INTEGRATION • OPTIMIZATION • VOICE TECHNOLOGY

WebSphere®

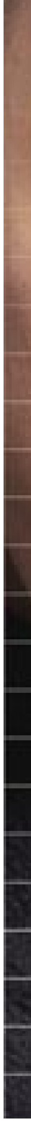
www.SYS-CON.com/WebSphere

The World's Leading Independent WebSphere Resource

MAY 2004 VOLUME 3 ISSUE 5

DISPLAY UNTIL JULY 31, 2004
$8.99US $9.99CAN

Securing Access to the WebSphere Application Server Namespace
BY STEPHEN PIPES PAGE 18



What//: are your Web applications saying to your customers?

diagnoSys

Are your Web applications open 24 hours? Join Sun's expert John VanSant and H&W for a Web seminar series April 6 and June 22 to learn how to make your J2EE applications run at top speed. Register today at: www.hwcs.com/wsj4.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

FROM THE **EDITOR**

IBM Puts Its Stinger in the Competition Again

BY JACK **MARTIN**

IBM is really on a roll as WebSphere continues to shellac BEA in market share and product extensions. Now they go and release Stinger (beta version) for the next-generation DB2 to give Larry Ellison and Oracle something to think about.

It has been over a year and a half since DB2 has been upgraded and it looks like good things do come to those who wait. Stinger vastly simplifies and automates many of the tasks associated with maintaining database management systems. IBM continues to slash development time for users who are building new applications, so that everyone can focus more on aligning the technology to support strategic business initiatives.

Stinger delivers over 200 new features to ease database administration and boost system performance. It even gives the .NET crowd, along with the WebSphere Studio user base, some very husky support.

Some of the new features are for automatically deploying, configuring, maintaining, and optimizing DB2 on the fly. With the introduction of DB2 Design Advisor, it is possible for database administrators (DBAs) to complete jobs 6.5 times faster than if done manually. Design Advisor tunes the database on demand as the workload fluctuates, automating any changes to the database structure, as well as backups and restores. The DB2 Design Advisor also suggests to DBAs how complex queries can be accelerated, providing the shortest path to the requested information. It does so by learning from the performance of previous information searches and by collecting, precomputing, and keeping commonly used information at the ready.

They have even included support for three-dimensional geospatial data that is location and time-and-space aware, and which enables you to build spatial applications. Restaurant chains and retailers sometimes use spatial data to identify new business opportunities, but the real big winner here is any business that engages in direct marketing.

And the Linux march continues, with a new focus on clustering that automatically partitions and optimizes large databases on many servers in just a few minutes instead of hours.

Stinger is also the first deployment of query optimization technology from IBM's LEO (learning optimizer) research and development project. LEO is the next generation of IBM's query optimizer technology, in which the database automates, simplifies, and accelerates queries without human intervention. With LEO, DB2 will now automatically and continually update query statistics about how the database is being used, where it keeps information, and how it is performing. As a result, DB2 now automatically creates and executes better plans for accessing data without prompting the DBA to take action.

By tapping existing in-house skills, you can bring DB2-based solutions to production faster. Stinger continues to support new tools that take advantage of the latest application development features of Java/Eclipse and Microsoft .NET, which are available to DB2 users even before Microsoft SQL Server customers. One of the SQL enhancements included with Stinger is the ability to write stored procedures using .NET languages such as Visual Basic .NET and C#. This capability enables developers to write their applications in the same programming language from start to finish.

I recently spoke with Jeff Jones, the director of strategy for DB2, and he told me, "No other release of DB2 has focused so strongly on automating administrator and developer tasks." When I asked him how all of these new features would affect the WebSphere community, he said, "This version of DB2 continues the strong inte-

–*continued on page 40*

Jack Martin, editor-in-chief of ***WebSphere Journal***, is cofounder and CEO of Simplex Knowledge Company, an Internet software boutique specializing in WebSphere development. Simplex developed the first remote video transmission system designed specifically for childcare centers, which received worldwide media attention; and the world's first diagnostic-quality ultrasound broadcast system. Jack is coauthor of ***Understanding WebSphere***, from Prentice Hall. jack@sys-con.com

WebSphere® JOURNAL

Advisory Board
Richard Arone, David Caddis, Mike Curwen, Devi Gupta, Lloyd Hagemo, Barbara Johnson, Shannon Lynd, James Martin, Doug Stephen, Victor Valle

Editor-in-Chief	JACK MARTIN
Executive Editor	GAIL SCHULTZ
Contributing Editor	PATTI MARTIN
Product Review Editor	JAY JOHNSON
Special Projects Editor	RICHARD ARONE
Editor	NANCY VALENTINE
Associate Editors	TORREY GAVER
	JAMIE MATUSOW
	JENNIFER VAN WINCKEL

Writers in This Issue
Atul Andharia, Bibhas Bhattacharya, Brian Garr, David Murphy, Dwight Peltzer, Stephen Pipes, Duncan Ross, Rick Souler, Mike Storey, Colin Yu, Kareem Yusuf

