY THE FORMULA ONE E.REPORT ENGINE

100% Pure Java: API-driven toolsets for building queries and creating reports against BEA Portal Server events.

Visual Access to BT_EVENT: Point-and-click wizards streamline report creation against the complex structure of BEA Portal Event and Behavior Tracking Data.

Deliver Reports Through BEA Portal Server: Embed reports that inherit the server features (security, connection pooling, etc.) of your BEA platform. No external report server to set up or maintain.

Analyze Server Events: Determine user traffic patterns. Find the most and least popular items in your Portal applications. Calculate click-throughs on marketing campaigns. And more!

Flexible: Available as an integrated WebLogic Workshop Extension or as a set of JAR files for use with JBuilder, IntelliJ IDEA, and other Java development environments.

888-884-8665 • sales@reportingengines.com • http://www.reportingengines.com/products/beaportal/overview.jsp

Service on Demand Portals

A PRIMER ON FEDERATED PORTALS

BY RAJUL RANA & SAI KUMAR

Enterprises are moving towards a highly collaborative environment to achieve higher competitive advantage. Availability of the right information across the enterprise at the right time has become a key capability to provide such an advantage. Though this was a well-understood objective, various architectures that evolved to manifest such an enterprise information delivery infrastructure were not elegant, intuitive, or aligned to governance and organizational dynamics.

With the emergence of service-oriented architecture (SOA) and Web Service for Remote Portals (WSRP), the enterprise information bus (EIB) topology has evolved to be the infrastructure for service on demand portals.

This article, which the first of two, provides a comprehensive introduction to federated portals, WSRP, and EIB.

Federated Portals

A few years back, in a spree to go "online," organizations started building individual Web sites that evolved into siloed portals. Time to market, inadequate product maturity, and budgetary constraints are just a few reasons why organizations find themselves with a large number of siloed portals.

Today, enterprises are becoming more complex, distributed with increasing lines of businesses, business processes, and siloed portals, while at the same time having the need to portray a unified face, in the form of an enterprise portal, to their customers. Though this means consolidation, they still want to protect their investments in existing portal-based applications and leverage them.

The consolidation of different portals, information islands, and business processes is not only a challenge, it is also simply unmanageable. It is practically impossible to collate information and data from different lines of business portals into a centralized enterprise portal for a variety of reasons. Rationalization and consolidation of a multitude of portals into a true enterprise portal, which is a common platform for delivery of user-interactive applications, is a task that remains high, but elusive on the enterprise wish list today.

The challenge of realizing a true enterprise portal becomes manifest when it is important to consider the governance, information ownership models, and trust within an enterprise for information life-cycle management.

Yesterday's portal proliferation and the need to leverage that, combined with today's need for a unified enterprise face, poses unique challenges in creating an information delivery framework for large enterprises. The solution lies in fabricating a highly distributed yet unified portal framework that can accomplish:

- Standards based information interoperability
- Loose coupling with manageable integration

The availability of the right technology and standards is important to realize truly distributed portal information domains and yet provide a seamless integration and delivery of information across enterprise.

Recent Standards in the Portal Space

Two complementary industry standards that are emerging in the portal space are:

- ***JSR 168:*** An industry standard that defines a standard way to develop portlets. It allows portlets to be interoperable across portal ven-

AUTHOR BIOs...

Rajul Rana is a senior architect with MphasiS Corporation (www.mphasis.com), a global IT services organization. Rajul leads the portal practice at Mphasis, and has architected several large, successful e-business and portal projects for Fortune 500 companies, built on a variety of products. His areas of interest span J2EE technologies, portal, content management, and SOA.

Dr. Sai Kumar is a senior architect with Mphasis Corporation, where he heads the Web services and SOA practice. He has provided strategic consulting in SOA to various companies and has architected several enterprise solutions for financial, health care, and retail industries.

CONTACT...

rajul.rana@mphasis.com
saikumar@mphasis.com

dors; e.g., portlets developed for, let's say, BEA WebLogic Portal, can be interoperable with IBM Portal. This will allow organizations to have a lower dependency on the portal product vendor.

- ***WSRP (Web Service for Remote Portals):*** Allows remote portlets to be developed and consumed in a standard manner and will facilitate federated portals.

JSR 168 complements WSRP by dealing with local rather than distributed portlets. As shown in Figure 1, a portal page may have certain local portlets which are JSR 168 compliant and some remote, distributed portlets that are executed in a remote container.

With JSR 168 and Web Service for Remote Portals (WSRP) maturing, the possibility of true enterprise service on demand portals has become a reality.

WSRP has combined the power of Web services and portal technologies and is fast becoming the major enabling technology for distributed portals in an enterprise. WSRP will lead to a new era that will allow enterprises to provide one face to the users resulting in consistent user experience, with unified information delivery. At the same time, WSRP will allow all various applications to remain distributed.

Introducing WSRP

Briefly, WSRP is a standard that enables a "portal based" Web application to easily consume services from any number of distinct providers on behalf of its end users (portal users), and to present this information to them with minimum integration effort. It allows dynamic binding to remote portlets without any installation or code running locally on the portal server

WSRP offers the following benefits:

- ***Reusable Presentation Tier:*** Presentation delivered with Web service, not just data
- ***Interoperability:*** Widely accepted standard supported by large players in the industry
- ***Portability:*** Can potentially be used with both J2EE and .NET platforms (although Microsoft has yet to formally announce their WSRP implementation).

WSRP builds upon the original Web services vision and uses the same underlying technologies, such as SOAP, WSDL, and UDDI. WSRP services can be published, found, and bound in a standard way. The main difference between WSRP and Web services is:

- Web services are data oriented, whereas WSRP is presentation oriented and interactive by nature.
- Web services return "raw" XML-based data, often quite complex, which the application has to transform into the HTML pages and additionally maintain state in a multipage (multistep) interaction. WSRP returns ready-to-use HTML markup fragments that the consumers can embed directly, without any further processing.

Consider using WSRP when:

- Integrating user-facing, presentation-oriented and visually rich functionality
- Integrating interactive application with complex flow: Multistep, multipage user interaction
- You want to reuse a user interface and look-and-feel tier
- Host services in a central environment best suited for execution, allowing maintained control over format and presentation of the content.
- Sharing code at the portlets level across different areas within the organization: reimplementation of the presentation layer on each portal is avoided.

Federated Portal Architecture

Let's consider that an organization has several existing departmental portals. Each of them could be on different technology and products such as Vignette Portal, WebLogic Portal, or even .NET (SharePoint) technology. Some of them may not even be on a portal product. Each portal may or may not have same look and feel or navigation, but all departmental portals participate under a central SSO umbrella. Each departmental portal is self-contained and provides unique functionality specific to its product or services.

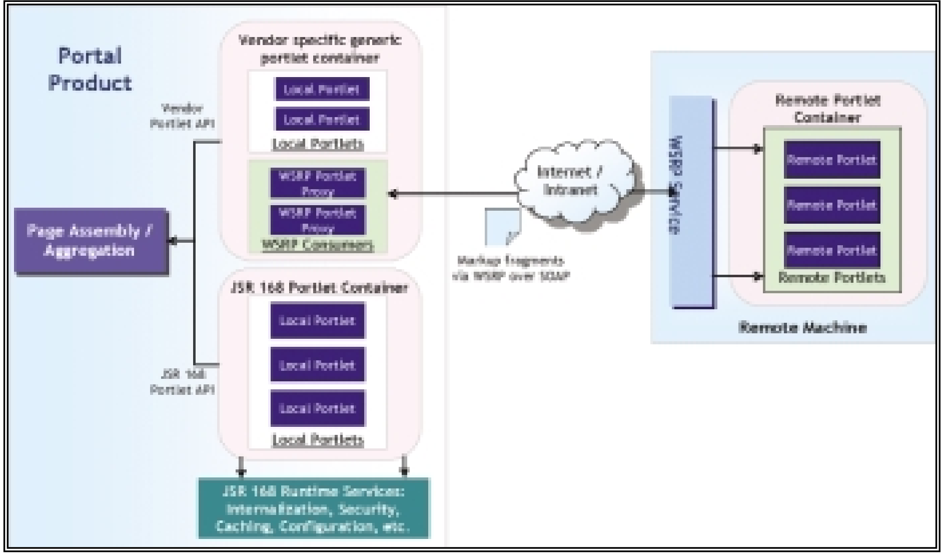
Now, let's consider that the organization wants to provide a centralized face to the users, instead of users having to access a multitude of applications. There are main two architectural options:

1. ***Monolithic enterprise portal model:*** Migrate the existing applications and create one large main portal that has all the features. This is practically impossible, as it is costly and time-consuming. Also, there could be a technology limitation in migrating heterogeneous applications to a common technology platform.
2. ***Federated portal model:*** Create a main portal that serves as an entry point or gateway to the enterprise. The main portal is a thin layer that sits on top of the individual departmental portals and leverages them.

A federated portal allows organizations to provide a common entry point, but, at the same time, independence, to individual departments to develop, maintain, control the release schedules, etc. Initial implementation of federated portals integrated the main portal with departmental applications or portals using traditional presentation layer-based integration techniques, such as:

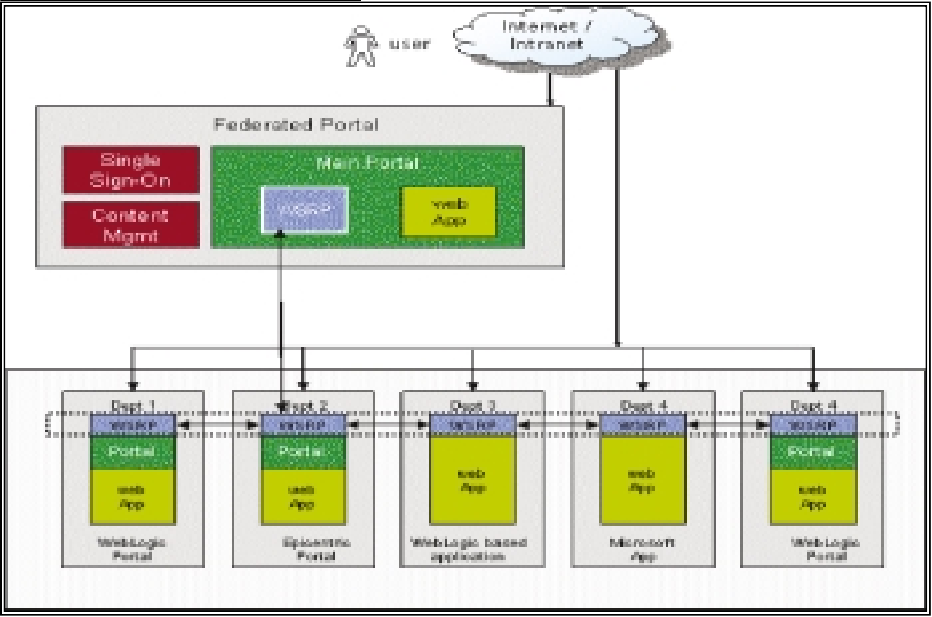
- ***New browser window:*** A link can be provided on the main portal and, on click-

FIGURE 1



Portal page with JSR-168–compliant portlets

FIGURE 2



Federated portal architecture

ing to this link, the target application can be opened up in a new window.

- ***Frame-based integration:*** Target application can be integrated as an HTML frame into the portal.
- ***Screen scraping:*** In this case, applications can be integrated into the portal by screen scraping the target Web application.

These techniques, however, present the following challenges and limitations:

- ***Application integration issues***
  - Session Management issues (timeout synchronization, etc.)
  - Propagating logoff requests to all active applications
  - Security around application launch (as departmental applications need to be directly accessible from the DMZ)
  - Window management (closing of windows user logs off

- ***Management issues***
  - Complicated new on-boarding application process; will have to develop custom application registry.
  - Each application will be required to have its own set of Web servers (and DMZ) and application servers.
  - Cost, administration, and version issues
  - Each Web server is required to have an agent to participate in an SSO environment.

- ***Over time, managing (version upgrade, installation, etc.) a large number of agents can be problematic.***

WSRP presents a more elegant, service-oriented approach to implementing federated portals. Service-oriented architecture provides flexibility and reuse of core service components. Service components traditionally are reusable business and infrastructure components in nature. A truly federated portal needs to integrate with transactional services and information across the enterprise, provided by various specialized portals and service infrastructures. With WSRP as the underlying technology, portlets are designed, built and exposed as reusable Web services on an enterprise information bus (EIB).

WSRP natively supports portlet concepts, such as modes (view, edit, help, preview, and custom), window states (maximized, minimized, solo, custom), portlet preference management, etc. This makes it much easier to integrate applications in a portal paradigm. Further, WSRP addresses some of the common integration issues (many of the above issues), such as transparent session management, timeout handling, caching, support for different markups, etc. It integrates applications as services rather then fragile links. Moreover, since WSRP is built over the SOAP stack, it can leverage enterprise Web service management infrastructure, including service metering, routing, prioritization, and lifecycle manageability, e.g., versioning, monitoring, and upgrade. With services and information exposed as Web services, they can also be discovered from a UDDI registry and secured using Web service security standards. For transport-level security, WSRP can be used with SSL.

All of these features, put together in an industry-standard manner, make WSRP an ideal technology for building federated portals.

Federated Portal Model Using WSRP

The main portal acts as a facade (see Figure 2), serving up a common logon and home page. It authenticates users against the central Single-Sign-On (SSO) infrastructure to display an entitlement-driven home page. The home page is composed of multiple portlets that display information from various departmental applications. Each portlet acts as a WSRP consumer that interacts with the WSRP producer on the other end. Thus, departmental services are integrated seamlessly into the main portal, using WSRP. To achieve this, the departmental application services are exposed as WSRP services. This requires adding a WSRP layer (specifically WSRP Producer) on top of existing application, which is much easier than rebuilding the application in the main portal.

The responsibility of the main portal (WSRP consumer) is limited to:

- Content/application service assembly.
- Portal (top Level) personalization and customization.
- Authentication.
- High-level entitlement: The main portal controls the access to the portlets on the home page and therefore access to the services offered by the departmental portal.

The departmental portals (WSRP Producers) continue to provide:

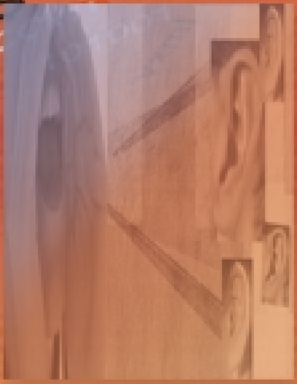
- Business logic
- Content rendering
- Portlet-specific customization and personalization
- Entitlement checks: Each producer should also check entitlements for access control, fine-grained entitlement check.

Another level of sharing between multiple portals is to integrate the touch points, such as directory, security, personalization profiles, metadata and portal components, which isn't easy. Issues to consider in this area:

–continued on page 38

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.

Download
30 Day Free Trial
www.sandcherry.com

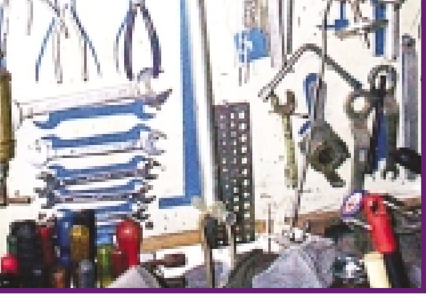
For additional information:

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

SandCherry



Server Configuration

TASKS MAY BE MADE EASIER WITH WLST

BY **SATYA GHATTU**

AUTHOR BIO...

Satya Ghattu has been with BEA Systems for more than three years as a senior software engineer focusing mainly in the OAM (Operations Administration and Management) area. Prior to BEA he worked as a Java developer/DBA for multiple consulting companies.

CONTACT...

sghattu@bea.com

WebLogic Server Scripting Tool (WLST) is a command-line scripting tool that BEA intends to support as part of the WebLogic Server 9.0 release. This tool is available today on BEA Systems' dev2dev Web site and works with BEA WebLogic Server versions 7.0 and 8.1, including all service packs.

Background

There has always been a need for WebLogic Server to support a scripting solution that allows developers and system administrators to perform both simple and complex changes to their WebLogic Server configuration reliably and interactively. Currently, WebLogic Server supports three command-line tools that require a running WebLogic Server instance: weblogic.Admin, weblogic.Deployer, and the WLConfig Ant task tool. Weblogic.Admin and WLConfig are used to make configuration changes and interrogate MBeans, whereas weblogic.Deployer is used strictly for application deployment. There are also other non-supported tools with which users are familiar such as WLShell, Config2Admin, and MBeanExplorer, etc.

There are many use cases where a simple scripting tool would be handy. For example, developers and administrators may need to write scripts to configure a server and want to enhance it with additional configuration elements via loops and other flow-control constructs. A use case like this would propel the need for a scripting language interpreter that can read user scripts and then run them against the WebLogic Server.

WLST makes an attempt to solve some of the challenges that our current WebLogic Server developers and administrators face. These include capturing repeatable configurations primarily aimed at the preparation of an environment (applications and dependencies) as part of a larger, deployment action; sharing complex configuration changes between users at one level of the life cycle or between levels (e.g., between development and staging); and applying the same change, iteratively, across multiple nodes of a topology, or at some other scope. Adding support for user-defined scripting to WebLogic Server greatly enhances the usability, resulting in higher user satisfaction. With WLST, users are able to customize WebLogic Server to suit their needs, ease their routine tasks, and extend added functionality. Repetitive tasks and complex procedures can now be simplified by providing scripts to handle them. With the help of scripts, components can be "glued" together to form applications. The scripting language augments the Java language and supports the accelerated development.

Introduction

The WebLogic Server Scripting Tool is a command-line scripting interface for WebLogic Server. The WLST scripting environment is based on the Java scripting interpreter, Jython. WLST lets you make use of the provided WebLogic Server scripting functions as well as common features of interpreted languages. These include local variables, conditional variables, and flow control statements. WebLogic Server developers and administrators can extend the WebLogic

Server scripting language to suit their environmental needs by following the Jython language syntax.

Jython is an implementation of the high-level, dynamic, object-oriented language Python seamlessly integrated with the Java platform. Jython is 100% Pure Java and is easy to learn with a clear and simple syntax. One of the main advantages of Jython is that you can call any Java object from within the Jython interpreter. What this means for a user is that he/she can reuse any Java code or Java tools that he/she had written. I won't delve too much into details about Jython for this article. For more information about Jython, check out www.jython.org.

WLST Flavors

WLST comes in two flavors, offline and online. Offline is used when a user isn't connected to any WebLogic Server instance and is configuring the domain by interacting with the domain file and configuration templates. This functionality is similar to the Configuration Wizard silent mode scripting that is being deprecated in WebLogic Server version 9.0. Online WLST is utilized when a user is connected to a running WebLogic Server instance and makes changes to the configuration artifacts or monitors the runtime data. Today, both online and offline WLST are available as two separate downloadable JARs. For WebLogic Server version 9.0, the functionality of online and offline WLST will be folded into one tool.

Modes of Operation

WLST provides three modes of operation: interactive, script, and embedded modes. In the interactive mode, the users enter commands and view the response on a command-line prompt. This mode is useful for learning the scripting tool and its capabilities. It is also useful for prototyping command syntax to verify the options before building a larger script. An example of an interactive session is shown in Listing 1. Script mode lets the user supply a set of script commands in a file that the tool executes as a program. Listing 2 shows a simple script that can be executed as a program on WLST. Embedded mode is where a user embeds the WLSTInterpreter in a Java application (see Listing 3).

Features

WebLogic Server implements the Java Management Extensions (JMX) 1.0 and all subsystems (JMS, JDBC, Security, etc.) are instrumented as MBeans and contain attributes by which they can be configured, monitored, and managed. All of these MBeans are arranged in a hierarchical fashion with the DomainMBean as the parent to all of the Configuration MBeans and the DomainRuntimeMBean as the parent to all of the Runtime MBeans. A user can navigate to all of the MBeans via WLST by invoking different WLST commands. For simplicity, the navigation is loosely based upon the file notation in any operating system.

Navigation

In OS terms, WLST has three drives: config, runtime, and custom. The config drive hosts all of the Configuration MBeans with DomainMBean as the parent. The runtime drive hosts all of the Runtime MBeans with DomainRuntimeMBean as the parent, and the custom drive hosts all of the Custom (non-WebLogic) MBeans that a user registers in the WebLogic Server. The MBean types or instances are like the directories in which a user can "cd" or navigate into these MBeans to interrogate the attribute names and values or operations (i.e., files). Listing 4 shows an example in which a user navigates to different drives and MBeans to list the attributes and values.

Configuration

A user can easily move resources from one domain to another by using the "configToScript" command. This command takes a config.xml and converts it into a WLST script that can be applied on another domain to create a similar configuration. It also creates a properties file that contains name-value pairs that can be changed to create configurations with different domain names, server names, etc. Listing 5 demonstrates a simple example where a config.xml is converted into a WLST script. The user can also create new configuration information, and retrieve and change existing configuration values that are stored in the domain config.xml file or in a domain template JAR created using Template Builder. Listing 6 demonstrates a simple example where a user reads an existing medrec template to create a medrec domain.

Current Managed Object (cmo)

Whenever a user navigates to an MBean instance, he has access to that MBean object via the cmo variable. This variable holds the MBean's proxy object so that a user can invoke all of the methods or operations that the MBean interface exposes. Listing 7 demonstrates an example where this variable can be used.

Java from WLST

If you have written any utility command-line tools in Java and would like to reuse these in WLST, you can certainly do so. For example, I wrote a simple Java program that queries the MBeanServer and retrieves all of the MBeans that are registered in this server and prints their object names. The advantage of this is that I do not need to throw away this piece of useful code or rewrite the same in Jython. All I have to do is call this program from WLST. Listing 8 shows the program in Java and shows how a user can call this existing Java program from WLST.

"There has always been a need for WebLogic Server to support a scripting solution"

Invoking WLST from Ant

There are many users out there using Ant to automate their WebLogic domain configuration and if you would like to embed your WLST configuration scripts into your Ant build files, you can certainly do so. Listing 9 shows a snippet of a build file that invokes WLST to run a script.

Extend WLST Commands

WLST defines a limited set of commonly used or popular commands out of the box, but this does not restrict a user to define custom commands or modules that can be imported into WLST to suit their needs. Listing 10 defines two custom commands – "createServer" and "deleteServer" – that will create and delete a server with the specified name. These commands will be active for the WLST session. Users can also write modules that define several useful functions and can be imported into WLST. Listing 11 shows an example in which a user defines a security class that has utility functions to create a user, change a password, etc.

WLST Module

Often times, users want to use WLST in conjunction with their other Jython modules. This can be done as easily as writing the WLST module to a file and importing it in your Jython modules. Listing 12 demonstrates how to write out the WLST module and use in a Jython module. You can also check out http://dev2dev.bea.com/codelibrary/code/posample.jsp.

WLST in Action

Let's work through a simple use case that demonstrates an end-to-end scenario where WLST fits in naturally. Consider an administrator trying to configure a domain for his developers. Essentially, he would like to create a domain (one admin server) from scratch, start the server, configure managed servers, and cluster and deploy applications. He would also like to monitor the health of these servers via a simple script. The steps involved here are:

- Create a simple domain (one admin server) from a template and start the server (see Listing 13).
- Configure two managed servers and a cluster. Add these two managed servers to this cluster and start the cluster after connecting to the running admin server. Also, deploy a simple application to the cluster (see Listing 14).
- With a simple monitoring script Monitor the server's health (see Listing 15).

Looking Forward

WLST is available today on BEA Systems' dev2dev site. This version of the tool works with WebLogic Server versions 7.0 and 8.1, including all service packs. Support for this tool is provided via the BEA Systems' Management Newsgroups. It will be officially available and supported in WebLogic Server version 9.0, slated for release sometime next year. BEA Systems is also considering retroactive support for WLST for WebLogic Server versions 7.0 and 8.1.

Some of the new features in WLST that you can expect to see are:

- Deployment via JSR 88 APIs including the editing of deployment plans via WLST
- Access to all diagnostic framework (new in WebLogic Server 9.0) artifacts via WLST
- Merging offline WLST (today available as a separate download) with the existing online WLST tool
- Access to the new (WebLogic Server 9.0) management APIs
- Support for all new node manager features that are coming in WebLogic Server 9.0

Summary

In this article, I introduced a new tool, the WebLogic Scripting Tool (WLST), that will empower developers and system administrators to conveniently interact with WebLogic Server (both offline and online) and be able to perform actions that achieve configuration changes to WebLogic Server. WLST can be used in an interactive mode, script mode, or in an embedded mode. It provides a generic and simple access to MBeans in WebLogic Server and also defines convenient commands that can be used to monitor the life cycle of a WebLogic Server instance. WLST is built on Jython, which is a 100% Pure Java implementation of the popular Python language.