Subscriptions
For subscriptions and requests for bulk orders, please send your letters to Subscription Department.
SUBSCRIBE@SYS-CON.COM
Cover Price: $8.99/Issue Domestic: $149/YR (12 Issues)
Canada/Mexico: $169/YR Overseas: $179/YR
(U.S. Banks or Money Orders)
Back issues: $15 U.S. $20 All others

PRESIDENT AND CEO	FUAT KIRCAALI
PRESIDENT, SYS-CON EVENTS	GRISHA DAVIDA
GROUP PUBLISHER	JEREMY GEELAN
SENIOR VP, SALES & MARKETING	CARMEN GONZALEZ
VP, INFORMATION SYSTEMS	ROBERT DIAMOND
VP, SALES & MARKETING	MILES SILVERMAN
PRODUCTION CONSULTANT	JIM MORGAN
FINANCIAL ANALYST	JOAN LAROSE
ACCOUNTS RECEIVABLE	CHARLOTTE LOPEZ
ACCOUNTS PAYABLE	BETTY WHITE
ADVERTISING DIRECTOR	ROBYN FORMA
DIRECTOR OF SALES AND MARKETING	MEGAN MUSSA
ASSOCIATE SALES MANAGERS	KRISTIN KUHNLE
	BETH JONES
CONFERENCE MANAGER	LIN GOETZ
LEAD DESIGNER	LOUIS F. CUFFARI
ART DIRECTOR	ALEX BOTERO
ASSOCIATE ART DIRECTORS	RICHARD SILVERBERG
	TAMI BEATTY
WEB DESIGNERS	STEPHEN KILMURRAY
	CHRISTOPHER CROCE
CONTENT EDITOR	LIN GOETZ
CIRCULATION SERVICE COORDINATORS	SHELIA DICKERSON
	EDNA EARLE RUSSELL
	LINDA LIPTON

Editorial Offices
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9637
SUBSCRIBE@SYS-CON.COM
WebSphere® Journal (ISSN# 1535-6914)
is published monthly (12 times a year).
Postmaster send address changes to:
WebSphere 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

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

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

Not any more.

Get Wily.™

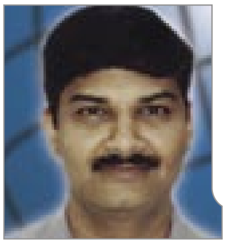
Enterprise Java
Application Management
1 888 GET WILY
www.wilytech.com

wily technology

Maximize efficiency in working with queue managers

Meeting the Needs of Administrators and Developers

BY ATUL **ANDHARIA**



WebSphere MQ, formerly known as MQSeries, is industry-leading middleware created by IBM Corporation. Due to its assured delivery of messages, data integrity and resource protection, time independence and message-driven processing, WebSphere MQ has become very popular as message-oriented middleware in the integration of applications enterprise wide. But working on WebSphere MQ as an administrator or developer is not so easy.

The majority of WebSphere MQ installations are in distributed environments that have more than one queue manager on heterogenous platforms. To work with all of these queue managers, you can use a Command line utility called "runmqsc" provided by IBM, on all platforms. IBM has also provided a graphical tool, WebSphere MQ Explorer (formerly MQSeries Explorer), which is the most reliable but has limited functionality. For example, you can only view the first 200 messages on a local queue. Moreover, you can view the data in Text and Hexadecimal forms only with other headers, if present. The RFH headers and data in EBCDIC are displayed as junk characters along with the real data. So, a tool is needed that provides you with various administration capabilities and development facilities with one click of a mouse.

WMQTool is a graphical tool I developed that addresses most of the needs of administrators and developers. It is developed in J2SDK 1.4.1. Using WMQTool, you can administer as many queue managers as you want, organized into different categories in a tree structure, on just a single window screen (see Figure 1).

You can have views of queues, channels, processes, client connections, and namelists in a sorted tabular form. You can select your own set of columns of these views and put them in any order. You can create any objects, update their properties, or delete them. Different signs are provided to visually indicate the status of the objects. You can view an indefinite number of messages, depending upon the availability of such hardware resources as memory and space on the hard drive. Data and headers can be viewed separately in a readable form, allowing you to cut and paste the readable data.

Cluster queues and channels can be viewed along with your other queues and channels. You can easily eliminate viewing unnecessary objects by setting a "Filter" on them, thereby leaving only important objects for you in views. You can identify the platform and WebSphere

Atul Andharia is a software engineer who has worked in the field since 1984 in various industries and positions, including three years of experience in IBM's MQ Series middleware.
atul@niratul.com

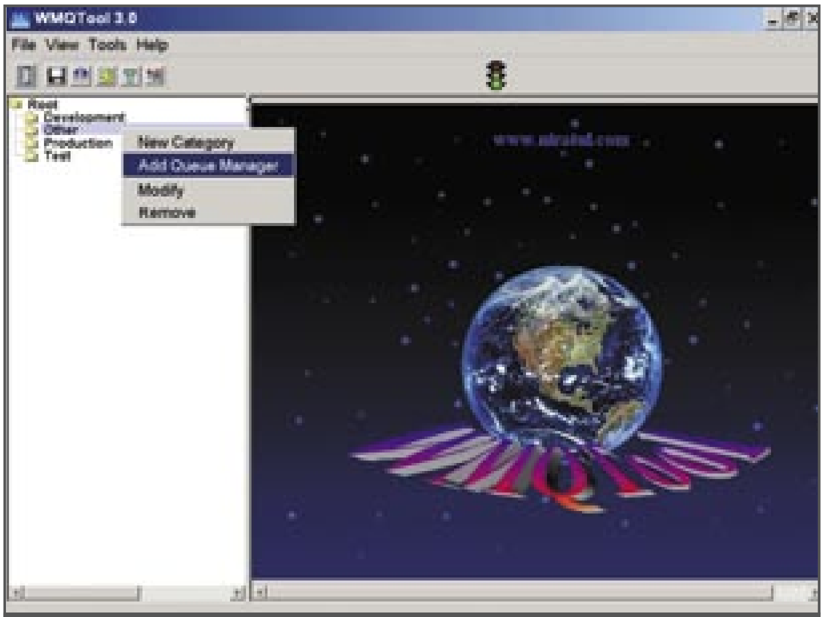


FIG 1: ADMINISTERING QUEUE MANAGERS IN A SINGLE WINDOW

MQ version of the queue manager on which you are currently working at the right bottom of the main window.

For WebSphere MQ Integrator developers, you can put a testing message on any queue with or without RFH headers. You can view any kind of data, ASCII or EBCDIC, in XML format or in an XML tree structure or plain text format. It's easy to locate a particular range of bytes within a large message. You can edit the data of the message so as to change any particular byte/s. If you so choose, you can remove a Transmission header or Dead Letter header, if present, from the messages and put them back on a different queue. You can take messages that have been backed up in a file on your hard drive and restore them to any queue whenever you require.

And what about security? Yes, WMQTool also has provisions for security. You can implement Security Exit of WMQTool and be assured that only authorized people can work on WMQTool and your queue managers. WMQTool provides security in very detailed levels, meaning that you can selectively allow users of WMQTool to work on only certain types of objects. For example, if you want to restrict application programmers from altering any queues but want to allow them to turn triggers on or off, or simply let them view the objects, you can do it easily with WMQTool.

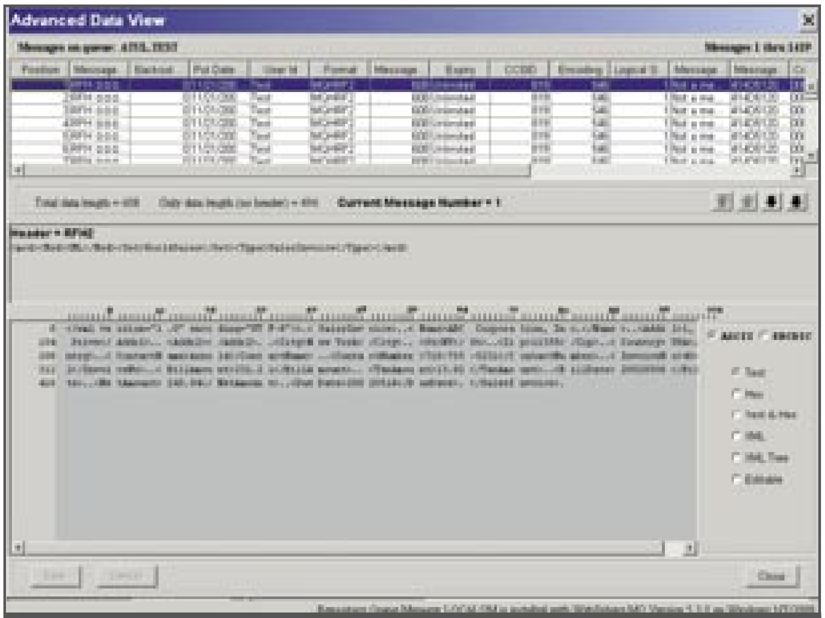


FIG 2: AN ADVANCED VIEW OF QUEUES AND CHANNELS

Building a Robust Integration Framework

Integrate applications and machines without major ripple effects

BY BIBHAS **BHATTACHARYA**

This article explores certain approaches that will result in a robust integration solution.

Messaging technology is designed to deal with requirements. Home-grown solutions can face significant challenges to meet these needs.

The rest of this article will deal with IBM's WebSphere MQ as the messaging product. The principles described here apply to any messaging product.

Most integration systems are given these NFRs.

1. Applications being integrated may not be running at all times.
2. The network layer between the systems may not be available at all times.
3. The same message may not be processed twice (unless a message is being retried due to earlier failure in processing).
4. Information sent to an application must belong to a transaction along with other activities, such as database updates. The information should be sent only if all of those activities go through.

The Principles of Messaging

Most developers are trained to write synchronous applications. In the popular programming languages, method calls are inherently synchronous in nature. Messaging requires a different way of thinking. To help the developers understand the uniqueness of messaging, I have the following maxims.

DO NOT ASSUME TIMELINESS OF MESSAGE DELIVERY

A message sent to a system will be delivered on a best effort basis. Systems cannot ensure delivery within a specific time period. For immediate delivery, consider using a synchronous technology, such as Enterprise JavaBeans (EJB).