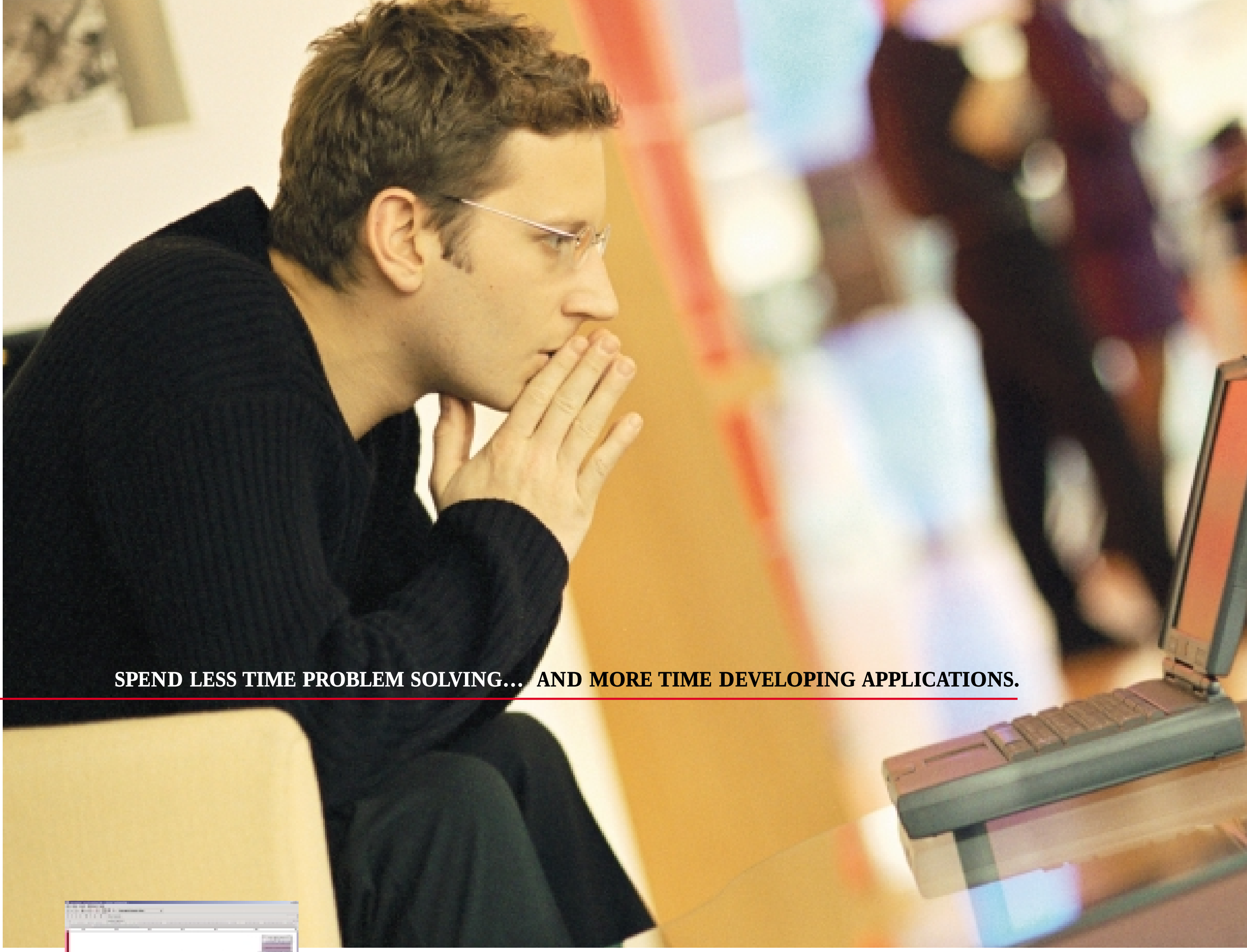
References

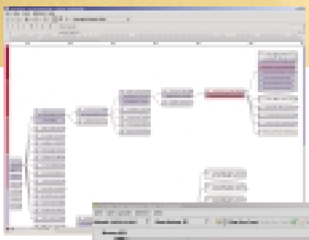
- *Online WLST Tool:* http://dev2dev.bea.com/codelibrary/code/wlst.jsp
- *Offline WLST Tool:* http://dev2dev.bea.com/codelibrary/code/wlst_offline.jsp
- *Jython's Main Web Site:* www.jython.org
- *PO Sample:* http://dev2dev.bea.com/codelibrary/code/posample.jsp

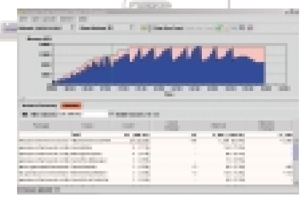
Listing 1: Interactive WLST session

```
wls:/mydomain/config> ls()
drw-   ApplicationManager
drw-   Applications
drw-   BasicRealms
drw-   BridgeDestination
drw-   BridgeDestinationCommon
...
```

```
wls:/mydomain/config> cd("Servers/myserver")
wls:/mydomain/config/Servers/myserver>cd("../..")
wls:/mydomain/config> ls('a')
-rw-   AdministrationPort                         9002
-rw-   AdministrationPortEnabled                 false
-rw-   ArchiveConfigurationCount                     5
-r--   CachingDisabled                            true
-rw-   ClusterConstraintsEnabled                 false
```


SPEND LESS TIME PROBLEM SOLVING… AND MORE TIME DEVELOPING APPLICATIONS.


PerformaSure – a system-wide performance diagnostic tool for multi-tiered J2EE applications running in test or production environments.


JProbe – a performance tuning toolkit for Java developers.

Join The Thousands of Companies Improving Java Application Performance with Quest Software.

Whether it's a memory leak or other performance issues, Quest Software's award-winning Java products – including JProbe® and PerformaSure™ – help you spend less time trouble-shooting and more time on the things that matter. Quest's Java tools will identify and diagnose a problem all the way down to the line of code, so you no longer have to waste time pointing fingers or guessing where the problem lies. Maximize your team's productivity with Quest Software by downloading a free eval today from **http://www.quest.com/weblogic.**

QUEST SOFTWARE®

```
-rw-   ConfigurationVersion                        8.1.2.0
```

Listing 2: Simple WLST script

```
"""
This script connects a server and creates two
managed servers, server1 and server2
"""
connect("weblogic","weblogic")
ls()
servers = "server1","server2"
for s in servers:
  print 'creating server '+s
  create(s,'Server')
print 'Done creating servers'
exit()
```

Listing 3: A Simple Embedded WLST Example

```
import weblogic.management.scripting.utils.WLSTInterpreter;
...

    interpreter = new WLSTInterpreter();
    interpreter.exec("connect('weblogic','weblogic')");
    interpreter.exec("create('server1','Server'));
...
```

Listing 4: Navigating to Different WLST Drives

```
wls:/mydomain/config> runtime()
wls:/mydomain/runtime> ls()
drw-   DeployerRuntime
drw-   ServerRuntimes

wls:/mydomain/runtime>ls('a')
-r--   ActivationTime                              Wed Jul 07
17:42:15 GMT-05:00 2004
-r--   CachingDisabled                             true
...
wls:/mydomain/runtime> config()
wls:/mydomain/config> custom()
wls:/mydomain/custom> ls()
drw- mycustomDomain:Name=MyCounter,Type=TheCounter
drw-   anotherDomain:Name=MyGauge,Type=TheGauge
wls:/mydomain/custom> config()
wls:/mydomain/config>
```

Listing 5: Use of the Command configToScript to Convert a config.xml to wlst script

```
wls:/(offline)> configToScript("config.xml","myDomain.py")
Converting resource ... Server/myserver
Converting resource ... EmbeddedLDAP/mydomain
Converting resource ... SecurityConfiguration/mydomain
Converting resource ... SSL/myserver
...
```

Listing 6: Create a medrec Domain from a Template

```
readTemplate("c:/bea/weblogic81/common/templates/domains/medrec.jar")

create user weblogic
cd('Security/medrec')
cd('User/weblogic')
cmo.setPassword('weblogic')

write the domain and close the template
setOption('OverwriteDomain', 'true')
writeDomain('c:/bea/user_projects/domains/medrec')
closeTemplate()
```

Listing 7: The cmo Variable

```
wls:/mydomain/config> print cmo
[Caching Stub]Proxy for mydomain:Name=mydomain,Type=Domain
wls:/mydomain/config> cmo.getServers()
array([[Caching Stub]Proxy for mydomain:Name=myserver,Type=Server],
weblogic.management.configuration.ServerMBean)
wls:/mydomain/config> cd("Servers/myserver')
wls:/mydomain/config/Servers/myserver> print cmo
[Caching Stub]Proxy for mydomain:Name=myserver,Type=Server
wls:/mydomain/config/Servers/myserver> runtime()
wls:/mydomain/runtime> print cmo
[Caching Stub]Proxy for
mydomain:Location=myserver,Name=mydomain,Type=DomainRuntime
wls:/mydomain/runtime>
```

Listing 8: Util Class that Queries the MBeanServer

```
public class QueryUtil
{
  Context ctx = null;
  String username = "weblogic";
..
..
..

void queryAllManagementObjects() {
    Set s = mbs.queryNames("*:*",null);
    Iterator iter = s.iterator();
    while (iter.hasNext()) {
      System.out.println(iter.next());
    }

  }

..
..

Calling the java util class in jython

myQuery = QueryUtil()
myQuery.queryAllManagementObjects()
```

Listing 9: How to Invoke WLST is an Ant Script

```
<target name="createServers">
    <java classname="weblogic.WLST" fork="no">
     <arg line="createServers.py" />
    </java>
 </target>
```

Listing 10: Defining Custom Commands

```
def createServer(serverName):
  """
  Creates a Server with the specified Name
  """
  print 'creating the server with name '+serverName
  create(serverName,'Server')
  print 'Done creating the server'

def deleteServer(serverName):
  """
```

```
  Deletes a Server with the specified Name
  """
  print 'Deleting the server with name '+serverName
  delete(serverName,'Server')
  print 'Done deleting the server'
```

Listing 11: Security Module that Defines a Few Utility Commands

```
class security:

  def __init__(self):
    print 'use help() for information'

  def help(self):
    print '  createUser - Takes the username, password and the descrip-
tion as arguments'
    print '  listUsers - Lists all the users in your default realm'
    print '  changePassword - changes the password for the given user-
name, takes username, oldpassword and new password as arguments'
    print '  unlockAccount - Unlocks a user account - takes the user-
name as argument'
    print '  listGroups - Lists all the groups in your default realm'
    print '  help - Prints this help'


  def createUser(self,username,password,desc="xx"):
    cd("/")
    cd("SecurityConfiguration/"+domainName)

cd("weblogic.security.providers.authentication.DefaultAuthenticator/Secur
ity:Name=myrealmDefaultAuthenticator")
    cmo.createUser(username,password,desc)
    print 'Created user '+username+' successfully'
    cd("/")

  def changePassword(self,username,oldPassword, newPassword):
    cd("/")
    cd("SecurityConfiguration/"+domainName)

cd("weblogic.security.providers.authentication.DefaultAuthenticator/Secur
ity:Name=myrealmDefaultAuthenticator")
    cmo.changeUserPassword(username,oldPassword,newPassword)
    cd("/")
```

Listing 12: Write the WLST Module to a File

```
wls:/mydomain/runtime> writeIniFile("wlstModule.py")
The Ini file is successfully written to wlstModule.py
wls:/mydomain/runtime>

# import the module into a new jython script as shown and use it
import wlstModule as wlst

wlst.connect("weblogic","weblogic")
wlst.create("mycluster", "Cluster")
```

Listing 13: Create a Default Domain from a Template

```
readTemplate("c:/bea/weblogic81/common/templates/domains/wls.jar")

# create user weblogic
cd('Security/mydomain')
cd('User/weblogic')
cmo.setPassword('weblogic')
# write the domain and close the template
setOption('OverwriteDomain', 'true')
writeDomain('c:/\\ satyaarticle\\mydomain')
closeTemplate()
```

Listing 14: Start an Admin Server

```
startServer(domainDir="c:\satya\article\mydomain", block="true")
connect("weblogic","weblogic")
svr1=create("server1","Server")
svr2=create("server2","Server")
cluster=create("cluster1","Cluster")
svr1.setCluster(cluster)
svr2.setCluster(cluster)
svr1.setListenPort(8001)
svr2.setListenPort(9001)
# Now start the cluster, assuming that a Node Manager is running
start('cluster1','Cluster')
deploy("wlst-demo","c:\satya\article\demo.war","cluster1")
```

Listing 15: Indefinitely Check the Health of a Server

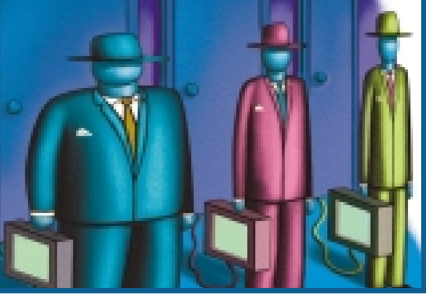
```
import thread
import time

def checkHealth(serverName):
  if connected=='false':
    connect("weblogic","weblogic")
  while 1:
    slBean = getSLCRT(serverName)
    status = slBean.getState()
    print 'Status of Managed Server is '+status
    if status != "RUNNING":
      print 'Starting server '+serverName
      start(serverName)
    time.sleep(5)


def getSLCRT(svrName):
       config()
       cd("/")
       cd("Servers/"+svrName)
       slcBean = cmo.lookupServerLifeCycleRuntime()
       return slcBean

checkHealth("server1")
```

2004 RCAs
Readers' Choice Awards
Nominations Open
"The Oscars of the Software Industry"
Coldfusion Developer's Journal
CFDJ
READERS' CHOICE AWARD
Nominate the Best of the Best:
Tell us what tools, solutions, services, and books deserve recognition.
Cast your vote at
www.SYS-CON.com



Transactions for the Next Generation

BAM BY THE BACK DOOR?