WebSphere MQ Client–based programs can also do an immediate delivery of messages. In this case, the application connects to a remote queue manager using TCP/IP and sends the message directly to a queue in the remote queue manager. In both cases, we fail to meet NFR #2.

DO NOT ASSUME THAT THERE IS ONLY ONE RECEIVER PROCESS

There may be more than one process picking and processing messages from a single queue. WebSphere MQ ensures that a message is delivered to one process only. Multiple receiver processes are run for scalability reasons. We can process more messages per minute, especially in a machine with multiple CPUs.

If the queue is clustered, the receiver processes may even run on multiple machines. This leads to the next maxim.

DO NOT ASSUME THAT A MESSAGE WILL BE DELIVERED TO A SPECIFIC MACHINE

The sender should not have any assumption about the specific machine to which a message will be delivered.

WebSphere MQ clustering allows you to host the same logical queue in multiple queue managers running in different machines. When you put a message in such a queue, the system will perform load balancing and fail over to distribute the messages among the queue managers in the cluster.

DO NOT ASSUME A SINGLE SENDER APPLICATION

Messages may be put in a single queue by applications running on different machines.

DO NOT ASSUME THE ORDER OF DELIVERY

Systems do not necessarily deliver messages in the order in which they were placed. In addition, messages placed by one application may be interspersed with messages placed by another application (if there are multiple senders). If the order of delivery is important, use the message grouping feature of WebSphere MQ. Messages in a group are delivered to a single receiver application in the strict order in which they are placed.

Communication Patterns

DATAGRAM

A datagram message is a one-way form of communication. The sender does not wait for a reply to come back from the receiver.

REQUEST RESPONSE

In this pattern, the sender sends a message and waits for a reply to come back from the receiver. The response is correlated with the request message using a correlation ID. The correlation ID of the reply message is usually set to be the same as the unique message ID of the request message. The sender instructs the GET operation to wait for a message with a specific correlation ID.

This pattern is an approximation of synchronized communication. In general, avoid request response type messaging.

DATAGRAM WITH CONTEXT

In this pattern we use datagram communication. But, the messages going back and forth between the systems can be correlated. For example, when an order is placed in a Web-based application, the system saves the order in the database and sends the order details message to the back-end ERP system. The order ID primary key is a part of the message. In a few days, the ERP system ships the order and sends an order status update message to the Web-based system. The order ID is again a part of this message. The Web-based system uses the order ID to update the order information in the database. In this case, the order ID serves as the context. Datagram with context is the most robust mode of communication.

Problem Domain

The integration solutions I am commonly asked to develop fall into two categories:
1. Web tier to back-end integration.
2. Retail location to head office integration.

WEB TIER TO BACK-END INTEGRATION

Here, one system is a Web-based application. The application is probably clustered in multiple machines. The other system is a legacy application. Datagram with context works best in a situation like this. Database primary keys of orders, users, or other entities are passed around as a part of the message. The latest information is stored by each system in its own database. Information in the database is updated upon the receipt of a message by the system.

In some cases, primary keys are too related to the internal implementation of a system and have little business meaning. In that case, unique key index values are used as context. For example, for an order, an enterprise-wide order ID is generated by the Web-based system (in addition to a primary key value).

RETAIL LOCATION TO HEAD OFFICE INTEGRATION

Multiple retail locations need to send daily order details data to a central location. The central location (known as head office) needs to push out price updates and promotion details to the retail locations.

The challenge is that outgoing messages from the head office to the retail locations may be store specific. The system should be able to pick the correct queue in which to put the message. In this article, we will not be exploring that problem domain.

Laying Down the Framework

WEB TIER TO BACK-END INTEGRATION

The Web-based system may be clustered. If it is not, start with the assumption that it may be clustered in the future. Machines in the cluster should be indistinguishable from one another. That is, the machine ID (host name or IP address) should not be featured as a context in a message. Contrast this with multiple retail locations. Here, the location matters. Messages may be store-specific.

The use of WebSphere MQ clustering works very well with WebSphere clustering. Create a queue manager in each Web tier application server machine. The queue manager names must be different in a network. Create one local queue in each queue manager to receive messages. The queue name must be the same in all cases (WEB_Q in our example).

In the back-end system, create a queue manager and a local queue to receive messages. Set up an MQ cluster that involves all queue managers. Share all local queues in the cluster.

In the Web-based application, develop a Message-Driven Bean (MDB) to act as the listener. The MDB will be bound to the local queue manager (WEB_QMA or WEB_QMB) and the local queue (WEB_Q).

Such a design will have several advantages:
1. MQ clustering is a simpler way to set up remote communication than using a remote queue definition and transmission queue. If a new machine is added to the Web tier cluster, we can very easily add it to the MQ cluster.

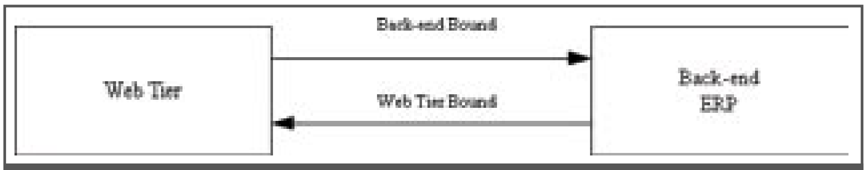


FIG 1: WEB TIER TO BACK-END INTEGRATION

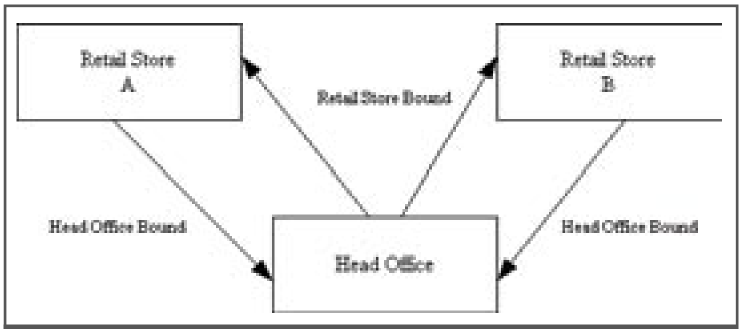


FIG 2: RETAIL LOCATION TO HEAD OFFICE INTEGRATION

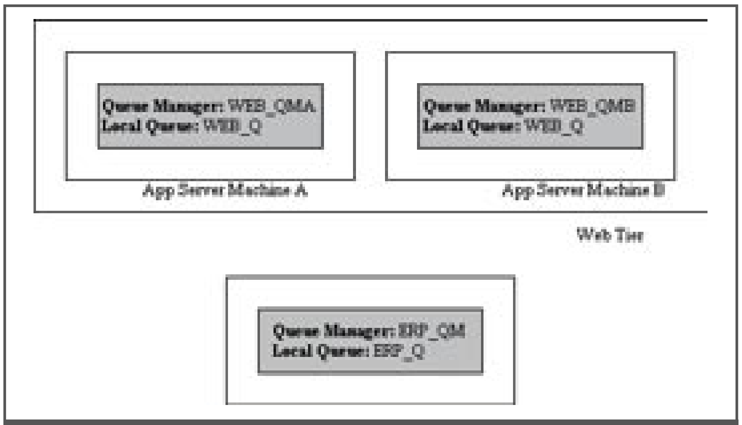


FIG 3: AN MQ CLUSTER THAT INVOLVES ALL QUEUE MANAGERS

2. Web tier bound messages will be load balanced between the available Web tier application server machines. This increases the throughput level and adds redundancy to the system.

BACK-END BOUND MESSAGES

The back-end queue manager participates in the MQ cluster and has shared its local queue (ERP_Q). As a result, ERP_Q will automatically appear as a local queue to the Web tier queue managers. We can create a JMS destination queue in the application server that refers to ERP_Q. At this point, any Servlet or EJB can put a message in the ERP_Q. Eventually, these messages will be transferred over to the back-end queue manager (ERP_QM) and physically stored in ERP_Q.

If you are following the Datagram with Context pattern, the Web tier must update the database and put in a message as a part of the same transaction. As a result, a session EJB (preferably stateless) is an ideal choice to perform that task.

WEB TIER-BOUND MESSAGES

Once again, due to MQ clustering, the Web tier local queue (WEB_Q) will appear to be a local queue to the back-end queue manager (ERP_QM). Any message put in this queue by the back-end system will be routed to one of the Web tier queue manager's local queue. The system will perform necessary load balancing and fail over.

MESSAGE FORMAT

A common problem with messaging is that any alteration of the message format creates a major ripple effect among all of the systems participating in the integration. A well-designed format should be relatively easy to extend or modify. Below are commonly used formats and their advantages.

- ***Character aligned:*** In this format, the sequence of fields and the number of characters for each field is precisely regulated. Text of this type is easy to read from a C or COBOL program without any need for parsing. Many legacy systems use this format. Writing the code by hand to create a message buffer from a Java object and initialize a Java object from a message buffer can be extremely time consuming. Use IBM's Extended Messaging technology to simplify working with strict messaging formats from a WebSphere application. Also, do not adopt this type of format in a new system. This is the most "brittle" format. Any change in the format will require careful code review and alteration for all systems.
- ***Comma separated:*** In this format, the length of a field can be anything. The sequence still matters. No standard parser exists for this format. We have developed an XML-style parser. The parser uses a schema file where the sequence of the fields is defined. This allows us to change the sequence and size of a field without necessarily doing major code changes to the systems.
- ***XML:*** This is by far the most powerful message format. It does not assume the order of attributes of an element. The order of elements themselves can be fixed or free. Standard parsers exists in many languages. XML is also great for multilingual text data.
- ***Java object:*** Java objects can be serialized and deserialized in a queue. Both JMS and MQ Java API allow us to do that. Systems can send each other fairly complex data structures with very little programming. If there is any change in the Java class code, make sure that you copy the latest class file to all the systems. This format only works when all systems in the integration are Java-based.