A realization has dawned across the industry that "service-oriented architecture" is a good thing. In fact, this is less of a dawning and more of a reawakening.

Ever since our Neolithic forebears picked up a bone and realized that separating the 3270 screen handling code from that which implemented their business rules was a "jolly good thing," three tier architecture, n-tier architecture, SOA, call it what you will, has been recognized as the best way to build systems. The facets that make it the best way are called out by the industry mantra "coarse-grained, loosely coupled, asynchronous interfaces," as we all know.

Looking around at a lot of enterprise systems in production today, the "asynchronous" part is often (maybe even usually) delivered via use of an asynchronous reliable queuing mechanism. The power of the "fire and forget" programming style that these technologies bring to the table is that your part of the system can simply post a message and rely on the fact that it will eventually arrive at its intended destination (or be processed by some harassed administrator who was paged in the middle of the night when the infrastructure went pop). The ability to rely on this behavior keeps the application code nice and simple – the concept of "fire and forget" is readily understood, and the technical complexity required to implement it is encapsulated within the queuing product. Couple that with transactional XA access to a database, and your application can rest easy in its bed at night, knowing that everything it wants to achieve will happen once and once only, with no user code needed.

Life is good. We know the chant, we sing it, and we have a trouble-free existence as developers. Everything is as it should be.

Of course, in all of life's rapture moments, something is bound to happen to cause a rude awakening. Imagine now the discordant ringing of a thousand alarm clocks…. Snap out of it! Life isn't that good!

For a start, the administrator got paged in the middle of the night when things went wrong… What did he have to do to fix things? Well, we can't answer that. It depends. Where was the message going? Why didn't it get there? Is the data in the message in error, did the queuing infrastructure break under the strain? Our administrator needs to answer these questions (hopefully in consultation with the documentation, rather than the actual developers that wrote the system, but either way, what to do to resolve the situation can only be decided in the light of hidden knowledge of the context around what was going on). For one thing, this highlights that coarse-grained, loosely coupled, and asynchronous are orthogonal attributes – the asynchronous behaviour of our transport has not bought us loose coupling – a point-to-point asynchronous link is still just that – point-to-point.

To address the problems of hidden complexity brought about by a point-to-point design, a choreography layer is introduced into the architecture, decoupling the message senders and receivers. Standards like BPEL, implemented by products such as BEA's WebLogic Integration (or WebLogic Process Edition), provide this place within the architecture for the "knowledge about interconnectedness" to live, so there is a point of reference for the guy who needs to fix things when they break – a single place to go to see the end-to-end message flow, and determine where things fouled up in the overall scheme of things to provide some context for fixing whatever failed. This is the realm that is emerging as the "enterprise service bus (ESB)."

Another rude awakening comes when the user of the application built on this asynchronous technology says, "What happened to my trade" (or, in general, what happened to my <whatever I initiated a while ago with my app>). Suddenly, the support staff is scrambling around with all kinds of headaches. Where did this particular instruction

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

get to? If you are using a choreography layer to solve the point-to-point problem outlined above, then this should provide a means of drilling into an individual instance of a business flow and tracking where it got to (and perhaps why it's stuck there) so you have a solution for this class of problem too.

Then, just as you were feeling smug and ahead of the game, the managers of the individuals who use the system come along and say, "How's it working? Is everything fine? How is our business performing?" Now, the ability to drill into an individual request is not so important. What's needed is to get a top-level overview of the system – how many business transactions are in flight? Where are they all up to in the chain of events from entry to completion? Have any taken too long to pass through the system, requiring intervention to fix whatever is wrong?

This last requirement is usually termed "business activity monitoring" (BAM) – and in traditional point-to-point messaging-based systems, its delivery generally constitutes some kind of dashboard that shows red lights when "too many" messages haven't passed through any given point (or when messages appear to have missed stages, or otherwise broken the rules) and the infrastructure that gathers the transaction tracking data needed to provide the aggregated view generally consists of yet more queues. As each message is delivered to the next application, a "BAM message" is sent to the BAM system, which processes it (along with all the others that are flooding in) to determine where in the end-to-end system each individual business transaction is. Again, in a world orchestrated by an ESB-type approach, the message bus provides an ideal point to collect and collate all this information.

So that's nice. We are happy again, and can start to go back to our sweet dreams.

So, What About "Next Generation Transactions"?

Well, the scenario I described is very much one in which the overall structure of the system is fairly static. OK, so you configured your ESB to wire the constituent services together so it wasn't point to point, but the flow follows the same consistent pattern execution after execution. What about other scenarios? Imagine an order fulfillment system that works out a price for an order by getting prices and availability from several sub-contractors and then makes some decision as to which set of purchases provide the best offer. For example, I get a quote for laptops from three suppliers, each responds with a different price, guarantee terms, and availability. I then get a quote to ship the laptops from each of these suppliers to my customer. My overall price for each permutation is the sum of the unit price and shipping cost, and my customer may wish to choose between the options based on the overall price, and required delivery dates, the guarantee terms, etc. The flow of the final transaction will only be decided when these decisions are made. Suddenly, we are in a situation where we are getting forced down the line of writing an "orchestration" application, which is relatively closely coupled to all the back ends it accesses.

Moreover, because we are getting painted into that corner, tracking our orders will be rather difficult: for every order, we will have to track the decisions that were made in order to know where to look for our in-flight order, once placed. This is where the new-generation transaction comes in. The notion of a cohesion – a unit of work similar to a transaction in that it can be cancelled or committed, but distinct in that it does not have to complete or cancel in an all-o-nothing fashion – was introduced in OASIS' BTP specification. The concept is currently working its way into the WS-BusinessActivity variant of the emerging WS-Transaction specification. Provisional orders for the laptops could be placed with the three suppliers in one cohesion, with every intention that by the time the order is finalized, two of these three laptop orders will be cancelled again. It will be similar for the shipment requests. Using a framework to drive this loosely coupled transactional behavior will greatly simplify the code of the orchestrating application (and provide loose coupling benefits similar to those the ESB offered in the fixed-flow case). The fact that a framework is being used to track all these interactions means that when the requirements for the BAM-style capability surface, the next-generation transaction framework provides a central point of reference, which will directly provide the "where has my transaction got to" information, and (via something analogous to transaction timeouts) be able to monitor service-level agreements and provide the information to drill in to to diagnose problems. In fact, this is an often-overlooked benefit of the traditional JTA style transaction manager: it provides a place you can go to and say, "What data did this operation touch?" at least while the transaction is in flight.

So, a framework supporting cohesions could become to the dynamic application system what the ESB is to the static one. In fact, it is also possible to envisage a cohesions framework augmenting an ESB-style approach, even for statically defined systems. The ESB provides a good "message oriented" way to visualisze a composite application, but business semantics are not described in messages, technology solutions are. Cohesions provide a standardised mechanism for associating messages (and processing of the messages) with business transactions - things like "place an order" that it doesn't take an IT expert to comprehend. Service-oriented architecture encourages us to build components in a loosely coupled fashion so that they can be combined in ways not thought of when they were initially created. Their loose coupling however is merely a technically loose coupling. The composite services are by definition coupled – why else does the business need invocations of the components to occur in sequence? The concept of cohesions provides a standardized way to identify the business semantics causing the message flow – it is this kind of information that will be needed for business activity monitoring at the business level (as opposed to message – flow spotting at the infrastructure level). With an ESB approach, this higher-level information is likely to be some kind of static commentary on a flow diagram, whereas a cohesions framework allows the information about the business semantics of each operation to flow dynamically with the messages that implement it, providing for better visibility into and management of the composite application system, not to mention the more traditional transaction-related business benefits of consistent recoverable outcomes thrown in for good measure. Maybe we had better watch out for the "enterprise transaction bus" appearing alongside the enterprise service bus. But that's the thing about buses. Never one when you want one, and then several arrive at once!

• • •

Transactions: Driving You to Distraction? Addenda!

In the May/June issue (Vol. 3, issue 5) of *WLDJ*, I wrote (after hearing the topic come up on numerous occasions, in many contexts) about the best choice of JDBC driver for use under WebLogic Server–based applications.

–continued on page 39

A Service-Oriented Management Approach for Service-Oriented Architecture

CREATING THE PROCESS APPLICATION

BY **FRANCO NEGRI**

AUTHOR BIO...

Franco Negri is the founder, chief technology officer, and chief strategist of Panacya Inc., a Columbia, Maryland based provider of next-generation systems management solutions.

CONTACT...

franco.negri@panacya.com

Much has been written about service-oriented architecture (SOA) and the many technology and business benefits of adopting this approach. Poised to change the computing landscape once again, progressive IT departments, software vendors, and service providers have all been eager to embrace its concepts – familiar to anyone acquainted with the many past attempts to represent applications and IT infrastructure as modular reusable services.

Contrary to the views of some, SOA is not about .NET or J2EE or any specific platform or standards, although the continued adoption and early successes of Web services implementations will likely galvanize the industry around its standards. Rather, SOA is an application architecture approach to building distributed systems that deliver application functionality as services to end-user applications or to build other services.

Because SOA not only represents a philosophical shift for developers but also has great implications for IT operations, understanding the operational aspects of managing and monitoring, SOA with what I will call a "service centric, end-to-end approach" is critical. It offers IT professionals a great opportunity to finally get it right throughout the application life cycle, from the development process, through the operational management aspects.

Background

Service-oriented architecture is an approach to loosely coupled, standards-based, and protocol-independent distributed computing, where coarse-grained software resources and functions are made accessible via the network. In an SOA, these software resources are considered "services," which are well defined, self-contained, and ideally do not depend on the state or context of other services. Services have a published interface and communicate with each other. Services that utilize Web services standards (WSDL, SOAP, UDDI) are the most popular type of services available today.

Many believe that SOA, by leveraging new and existing applications and abstracting them as modular, coarse-grained services that map to discrete business functions, represents the future enterprise technology solution that can deliver the flexibility and agility that business users want. These coarse-grained services can be organized/orchestrated and reused to facilitate the ongoing and changing needs of business.

Advantages of an SOA

Implementing SOA provides both technical and business advantages. From a technical point of view, the task of building business processes is faster and cheaper with SOA, because existing services can more easily be reused and combined to define the business processes themselves. Applications can expose their services in a standard way and, hence, to more diverse clients and consumers. From a business perspective, IT staff can communicate more easily with business people, who understand services. Because business processes become explicit, they can be understood and improved with greater ease. Additionally, applications or business processes can be managed internally more easily or outsourced, because they're well-defined and discrete. As business changes and new requirements are generated, IT can reuse services to meet new demands in a much more efficient and timely manner.

The value and ultimate success of SOA is based on the assumption that everything enterprise IT does is ultimately manifested in the service of some business process. Given this assumption, SOA is about making business processes better, cheaper to create, and easier to change and manage.

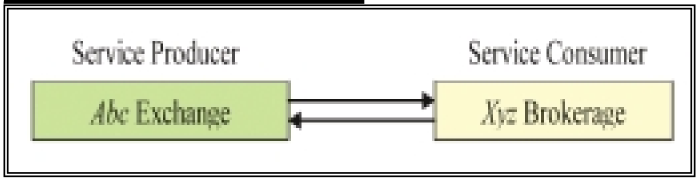
New Operational Challenges for Managing SOA

Ironically, IT operations tasked with managing and monitoring SOA face the same major challenges as developers: the fundamental philosophical shift that SOA represents. Operations staff currently manages IT assets from a technology perspective. With SOAs in place, the focus needs to shift to a service centricity. wihout understanding their interractions and interdependencies, or how they impact the services provided SOA, managing technology from the perspective of services was difficult for IT operations, which have difficulty understanding and defining services in general. In the absence of clear definitions of business services, IT operations have traditionally focused on managing and monitoring all of the individual tiers of technology separately without understanding their interactions and interdependencies or how they impact the services provided.

However, when you consider the adoption of SOA many obvious operational questions arise:

- Who is going to own the management of business services?
- How will the health, performance, and capacity of these services be monitored?
- When a problem arises, how will operations personnel be able to relate coarse-grained business service degradation to infrastructure bottlenecks?
- What enabling technologies or techniques need to be made available to enable personnel across multiple departments (development, QA/Test, operations support, etc.) to work together in real time to prevent service failures or performance degradations?

FIGURE 1



A simple Web services application

- Do the current technology-segregated IT processes work in an SOA-enabled environment?

The answers lie in a best practice approach that manages as one cohesive and integrated solution, manages the interaction between the services and underlying infrastructure. This management id done from an end-to-end perspective, using measurements of capacity, availability, and performance (CAP) to integrate and simplify management functions.

Best Practices for Managing SOA

Using such an end-to-end approach offers IT operations far more flexibility and adaptability in an SOA environment than traditional, more piecemeal management of underlying systems or of services and their interfaces.

Measurements of CAP at the services layer should act as a trigger for all other management functions and actions so that the proper focus on services and service quality can be maintained throughout an organization. The advent of clearly articulated business services via SOA can and should drive all operational management functions from the perspective of service quality, expressed as measurements of service capacity, service availability, and service performance. This could eliminate once and for all the finger-pointing and ambiguities we all encounter in operations when finding and fixing problems during runtime. For the purpose of this best practice approach to service-oriented management, Web services standards are implied.

Figure 1 depicts a simple service. It is presumed that Web services standards are used to abstract and integrate the functions of two existing applications on different platforms, written in different languages and in different locations. Both the service producer and consumer's services become interoperable via Web services standards, including SOAP for messaging, XML for message and data format, WSDL for description of services, and UDDI for service discovery. With the application's services clearly articulated and defined, the opportunity exists to coherently "instrument" it and make its measurements available for the runtime management of the service.

Applying CAP metrics and measurements to the Web Service enables operations to clearly understand the behavior of the service and its interactions. For example:

- ***Capacity/load metrics:*** Is the number of connections, sessions, and requests/responses within the intended design limits? Is the number of connections/requests within the defined service-levels for capacity?
- ***Availability metrics:*** Is the service accessible and functioning? Is it returning the expected results? Is it operating within the defined service-levels for availability?
- ***Performance metrics:*** Is the response time within an acceptable range? Is response being impacted by load? Is the response time within the defined service levels for performance?

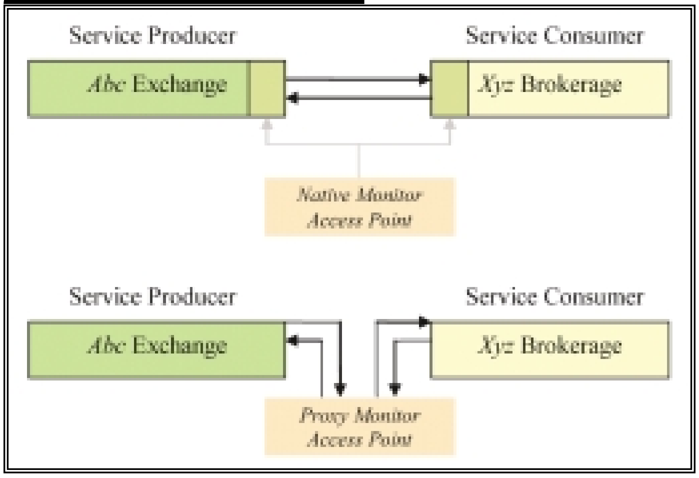
Instrumentation Techniques: Getting the Service Measurements

Two fundamental principles can be applied to accurately and proactively measuring and monitoring services – active and passive monitoring. Active monitoring implies creating "synthetic transactions" that actively test a service by periodically executing over specified intervals. Passive monitoring looks at the transactions and interactions as they occur. Active monitoring is inherently proactive in that it, doesn't wait for an error or degradation to occur before detecting it even though it does not reflect the actual interactions between services. Experience shows that using both of these techniques together produces the best result.

Web services enable powerful instrumentation without the need to modify applications. Definitions exist for inserting instrumentation to reveal characterization of the transaction, its start time, its stop time, its transaction type, and the service with which it is communicating.

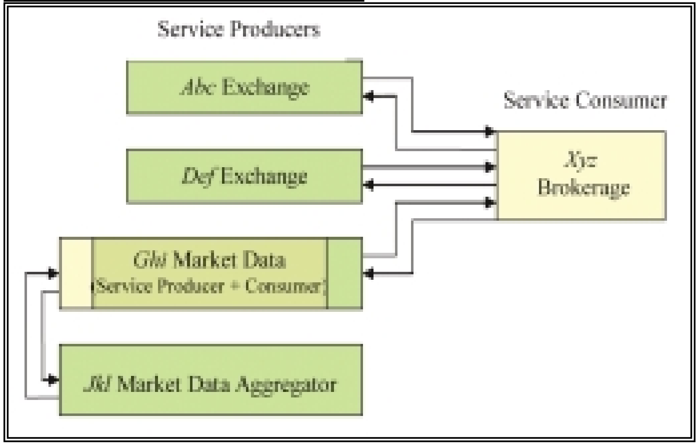
Instrumentation techniques fall into two categories: proxy and native instrumentation. The proxy method involves modifying the IP address to intercept messages between service providers and service consumers. The native approach requires you to use available exits in SOAP processors contained in both the service providers and consumers. Of course, both techniques involve tradeoffs. Basically, the proxy method enables you to be SOAP-processor neutral but you take a performance hit by being in-line with all messages. The native method doesn't entail these consequences, but it does require specificity to a particular SOAP processor.

FIGURE 2



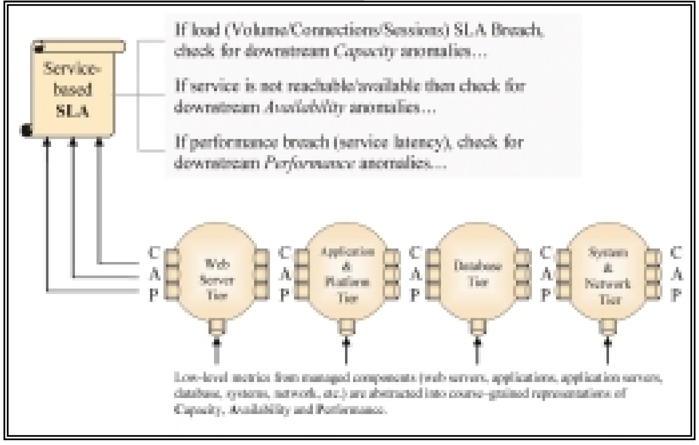
Native or proxy instrumentation techniques

FIGURE 3


Service Producers
Abc Exchange
Service Consumer
Def Exchange
Xyz Brokerage
Ghi Market Data (Service Producer + Consumer)
Jkl Market Data Aggregator

Web services – many complex interactions

FIGURE 4


Service-based SLA
If load (Volume/Connections/Sessions) SLA Breach, check for downstream Capacity anomalies…
If service is not reachable/available then check for downstream Availability anomalies…
If performance breach (service latency), check for downstream Performance anomalies…
CAP
Web Server Tier
Application & Platform Tier
Database Tier
System & Network Tier
Low-level metrics from managed components (web servers, applications, application servers, database, systems, network, etc.) are abstracted into coarse-grained representations of Capacity, Availability and Performance.

CAP measurements are used to standardize and correlate information between infrastructure tiers and services

Managing Interactions Between Services

Because Web services applications are likely to have many producers and consumers active at any time, the interactions between them must be managed. With this increased complexity (see Figure 2) comes increased concerns about service availability and performance. Being able to perform both active and passive monitoring of Web services and the interactions between them becomes paramount. From an operational perspective, managing interactions with external services (across enterprises) also represents an added complexity. Service-level-agreements (SLAs) need to be in place in order to clearly define and monitor the expected performance characteristics of the service.

Applying SOA Concepts to Infrastructure

As we have seen, a best practice approach for managing SOA-enabled business services will require the management of the interaction between the services and the underlying system from the perspective of capacity, availability, and performance and as one integrated solution. Using the right technology and approach, it is possible to manage applications stacks end-to-end and provide "coarse-grained" representations of CAP at each infrastructure tier instead of monitoring capacity, availability, and performance of IT assets in a piecemeal fashion. Unlike current methods, this approach would enable IT operations to quickly locate emerging problems.

However, from a service-support perspective, operations must be able to make sense of the torrent of information and events that they receive from a myriad of monitoring and analysis tools that neither abstract low-level measurements into cohesive and easy to understand information nor provide a contextual reference for interpretation. A solution to this problem is to limit the number of monitoring and analysis tools and insist that they minimally automate the analysis process out-of-the-box. Such tools should not require operations staff to set thousands of static thresholds to manually define how an alarm/event is generated. Modern management solutions should come configured to automatically detect any abnormal conditions in the environment. They should also provide a mechanism to aggregate and convert low-level events/alarms into coarse-grained and humanly understandable measurements of CAP.

In order to manage IT assets in the context of business services, all underlying infrastructure measurement and monitoring technology related to the service needs to be standardized into a unified taxonomy. Figure 3 depicts how individual infrastructure elements can be monitored and analyzed across tiers in real time. Individual metrics (informational events, alarms, etc.) from each managed element must be abstracted into overall measures of

Want to spend more time developing software and less time supporting it? Spend some time discovering hp OpenView—a suite of software management tools that enable you to build manageability right into the applications and Web services you're designing.

Find out how the leading-edge functionality of hp OpenView can increase your productivity.

http://devresource.hp.com/d2d.htm

hp®
invent

capacity, availability, and performance in order for them to be humanly consumable and then to enable automated and "standardized" monitoring across tiers of infrastructure.

Figure 3 is not meant to be "anatomically correct," but rather to illustrate that a cohesive, end-to-end CAP monitoring strategy is not only possible but also necessary as infrastructure stacks become more complex and dynamic. Like the SOA concept of abstracting fine-grain application functions into coarse-grained services, end-to-end application and infrastructure stack component measurements (Web server, application server, database, etc.) could be abstracted into higher-level measurements of CAP. New service-oriented management systems will leverage these standardized measurements and provide a means to aggregate and correlate them to the services that they provision. Imagine being able to categorize and find a capacity or performance bottleneck down to at least the level of an element or component. How are application server performance measurements impacted by network performance measurements did what impact do they have on the service layer? In my experience, most IT shops do not have this down to a science, but it is possible – and in an SOA-enabled enterprise, where services are providing business differentiation, service quality will be increasingly important.

Bridging the Gap Between Web Services and Application and Infrastructure Management

Having standardized measurements of both Web services and the supporting application stack makes it possible to trigger downstream infrastructure stack analysis and alerting to the measurements of Web service quality, as described earlier. If the Web service layer measurements and monitors detect a performance problem, they can automatically trigger downstream analysis in order to determine like-kind (performance) infrastructure problems that mayhave been occurring at the time the service degraded. Figure 4, a high-level diagram of an actual project to provide end-to-end proactive monitoring for a BEA WebLogic Integration 8.1 SOA platform, depicts a Web services-centric operational management diagram along with a simple SLA-based analysis workflow for correlated problem detection. BEA made it easier than usual by publishing Web services statistics via JMX and also publishing performance statistics of Web services that were organized into business processes via their Workshop product.

"The promise of enhanced flexibility, adaptability, and agility in the context of "everything services" will win in the end"

Summary

The SOA trend – already pronounced across the industry – will, in my view, only accelerate over the next several years. The promise of enhanced flexibility, adaptability, and agility in the context of "everything services" will win in the end. However, the complete value of SOA will be fully realized only when all parties involved in IT service delivery, and service support of the entire application life cycle, work together with the common goal of designing, coding, testing, deploying, and managing services from the common objectives of business.

This is an exciting time for IT developers and operations. In an SOA world, they are both seated at the head table as trusted advisors to the business and as critical partners for any key revenue-generating or cost-reduction objectives. By taking a unified, service-oriented approach to designing, deploying, and managing business services, they have a wonderful opportunity to get it right.