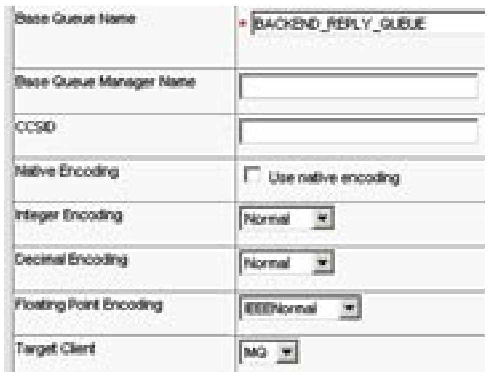
By default, WebSphere adds a JMS-specific XML header at the beginning of a textual message. If your back-end system is not using JMS, this will cause obvious problems. To avoid this, set the Target Client property of the queue destination to MQ.

JMS listeners do not require this field to be set to JMS. As a result, you can safely configure the client type of all JMS destinations to MQ.

TRANSACTION

In the datagram with a context communication pattern, it is essential that we perform a distributed transaction involving the database and WebSphere MQ. Fortunately, WebSphere does all of the hard work for us. All we have to do is:

1. Put in a message and update the database from an EJB, preferably a stateless session EJB and MDB (on the receiver side). All JMS and JDBC operations done within an EJB method automatically belong to a distributed transaction.
2. Make sure that the EJB is using a data source to open the database connection. Opening a plain JDBC connection using DriverManager.getConnection() will not give us a distributed transaction.
3. Make sure that the EJB looks up the data source using a resource reference. We have seen WebSphere crash repeatedly when the data source is looked up using its JNDI name and the connection from it is used in a distributed transaction.



Base Queue Name	* BACKEND_REPLY_QUEUE
Base Queue Manager Name	
CCSID	
Native Encoding	☐ Use native encoding
Integer Encoding	Normal
Decimal Encoding	Normal
Floating Point Encoding	IEEENormal
Target Client	MQ

FIG 4: **SET THE TARGET CLIENT PROPERTY OF THE QUEUE DESTINATION TO MQ**

4. Make sure that you are using an XA-compatible JDBC driver. We could not get the new DB2 Universal JDBC driver – com.ibm.db2.jcc.DB2XADataSource – to work (the non-XA driver works just fine). In any case, the XA-compatible Universal driver currently supports Type 2 mode only. Given all that, we recommend using the old legacy driver – COM.ibm.db2.jdbc.DB2XADataSource.
5. Make sure that the JMS connection factory has XA enabled.
6. In case of a business rule violation or system error, roll back the transaction (invoke setRollBackOnly() on the MessageDrivenContext or the SessionContext object) (see Listing 1).

JMS Administration Hints

You will need to install WebSphere MQ in all WebSphere Application Server machines. Create the queue managers, local queues, and the cluster. Ideally, this should be scripted using the MQSC commands. Start the listeners and the sender channels. Use the MQ test programs to make sure that remote communication is working.

Now all we have to do is create various JMS objects (such as connection factory and queue destination) in WebSphere administration console.

In the WebSphere administration console, set the MQ_INSTALL_ROOT variable for all nodes. Without this, the JMS server will fail to start up.

WebSphere allows you to create JMS objects for three types of JMS providers.

1. ***WebSphere MQ JMS provider:*** This is the provider to use if you are using WebSphere MQ.
2. ***WebSphere JMS provider:*** This is a built-in all-Java JMS server, which is used mainly within process messaging.
3. ***Generic JMS provider:*** Any other JMS-compliant messaging product can be configured here.

In the case of the WebSphere MQ JMS provider, a queue connection factory object is simply a link to a queue manager. Creating a queue connection factory does not actually create the queue manager. You need to create the queue manager separately (using the crtmqm command). The same concept applies to a JMS queue destination. In this case, a destination is a reference to a MQ queue.

A JMS connection factory or destination can be defined at the cell, node, or application server level. As we discussed in the framework section, the MQ queue manager names must be unique in the network. This means that we should define our queue connection factory at the node level. The factories will share the same JNDI name but will have a different MQ queue manager name. It is important that we keep the JNDI name the same to ensure that the resource references for the queue connection factory in the enterprise application will work in all of the machines.

In an MQ cluster, the shared local queue names in each application server machine must be the same. This allows us to define the JMS destination at the cell level. To be consistent, you may also define them at the node level.

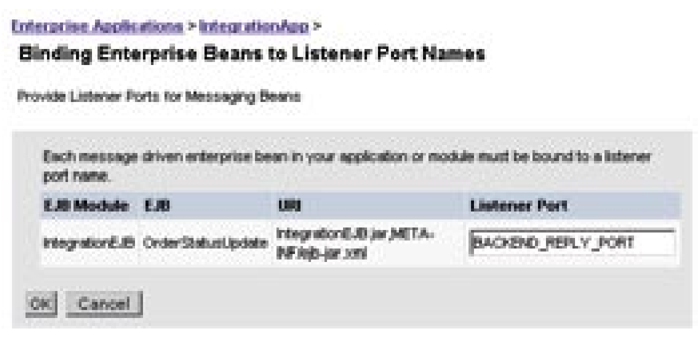
An MDB is bound to a JMS listener port. A listener port is just a JMS connection factory (queue manager) JNDI name and destination (queue) JNDI name pair. In WebSphere, a listener port is defined for an application server. Currently, all application servers in a cluster must be configured individually. Take extra time to define a listener port for each one of them. The port name and the JNDI names will be the same in all application servers. An MDB is bound to a listener port at the enterprise application level. You can look up the valid name of the listener port for an application by opening the application's properties screen in WebSphere administration console and clicking on the Provide Listener Ports for the Messaging Beans link.