Service on Demand Portals

–continued from page 24

- ***User experience:*** All participating applications (producers) should conform to a common look-and-feel standard. Navigation should also be consistent across them.
- ***Single Sign-On:*** Since the departmental portals are no longer Web applications, but are services similar to Web services. The WSRP spec does not address security directly but encourages leveraging standards such as WS-Security, SAML, XML Signatures, and XML-Encryption. Consumers should be able to provide SAML assertions to authenticate and propagate user identity to the Producer. BEA WebLogic's WSRP Identity Assertion Provider, used on the producer end, validates user identity through the SAML assertions.
- ***Entitlements:*** High-level entitlement-based access to the main portal can be accomplished by setting role-based entitlements on WSRP consumer (proxy) portlets at the consumer end. This indirectly allows/blocks access to the WSRP producer, provided they are not directly available. It is recommended that the producers also enforce certain authorization checks and not rely completely on the consumer to enforce that. Furthermore, low-level entitlements can be independently enforced by the producer. In order to do this, the producer needs to get the user identity and information via SAML assertions.
- ***User Management:*** All portal products have their own user repository to store user customizations and personalization profile elements. Ideally, a central user directory should be leveraged as much as possible for user identity and basic profile information.
- ***User profile information:*** Though defined in the WSRP specs, BEA's current implementation (WebLogic Portal 8.1 SP3) does not support the passing of user information in the initial request. One option is for the Producer to directly access the user repository and fetch the user profile.
- ***Sharing content management infrastructure:*** Content management (CMS) can be a centrally deployed, shared service that is leveraged by all the portals. Though the CMS is on a centralized infrastructure, individual portals can have their own sandboxes with independent release schedules.

Additionally, with the federated approach, each departmental portal's services and information can be freely shared and used to support other portals. It allows loose coupling between portals into a federation of shared services where any portal can leverage information from any other portal.

• • •

Next month we'll look more deeply at federated portals, WSRP, and EIB.

–continued from page 33

As every consultant tries to do, I hedged my bets and gave some good generic advice (namely, benchmark your application using the XA configuration you need for production to ensure that your exact setup will give the throughput you require from your application), but I also (after much hesitation and deliberation) came off the fence and gave specific advice: "with the technology where it is today, it seems safe to conclude that you should try to stick to the type 2 driver for XA database access, unless you have a pretty compelling reason to do otherwise."

I stand by that advice (and boy, am I glad that I included the "technology where it is today" caveat!!!), but needless to say, technology today isn't where technology was at the time of the copy date for the article!!!

A new driver came out of Oracle pretty much as my article left my laptop. That driver is the Oracle 10g thin driver. It supports versions of Oracle prior to 10*g* (as well, of course, as 10*g* itself); it's a type 4 driver so it doesn't need native libraries on the client, it supports XA. Not only does it support XA, but it does so without the overhead on database CPU consumption that the old thin drivers had in XA mode. Hallelujah!

So the new specific advice (what am I doing… I see another correction coming in the future!) is to take a long, hard look at the Oracle 10*g* thin driver as the way to access your database, especially if you're planning to use XA transactions. Not only can you use it in your own database access code, but this driver is supported by all the components of the WebLogic platform in WebLogic 8.1 service packs 2 and 3 (see http://e-docs.bea.com/platform/suppconfigs/configs81/81_over/supported_db.html#1127404).

Now, back to drawing bubbles, clouds, and lines. Why is it my whiteboard architecture diagrams never have these problems?

A Real-World Business Process Model Part 2

CREATING THE PROCESS APPLICATION

BY **ANJALI ANAGOL-SUBBARAO**

AUTHOR BIO...

Anjali Anagol-Subbarao works in HP's IT organization as an IT architect. She has 12 years of IT experience, the last 5 in Web services. Her book on J2EE Web services on BEA WebLogic will be published this year.

CONTACT...

anjali.anagol-subbarao@hp.com

In my first article (WLDJ, Vol. 3, issue 6), I provided an overview of BPM specifications in this area. I described the order change example and the steps needed to create the business process in WebLogic Integration.

In this article, you will see how to create a process application. We can call this application orderChange. In this application we need to create a new process called orderChange.jpd. To start the process, we need to add a ClientRequest received. Next we will add the Web service validate. We will be looking in detail at the steps to create this business process in WebLogic Workshop.

Create New Application and New Process

When you start modeling a business process in WebLogic Integration, you first need to create a business process application called orderChange-process, in which you create the business process OrderChange.jpd. When you create orderChange-.jpd in the Design View, you will see only the Start and Finish nodes

Create Client Request to Start Process

In WebLogic Workshop there are five different ways to start a business process.

- Invoked via a client request
- Invoked synchronously via a client request with return
- Subscribe to a message broker channel and start via an event (timer, e-mail, file, adapter, etc.)
- Subscribe synchronously to a message broker channel and start via an event
- Invoked via one of several client requests or subscriptions (event choice)

In our example, an XML document is sent with changes to request a ChangeOrder. The XML document has the format of RosettaNet PIP 3A8. Let us call it orderchange.xsd. First add the orderchange.xsd to the schemas folder by importing it in the application. When you add the schema, XML Beans are created. We will look at XML Beans in detail in the next section.

We will use Invoked via a Client Request to start the business process as the client is requesting an order change. To do this, you need to create the method and parameters that your client uses to trigger the start of your business process, which is specified in the General and Receive Data settings of this node. General Settings specifies the method exposed by your business process to clients. Clients invoke the orderChange method to start and make requests on your business process. We map this method to a typed XML, orderchange.xsd. That is, the messages received from clients must contain XML that is valid against an XML Schema orderchange.xsd.The General Settings tab is updated to indicate that you successfully completed the specification of a method name and parameters:

In Receive Data, specify a variable to which an Order Change request, received from a client, is assigned at run time. You can use Variable Assignment mode or Transformation mode. Here, you will assign the XML message received from the client directly to a variable orderChangexsd of the same data type as shown in Figure 1.

The JPD process looks like this:

```
@jpd:process process::
 * <process name="orderchange">
 *  <clientRequest name="orderChangeRequest" method="orderChangeRequest"/>
* </process>::
```

The code behind it looks like that in Listing 1.

Add a Web Service

The next step is to call the validateConfig Web service. You can create a Web service control to invoke this Web service. I assume here that you have a WSDL defined for this service. We will create the Web service control through a WSDL. The first step is to import the WSDL into the schemas folder.

When you import the WSDL to the schemas folder of your process application XML, Beans are created. Let's look at XML Beans in detail first.

XML BEANS

XML Schema is the starting point for XMLBeans. The XML Schema specification provides a rich data model that allows you to express structure and constraints on your data. In an XML Schema you can enforce control over how data is ordered in a document or how you can constraint particular values (for example, price has to be more than $150.00). To do this in Java you have to write custom code. XMLBeans honors these schema constraints.

Let's take the schema and XML for outValidateConfig shown in Listing 2 and see how it is converted to XMLBeans. This is the output schema for the validateConfig service and specifies if the configuration is valid or not valid and if it is not valid, the error.

You have one complex type element outValidateConfig. In a schema, a complex type is one that defines an element that may have child elements and attributes. The sequence element nested in the complex type lists its child elements. Since outValidateConfig is at the top of the schema, it is a global type.

Within a complex type, outValidateConfig, you use simple types like ConfigID and complex types like Status. The simple type, which is a built-in type, is part of the schema specification. There are 46 built-in types defined in the specification. When you compile XML schema, the resulting API is made up of two categories of types, built-in types that mirror those in the schema specification and others that are generated from user-derived schema types. To compile the XML schema, you can import the schema or WSDL into the schemas directory in WebLogic Workshop; it will create the XMLBeans.

The compiled XML Schema gives you two generated XMLBeans interfaces: OutValidateConfigDocument and outValidateConfig-Document.outValidateConfig. From the schema point of view, the generated outValidateConfig interface represents the complex type you see inside the schema's outValidateConfig element declaration. This complex type translates into a sequence of four elements: Status, Error, ConfigID, and ShipDate. The outValidateConfig interface exposes methods such as getStatus and setStatus to get and set the value status element.

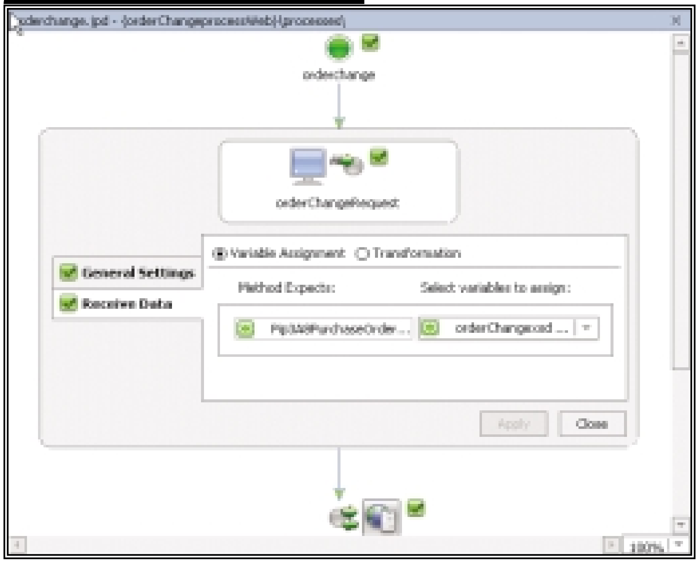
The outValidateConfigDocument interface represents the outValidateConfig document that contains the root outValidate-Config element. XMLBeans create a special "document" type for global element types. A document type provides a way for you to get and set the value of the underlying type, here represented by outValidateConfig. The outValidateConfig element is considered a global element because it is the root element and can be referenced from anywhere else in the schema. To set or get the value of the user defined types get and set methods are provided.

Once we have the XMLBeans, we will create a Web service control from the WSDL. To do this you can add a Web services control with the option of specifying it from a WSDL. Let us look in detail at the control framework.

Control Framework

You can use the control framework of WebLogic Workshop 8.1 to easily connect to and use databases, back-end systems like ERP or legacy systems, custom or vendor applications, and Web services with Java controls. These controls wrap up other controls and add business logic to create reusable, composite components that are part of a business process.

FIGURE 1

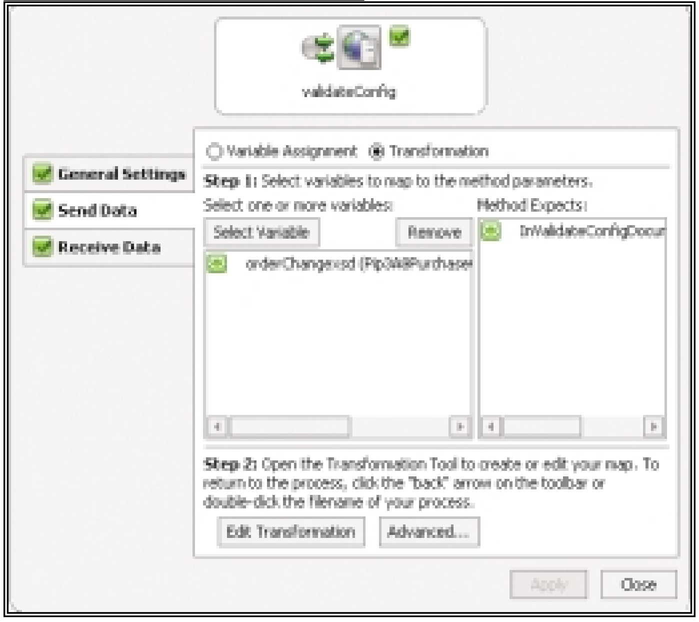

orderchange.jpd - (orderChangeprocessWeb)-(processes)

(◉ Variable Assignment ○ Transformation)

Method Expects: PIP3A4PurchaseOrder...
Select variables to assign: orderChangexsd ...

XML message received from the client

FIGURE 2


(○ Variable Assignment ◉ Transformation)

Step 1: Select variables to map to the method parameters.

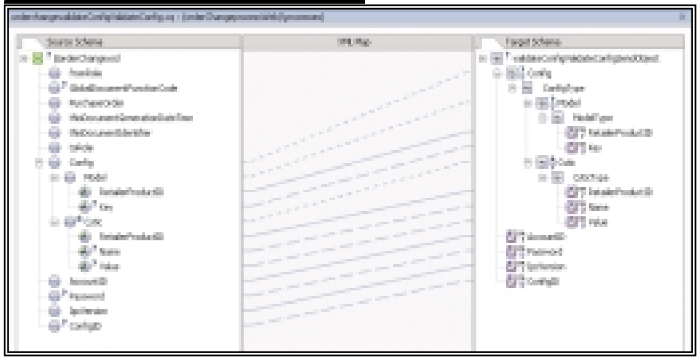
Select one or more variables: orderChangexsd (Pip3A4Purchase

Method Expects: InValidateConfigDocur

Step 2: Open the Transformation Tool to create or edit your map. To return to the process, click the "back" arrow on the toolbar or double-click the filename of your process.

Data transformation for the Web service control

FIGURE 3



Transformation

BEA WebLogic Workshop ships with the following controls:

- ***Web service control:*** The control can import Web services from their WSDL files, which can be local or dynamically picked up from UDDI server. These controls are interoperable with any .NET or Java Web service.
- ***EJB control:*** The EJB control contains the code for JNDI lookups, object creation, and casting steps. EJBs can be invoked through control objects.
- ***Database control:*** Enterprise databases are integrated via JDBC through the database control. SQL Maps allow you to indicate how their Java parameters should be replaced into the query.
- ***JMS control:*** The client listens on this JMS control via callbacks defined in the client. This control enables publishing and subscribing to queues.
- ***J2EE CA Adapter control:*** Enterprise applications and legacy systems can be accessed via J2EE CA adapters, which are controlled by J2EE CA controls. J2EE CA interfaces are integrated with Java through XML maps.
- ***Timer control:*** Scheduling for responses and requests can be managed through this control, which triggers the Web service periodically. This enables it to send status updates to the client, poll a data source, or establish a time-out on a request to a resource.

Once we have created the Web service control, we need to specify the call to the validateConfig Web service control in the business process definition. We need to send data specified by the validateConfig method. To do this, we build a data transformation from the data received in the orderchange.xsd to the validateConfig as shown in Figure 2. The code for the data transformation is shown in Listing 3. In Listing 4, the model element from orderchange.xsd is mapped to the model element in the validateConfig method:

The cstic element from orderchange.xsd is mapped to the cstic element in the validateConfig method in Listing 5.

Figure 3 shows the transformation.

Summary

In my next article I will look at how to create decision points and the order status control in the business process. In the fourth article I'll write the change to a file and end the process. I'll discuss process monitoring in the last article in the series.

Reference

- *BEA WebLogic Workshop Help:* http://e-docs.bea.com/workshop/docs81/doc/en/core/index.html.

Listing 1: Code behind JPD process

```
public class orderchange implements com.bea.jpd.ProcessDefinition
{
public noNamespace.Pip3A8PurchaseOrderChangeRequestDocument
orderChangexsd;
public void
orderChangeRequest(noNamespace.Pip3A8PurchaseOrderChangeRequestDocument
orderChangexsd)
    {
        //#START: CODE GENERATED - PROTECTED SECTION - you can safely
add code above this comment in this method. #//
         // input transform
         // parameter assignment
         this.orderChangexsd = orderChangexsd;
         //#END  : CODE GENERATED - PROTECTED SECTION - you can safely
add code below this comment in this method. #//
     }
```

Listing 2: Schema for outValidateConfig

```
<xsd:element name="outValidateConfig">
 <xsd:complexType>
   <xsd:sequence>
     <xsd:element name="Status" type="TrueFalseType"/>
     <xsd:element name="Error" type="ErrorType" minOccurs="0"/>
     <xsd:element name="ConfigID" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
     <xsd:element name="ShipDate" type="AtLeast1Type" minOccurs="0"/>
   </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

Listing 3: Data transformation

```
{--
processes/orderchangeTransformation.dtf#validateConfignewValidateConfigS
end
--}

declare namespace ns0 = "http://production.company.com/types"

<ns0:inValidateConfig>

Listing 4: Model element
     {
         let $Config := $orderChangexsd/Config
         return
             <ns0:Config>
             {
                 let $Model := $Config/Model
                 return
                     <ns0:Model RetailerProductID = "{
data($Model/@RetailerProductID) }"
                                 Key = "{ data($Model/@Key) }"/>
             }
```

Listing 4: Model element

```
     {
         let $Config := $orderChangexsd/Config
         return
             <ns0:Config>
             {
                 let $Model := $Config/Model
                 return
                     <ns0:Model RetailerProductID = "{
data($Model/@RetailerProductID) }"
                                 Key = "{ data($Model/@Key) }"/>
             }
```

Listing 5: The cstic element