Conclusion

Messaging technology solves a very unique problem when all parties involved in a communication may not be present or reachable. WebSphere clustering in conjunction with MQ clustering changes the definition of sender and receiver. We need to think in terms of sending and receiving applications instead of processes. Multiple processes running in multiple machines can constitute a sender or receiver.

The tips and best practices mentioned in this article will help you create a messaging architecture that is robust, fault tolerant, and easy to manage.

A former lead developer of WebSphere Commerce Suite at the IBM software lab, Bibhas Bhattacharya is the chief technology officer at Web Age Solutions. The company is one of the main WebSphere training vendors after IBM Learning Services. It aids large businesses in building large-scale WebSphere implementations using a variety of practices.
bibhas@webagesolutions.com



Enterprise Applications > IntegrationApp >

Binding Enterprise Beans to Listener Port Names

Provide Listener Ports for Messaging Beans

Each message driven enterprise bean in your application or module must be bound to a listener port name.

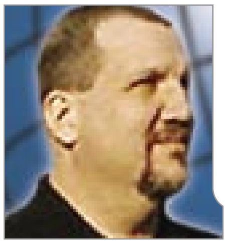
EJB Module	EJB	URI	Listener Port
IntegrationEJB	OrderStatusUpdate	IntegrationEJB.jar,META-INF/ejb-jar.xml	BACKEND_REPLY_PORT

FIG 5: **AN MDB IS BOUND TO A JMS LISTENER PORT**

Performance tuning J2EE code

Optimizing Code to Minimize the Impact of Heap Management

BY MIKE **STOREY**



Performance tuning practices are sometimes based on the run-time characteristics of vendor-specific Java Virtual Machines (JVMs) or application servers. Because Java code will likely outlast the environment for which it is initially designed, developers must ensure that their code has the flexibility to adapt to new technologies and environments.

Performance analysis of an application often yields the vague diagnosis of "excessive object creation." Upon further investigation, many developers learn that correcting performance problems requires significant changes to the design and code of an application. As a starting point, developers must have a basic understanding of Java memory management before they can accurately diagnose performance problems and take action to eliminate bottlenecks.

Each JVM implementer must determine how to manage shared memory, which is also known as the heap. The techniques that often yield the most significant performance gains address heap management optimization. Within the JVM, heap management falls into two broad categories: object allocation and garbage collection. Object allocation determines how a Java object – or primitive – receives memory from the heap. Garbage collection manages how objects are released – or de-allocated from the heap – to free memory.

Not all JVMs are not created equal. Most vendor-centric JVMs are distinguished by their heap management optimizations. Some heap management optimizations are most effective in server-side applications that have high transaction volumes and can tolerate significant run-time pauses to increase overall throughput. Other algorithms are suited for interactive graphical user interface (GUI) applications that have low transaction volumes and cannot tolerate noticeable performance delays. Further, newer JVMs have additional local allocation space management capabilities that simplify memory management. For the purpose of this column, new JVMs are IBM JVM v1.3 and higher, and Sun Microsystems and BEA Systems JVM v1.4 and higher.

Development teams must understand how each vendor optimizes its JVM and associated heap management to write code that will yield the highest performance for its specific environment. Code optimized for an IBM JVM may not, for example, provide the same level of performance in a Sun JVM.

Nine Common JVM Heap Management Optimizations

Various JVM memory management terms are discussed below. Many of the terms describe techniques used by vendors when creating a JVM and are key to understanding coding best practices that can improve performance.

PRIVATE ALLOCATION SPACE

A private allocation space, sometimes called the Thread Local Heap, is a block of memory allocated to each thread. The memory is used for object allocations. This technique eliminates the need to acquire a lock on the heap, and significantly improves object allocation times and overall throughput. Most JVMs implement some form of this technique.

STOP THE WORLD

Current garbage collection requires that at least part of the memory de-allocation process has exclusive access to the heap. This implies that garbage collection stops all other application activity for some time. Most collection optimizations seek to minimize the time spent in this state.

OBJECT MARKING

This is the first step in garbage collection. Most JVMs identify, or mark, all reachable objects. Subsequently, all nonmarked objects are collected. The ability to eliminate most memory leaks is the key advantage of this technique.

COPY COLLECTORS

A copy collector, as the name implies, copies noncollectable objects from one memory space to another during the last phase of garbage collection. This is CPU intensive in terms of operations, but the expense is easily offset by the elimination of a compression cycle during collection.

SWEEP COLLECTORS

A sweep collector releases memory for all objects that were not marked during the mark phase of collection. While a sweep collector will typically be faster than a copy collector, it will fragment memory in the heap, which then requires compression.

HEAP COMPRESSION

Used in conjunction with sweep collectors, this phase of garbage collection is used to defragment memory. This process can be very expensive, and compression collectors typically employ strategies to make the compression optional, occurring only when fragmentation has become severe enough to interfere with efficient allocation.