```
             {
                 for $Cstic in $Config/Cstic
                 return
                     <ns0:Cstic RetailerProductID = "{
data($Cstic/@RetailerProductID) }"
                                 Name = "{ data($Cstic/@Name) }"
                                 Value = "{ data($Cstic/@Value)
}"/>
             }
             </ns0:Config>
     }
```

web services EDGE conference&expo

Web Services Edge 2005 East

International Web Services Conference & Expo

Hynes Convention Center, Boston, MA

February 15-17, 2005

The Largest i-Technology Event of the Year!

Guaranteed Minimum Attendance 3,000 Delegates

Tuesday, 2/15:	Wednesday, 2/16:	Thursday, 2/17:
Conference & Expo	Conference & Expo	Conference & Expo

Join us in delivering the latest, freshest, and most proven Web services solutions… at the *Fifth Annual Web Services Edge 2005 East – International Conference & Expo* as we bring together IT professionals, developers, policy makers, industry leaders and academics to share information and exchange ideas on technology trends and best practices in secure Web services and related topics including:

- Transitioning Successfully to SOA
- Federated Web Services
- ebXML
- Orchestration
- Discovery
- The Business Case for SOA
- Interop & Standards
- Web Services Management
- Messaging Buses and SOA
- SOBAs (Service-Oriented Business Apps)
- Enterprise Service Buses
- Delivering ROI with SOA
- Java Web Services
- XML Web Services
- Security
- Professional Open Source
- Systems Integration
- Sarbanes-Oxley
- Grid Computing
- Business Process Management
- Web Services Choreography

3-Day Conference & Education Program

features:

- Daily keynotes from companies building successful and secure Web services
- Daily keynote panels from each technology track
- Over 60 sessions and seminars to choose from
- Daily training programs that will cover Web Service Security, J2EE, and ASP.NET
- FREE full-day tutorials on .NET, J2EE, MX, and WebSphere
- Opening night reception

Interested in Exhibiting, Sponsoring or Partnering?

Becoming a Web Services Edge Exhibitor, Sponsor or Partner offers you a unique opportunity to present your organization's message to a targeted audience of Web services professionals. Make your plans now to reach the most qualified software developers, engineers, system architects, analysts, consultants, group leaders, and C-level management responsible for Web services, initiatives, deployment, development and management at the regions best-known IT business address – The Hynes Convention Center in Boston.

For exhibit and sponsorship information please contact Jim Hanchrow at 201.802.3066, or e-mail at jimh@sys-con.com.

Sponsored by:

JDJ · WebServices Journal · SD Times · wldj · .NET Journal · WebSphere Journal · Information Storage + Security Journal · XML Journal · asp.netPRO · MX developer's journal · eaIndustry Consortium

All brand and product names mentioned above are trade names, service marks or trademarks of their respective companies.

Contact for Conference Information: Jim Hanchrow, 201-802-3066, jimh@sys-con.com

PRODUCED BY SYS-CON EVENTS

www.sys-con.com/edge

Building Business Processes PART 2

WEBLOGIC INTEGRATION DEVELOPMENT BEST PRACTICES

BY **VIJAY MANDAVA & ANBARASU KRISHNASWAMY**

AUTHOR BIOs...

Anbarasu Krishnaswamy is a senior principal consultant with BEA professional services. He has been working with BEA Systems for over 5 years and BEA technology for about 10 years. He is a Sun certified Java programmer and BEA certified WebLogic Server developer.

Vijay Mandava joined BEA as a technical manager in the Professional Services organization in 1999. He now works as a principal systems engineer in the Systems Engineering organization. Vijay is a Sun certified Java programmer and a BEA certified WebLogic Server developer.

CONTACT...

anbarasu@bea.com
vijay.mandava@bea.com

This article is the second of two on best practices in building business processes on BEA WebLogic Integration 8.1. The first installment (WLDJ, Vol. 3, issue 6) focused on team development and maintenance best practices. In this article, we will focus on best practices in building business processes with scalability, recovery, exception handling, guaranteed delivery, and high performance. This article is intended for the developers and architects of WLI applications.

Process Versioning

Best Practice

Version the process files.

Reason

It's easy to overlook versioning because the process works without any version. But it is really important to version the processes, especially for long-running processes. When a process is upgraded, the in-flight instances will either be aborted if the processes were not versioned or the deployment will fail depending upon the mode on which the server is running. The process is aborted if the server is running in iterative mode and the deployment fails if it is running in production mode.

Details

Using WebLogic Workshop's versioning feature, you can make changes to your business process without interrupting any instances of the process that are currently running. When you version a business process, you create a child version of a business process that shares the same public URI (interface) as its parent. At run time, the version of the process that is marked as active is the process that will be accessed by external clients through the public URI.

You can version business processes, but not the individual controls associated with that process or other business process–related components, such as schemas and transformations. When you version a business process, you must also version the sub-processes of that process, they are not versioned automatically when their parent process is versioned. The version of subprocess that is called depends upon the version strategy applied on it. The two strategies available are loosely coupled (version is determined at invocation time) and strongly coupled (version is set at the time the parent process is invoked).

It is important to note that versions cannot be completely different because all of them are different implementations of the same interface. This implies that the versions should have the same request, callback methods, and static message broker subscriptions.

Exception Handlers

Best Practice

Create appropriate exception handlers for the processes. Create global exception handlers for all processes.

Reason

It is natural that at any stage of the business

process a checked or unchecked exception may occur. Exceptions should be properly handled in the business process. If an exception is not explicitly handled, the process may be aborted and never retried. This may also result in unprocessed messages or lost messages.

Details

Exception handlers can be created at three levels.

- Global exception handler
- For a group of nodes
- For an individual node

In general, exceptions propagate upwards from a node exception path, to a group exception path, to a global exception path until they are handled. In other words, the exception path associated with a node executes first, then the path associated with a group, and then the path associated with a start node (global path). The exception is only handled once, unless your exception path throws an exception; then the exception propagates upward again in the same order. You can take advantage of this behavior and create exception path logic that satisfies the particular exception handling necessary for your business process.

For non transactional resources, compensating transactions can be done in the exception handlers.

Recovery

Best Practice

Set appropriate redelivery properties at the process level and/or JMS queue level.

Reason

Where there are exceptions in the process, the transaction is rolled back and the process may be aborted. If appropriate redelivery properties are not set, the message that started the process will be moved to the error queue. A message that is moved to the error queue will not be processed by the business process again. Instead, an internal message-driven bean will read that message and invoke the global exception handler of the business process.

A recommended approach to avoid this situation is to have high values for retry count and retry delay.

Details

The process level properties are:

- ***Retry count:*** Specify how many times, after the first attempt, the process engine should try to execute the business process.
- ***Retry delay:*** Specify the amount of time (in seconds) you want to pass before a retry is attempted

Make sure that retries and retry intervals are appropriate. Retries multiplied by retry intervals should exceed the time it takes for JTA recovery to run.

If you need to tune your retries and retry intervals, you have the following choices:

- Carefully set retry count and retry interval in your JPDs.
- Set the retry count and retry interval of the async and error queues for each JPD project (WebApp). Note that this will break explicit retry settings on the JPD, but it is the easiest and recommended approach.

Message Broker Channels

Best Practice

Prefer using a JMS header value than using a document member.

Reason

Using a header value is much faster than using a document element since no parsing is required.

Details

A message broker channel has similar properties to a Java Message Service (JMS) topic, but is optimized for use with WebLogic Integration processes, controls, and event generators. We recommed creating multiple channels instead of having multiple JPDs listening on a single channel with subscribtion filters.

Best Practice

Create a business process to consume messages in dead letter channel.

Reason

If the channels don't have subscribers or registered subscribers don't match because filter conditions are not satisfied, then the message is placed into the deadletter channel. If there is no subscriber for the dead letter channel, the message that is put on the channel is lost as it is discarded.

Details

When a message is published to a channel and no matching subscribers are found, the message is republished to the dead letter channel that corresponds to the channel's type. WebLogic Integration provides the following dead letter channels:

- /deadletter/xml
- /deadletter/string
- /deadletter/rawData

For example, an unmatched message published to an XML channel (that is, a channel that has messageType = "xml") is routed to the / deadletter/xml channel.
At design time, the dead letter channels are available when you create MB Publish and MB Subscription controls. Your business processes can publish and subscribe to the dead letter channels. For example, you can use the dead letter channels when you design error handling – you can create a business process that includes static subscriptions to the dead letter channels and design error handling code to handle the unmatched messages published to those channels.

The Message Broker module in the WebLogic Integration Administration Console allows you to monitor and manage all of the Message Broker channels in your application, including the dead letter channels.

BP Transaction Boundaries

Best Practice

Keep transaction boundaries in mind while designing a process.

Reason

There are implicit transaction boundaries in the process that will affect the behavior of a process. If they are not considered, it may result in unexpected results. Adding transactional boundaries will introduce a quiescent point for the process definition.

Details

Processes in WebLogic Integration are transactional in nature. Every step of a process is executed within the context of a JTA transaction. A transaction ensures that one or more operations execute as an atomic unit of work. If one of the operations within a transaction fails, then all of them are rolled back so that the application returns to its prior state. Depending on whether you design your business process logic such that your process is stateful or stateless, there may be one or more transactions within the context of a given process.

When you build a process, implicit transaction boundaries are formed based on where in the process you place blocking

elements, for example, a control receive or a client receive. The transaction boundaries within a process change as you add process nodes. You can also create explicit transaction boundaries by selecting contiguous nodes and declaring them to be in a transaction separate from those created implicitly by the application. Resources accessed by a process may also be part of the transaction, depending on the nature of the resource and the control that provides the access.

Some of the nodes that will introduce implicit transaction boundaries are

- Event choice
- Parallel nodes
- Control receives

Stateless Processes

Best Practice

Prefer stateless processes over stateful processes

Reason

Stateful processes are implemented as entity beans and are persisted into the database when there is a quiescent point.

Details

A stateless process is one that is executed in memory only and does not persist its state in the database. In technical terms, it is compiled into a stateless session bean.

Stateless processes are intended to support business scenarios that involve short-running logic and have high performance requirements. Because it does not persist its state to a database, it is optimized for lower-latency, higher-performance execution. An example of a stateless process is one that receives a message asynchronously from a client, transforms the message, and then sends it asynchronously, or synchronously, to a resource using a control. Another example is a process that starts with a message broker subscription, transforms a message, and publishes it to another message broker channel. A common use case in the field is to model a long-running business transaction. This can be built as a group of stateless business processes loosely coupled by publishing messages to channels.

Since stateless processes are compiled into stateless session beans, they ensure that data members only contain non-instance–specific data.

You cannot explicitly configure a process to be stateless. By default, a business process is Stateless until you add any blocking construct to the data flow, such as waiting for a reply or a message. This, in turn forces a transaction boundary and the process becomes stateful. Therefore, a process is stateless if it runs within only one transaction.

Stateful Processes

A stateful process is a process that runs within the scope of more than one transaction. The process has persistent state in the database and is compiled into an entity bean.

Stateful processes are intended to support business scenarios that involve complex, long-running logic and therefore have specific reliability and recovery requirements. A process is made stateful by the addition of stateful nodes or logic that forces transaction boundaries (see Transaction Boundaries). For example, a process that receives a message, transforms it, sends it to a business partner, and then waits for an asynchronous response is stateful because the act of "waiting" forces a transaction boundary.

The list of nodes that, when used, induce a quiescent point are:

- Control receive
- Event choice (except as a start node)
- Parallel branching

Some controls typically require a control receive and therefore will force a process to become stateful. A non-exhaustive list of such controls includes:

- Worklist
- Timer
- Asynchronous two-way process
- JMS subscription
- Message broker subscription

Stateful processes, because they have state in the database, have lower performance than the memory-only stateless processes. However, the persistence of the state is necessary to ensure that:

The process can recover and continue execution without loss of data in the event of a system outage during this waiting period.

System resources are used efficiently during this waiting period. During this time, the entity bean may be passivated, which will free memory on the server.

You cannot explicitly configure a process to be stateful. You know when a process is stateful when the process start node at the top of your process has a green ring.

PARALLEL NODES

Best Practice

Understand parallel nodes before using them.

Reason

Using parallel nodes alters process behavior implicitly. A parallel node changes the transaction behavior and state behavior of the process. Parallel nodes are also only logically parallel.

Details

Parallel branches of execution in a business process are *logically* parallel; physically the branches are executed serially by the business process engine. Business processes benefit from this logical parallelism when communication with external systems can involve waiting for responses from those external systems. While one branch of execution is waiting for a response, another branch of execution in the parallel flow can progress.

Parallel branches are synchronized only at their termination points. A join condition is defined at the termination of multiple branches to specify how the termination of branches terminates the overall parallel activity. Valid join conditions are AND and OR.

WebLogic Integration Database

Best Practice

Use separate schema (not physical database) for WLI database and application database.

Reason

This separates the WebLogic Integration tables and application tables so that different policies can be applied.

Details

Even if you have only one shared database, it is a good idea to use a separate schema (separate user) for WebLogic Integration and the application. Sometimes WebLogic Integration may create tables on the fly in development mode, which means that the WebLogic Integration user ID needs to create permissions, but the application database user is not usually given such privileges. This also separates WebLogic Integration tables and application tables, which helps in archiving and maintaining data.

Configuring Timeouts

Best Practice

Associate timeout paths to ensure that your business processes eventually terminates.

Reason

The event might never happen and the

business process could be blocked forever.

Details

Timeouts can be set on a node, a group of nodes, or on the entire process. The timer on the start node starts when the process begins. The timer on a node or a group starts when the process reaches that point of execution.

Summary

The art of building software has changed dramatically in the last few years, and we don't see a single real-world project that does not use frameworks for specific modules. The projects being built integrate with disparate systems with different service-level agreements and business leaders expect visibility and management at the macro business process level. In keeping with this trend, we expect that business process management systems to become as pervasive as databases in the next five years.

In this article, we discussed best practices for building business processes in BEA WebLogic Integration 8.1, emphasizing characteristics such as scalability, recovery, exception handling, guaranteed delivery, and high performance.

References

- *BEA Weblogic Workshop Documentation:* http://edocs.bea.com/workshop/docs81/index.html
- *BEA Weblogic Integration Documentation:* http://edocs.bea.com/wli/docs81/index.html

WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Acsera	www.acsera.com/register	408-245-1000	9
Compuware	www.compuware.com	800-521-9353	51
H&W Computer	www.hwcs.com/wldj08.asp	800-338-6692	15
HP	http://devresource.hp.com/d2d.htm	800-752-0900	37
Information Storage & Security Journal	www.issjournal.com	888-303-5282	49
Intersperse	www.intersperse.com	626-535-2000	3
IT Solutions Guide	www.sys-con.com	201-802-3021	39
Motive	www.motive.com/within	512-531-2527	52
NetIQ	www.netiq.com/solutions/web	408-856-3000	19
Pantero	www.pantero.com	781-890-2890	13
Quest Software	www.quest.com/weblogic	949-754-8000	29
ReportingEngines	www.reportingengines.com	888-884-8665	21
SandCherry	www.sandcherry.com	866-383-4500	25
SYS-CON Publications	www.sys-con.com/suboffer.cfm	888-303-5282	44-5
SYS-CON Readers' Choice Awards	www.sys-con.com	201-802-3040	31
SYS-CON Reprints	www.sys-con.com	201-802-3026	27
Tangosol	www.tangosol.com	617-623-5782	2
Web Services Edge 2005 East	www.sys-con.com/edge	201-802-3004	43
Wily Technology	www.wilytech.com	888-GET-WILY	5
WLDJ	www.sys-con.com/wldj	888-303-5282	39

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

News & Developments

BEA, IBM, Microsoft, SAP, Sun Submit New Spec

BEA Systems, Inc., IBM Corp., Microsoft Corp., SAP AG, and Sun Microsystems, Inc., have submitted the latest version of a key Web services specification, WS-Addressing, to the World Wide Web Consortium (W3C) as input into the standardization process.

WS-Addressing helps enable organizations to build reliable and interoperable Web services applications by defining a standard mechanism for identifying and exchanging Web services messages between multiple end points.

The joint submission of WS-Addressing represents a milestone in the collaboration among BEA, IBM, Microsoft, SAP, and Sun to further advance Web services technology. Backed by significant industry support, the submission of WS-Addressing is also part of a longer-term effort to provide a standards-based foundation for the development of secure, transacted, asynchronous, and reliable Web services.

WS-Addressing is designed to underlie other specifications, such as WS-ReliableMessaging, WS-Federation and WS-AtomicTransaction, providing a consistent, interoperable, and standards-based mechanism for Web services addressing. www.w3c.org

WebLogic Platform 8.1 Chosen as SOA Infrastructure of Choice

(San Jose, CA) – BEA Systems, Inc., closed several significant customer deals in the fiscal quarter ended July 31, 2004, as customers in major market sectors chose BEA WebLogic Platform 8.1 as a trusted foundation for both service-oriented architecture (SOA) and Linux-based enterprise applications.

BEA sees many large enterprise customers implementing SOA to help reduce IT overhead, facilitate smart workflows and timely information exchange, and accelerate the speed at which change could be implemented. In addition, many customers began deploying enterprise applications on the Linux operating system to take advantage of potentially lower hardware and software costs.

Among the most common solutions being built on BEA are employee self-service and customer self-service portals. These portals can provide personalized views into data and processes to make users as efficient as possible, comprehensive one-stop access to information and functionality, and the adaptability to facilitate rapid change to accommodate the changes in business. www.bea.com

BEA Sets New Price/Performance Record Validating the Best TCO

(San Jose, CA) – BEA Systems, Inc., has announced two new sets of results in SPECjAppServer2002, the industry-standard benchmark that evaluates the performance and cost of Java application servers. The BEA results set a new record for the lowest price-to-performance ratio in the MultipleNode category of the benchmark, reinforcing to customers that BEA can offer not only the highest performance, but can also provide the lowest cost. According to the benchmark, BEA outperformed IBM at less than half the cost, and surpassed Oracle performance at about one fifth the cost of their respective highest throughput results in this category. The results show that BEA WebLogic technology coupled with Unisys hardware can help to further enable customers to benefit from a more scalable enterprise application infrastructure at reduced costs.

BEA outperformed IBM at only 42% of the cost of IBM's configuration. Further, BEA required only four systems vs. IBM's nine systems of a comparable type and power to achieve higher performance. www.bea.com

Intersperse Announces Intersperse Manager 3.0

(Pasadena, CA) – Intersperse, Inc., a provider of management solutions for service-oriented enterprise applications, has released Intersperse Manager 3.0, an application management solution targeting the needs of next-generation J2EE and SOA-based applications. Intersperse Manager 3.0 provides Fortune 500 and other large organizations' performance advances and new levels of scalability that enable users to continually and effectively monitor the health of their enterprise IT environments.

Intersperse Manager 3.0 gives users a consolidated view, providing whole-system visibility and real-time control over an enterprise's distributed or service-oriented application environments. www.intersperse.com

TIM Peru Deploys "One Call" Solution on WebLogic Platform 8.1

(San Jose, CA) – TIM Peru, a subsidiary of Telecom Italia Mobile, has built and deployed its new One Call call-center solution on BEA WebLogic Platform 8.1. This marks a key milestone in TIM Peru's mission to build a service-oriented architecture (SOA) that can enable the company to reduce overhead, improve customer service, and increase revenue.

TIM Peru operates one of the most advanced wireless networks in Latin America, serving more than 800,000 customers. In the past, a TIM Peru representative had to access multiple applications in order to answer a single customer question, often resulting in customer service calls that lasted as long as 15 minutes. In some cases, representatives had to end the calls to track down the necessary information then redial the customer. The One Call solution is designed to utilize an SOA to integrate back-office systems and expose their data and functionality to representatives within a unified desktop.

When evaluating technology platforms upon which to build the One Call solution, TIM Peru considered enterprise offerings from many vendors. They selected WebLogic Platform 8.1 because of its support for open standards and because it can help deliver rapid time to value. www.tim.com.pe

Compuware

OptimalJ®

THE POWER TO Develop, Transform, Reuse

Put your J2EE™ application development into overdrive—with unmatched quality and unprecedented flexibility—using Compuware OptimalJ. Increase developer productivity up to 90%. Seamlessly instill quality into development. Forge existing infrastructures and new technologies into an integrated whole. Realize business agility with applications that are truly reusable, time and time again.

THE POWER IS RIGHT HERE.

THE LEADER IN IT VALUE.

COMPUWARE®
www.compuware.com

Motive brings an enlightened approach to application management, building management intelligence into the application itself. Only Motive transcends the fragmented, disjointed reality of traditional application management, to deliver the visibility, insight and automation that support and development teams need to keep applications running smoothly.

This is no mere vision for the future. It's here now in application management products from Motive. Learn more about Motive's breakthrough approach in a new white paper about built-in management at www.motive.com/within1.

motive

www.motive.com