CONCURRENT COLLECTION

Some JVMs employ a concurrent mark approach that allows the mark phase of collection to run in the background during application execution. When heap sizes are large (640MB and higher), this technique can provide significant gains in JVM throughput by minimizing time in the stop-the-world state.

PARALLEL COLLECTION

Some JVMs employ parallel mark and sweep approaches that will utilize multiple threads on a machine with multiple CPUs. This technique can increase performance by decreasing the time spent in the stop-the-world state.

GENERATIONAL COLLECTORS

JVMs that employ a generational garbage collection approach divide the heap into segments for newer and older objects. Because a large percentage of objects are used for a short time, the generational garbage collector can manage a small space containing only new objects. As they age, objects are moved to the older generational space. The cost of copying objects between generations is offset by the gains of managing a smaller heap of young objects.

Three Tips for Avoiding Memory Management Performance Traps

Developers must consider a number of variables when optimizing J2EE application performance. Some processes, for example, may accelerate the development process, but cause organizations to spend significant time and resources eliminating performance problems in production. The performance-tuning techniques presented in this column are based on best practices and are designed to provide higher value across the enterprise, rather than optimization techniques that address only a specific JVM environment. Key considerations for three common memory optimization practices follow.

AVOID OPTIMIZATIONS THAT ARE SPECIFIC TO A PARTICULAR JVM OR APPLICATION SERVER

JVMs from IBM, Sun, and Hewlett-Packard Co. (HP) each have unique optimizations for their specific environments. IBM WebSphere Application Server v5 supports multiple JVMs on different platforms. The IBM JVM v1.3.1 is used on the Windows, AIX, and Linux-Intel platforms. Additionally, IBM has specialized JVMs for the AS/400 and z/OS platforms. The IBM JVM v1.3.1 uses a mark sweep-optional compress collector with concurrent-mark and parallel-mark sweep optimizations. Similarly, the Sun JVM is optimized for Solaris server environments and the HP JVM tuned for HP-UX systems. The Sun JVM uses the generational copying collector with its own optimizations. IBM is in the process of deploying JVM v1.4.1 for WebSphere Application Server.

DON'T SACRIFICE CODE MAINTAINABILITY FOR PERFORMANCE GAINS

Some performance tuning techniques improve application performance at the expense of code maintainability. While this approach is sometimes justifiable, it should always be weighed according to the specific requirements of an organization's J2EE environment. Because each vendor has a unique JVM, it is important to note that there is no single approach to optimizing code to minimize heap management overhead. There are, however, benchmarks that show how specific performance tuning techniques will deliver performance gains in one environment while causing performance penalties in another. It is reasonable to assume that sacrificing code maintainability for performance introduces the risk of increasing the total cost of operation for an application.

KNOW YOUR ENVIRONMENTS

Development and management teams should have a good understanding of their JVM performance in the development and production environments. Development teams should use the same JVM in both environments whenever possible. Heap management is the most important feature of an organization's JVM. Understanding heap management's impact on JVM performance will enable organizations to determine which code optimization techniques are best suited for their environments. While the default configurations provide good performance for most applications, operations staff should have a good understanding of garbage collection configuration options in order to tune performance in the production environment. Many best practices for optimizing code are influenced by the characteristics of a specific JVM. This becomes even more significant when deploying WebSphere Application Server v5, which supports heterogeneous horizontal scaling. Because WebSphere Application Server utilizes native JVMs on many platforms, understanding the relationship between performance tuning techniques and JVM optimization techniques is critical.

Code Optimization Principles

There are several guiding code optimization principles related to heap management. As with any design issue, the optimization of an object architecture requires a delicate balance of maintainability and performance. Development teams should minimize object allocation – specifically, object collection. This approach sidesteps many performance problems caused by heap locking, which causes the frequent diagnosis of excessive object creation. Most modern JVMs, including the IBM JVM, implement local allocation space techniques that virtually eliminate allocation overhead.

Do these JVM object allocation techniques render minimizing object allocation ineffective with regard to performance optimization? Some code optimization techniques introduce enough overhead to be detrimental to performance. However, object allocation is only part of the performance equation. With very few exceptions, most allocated objects are collected after a very short life. Garbage collection is a significant factor in optimizing application code for performance. Minimizing object collection – and by inference, allocation – can be a very effective way to optimize an application.

In addition, there are a number of infrastructure tuning measures that can be used to optimize heap management. Each application, however, will have a unique set of garbage collection characteristics. These characteristics will be largely influenced by the object allocation/collection patterns used. The optimization techniques below strive to minimize object collection by increasing the lifetime and reuse of an object. In doing so, we reduce the number of objects allocated and collected. Effective reduction in object allocation and collection will result in more significant performance gains. Techniques for optimizing heap management include the following.

OBJECT POOLING

This technique creates a pool of objects that are checked out when needed and returned to the pool when processing is complete. J2EE provides several examples of this technique. Most J2EE developers are already familiar with database connection pools, and the various Enterprise JavaBean (EJB) pools used for all EJBs (with the exception of stateful session beans). In addition, the application server implements thread pools for the Web container and the CORBA engine in the EJB container. Use this pattern with caution, limiting its use to objects that are expensive to create or where a small number of objects can serve a large body of requestors. Expensive objects, such as database connections or large collections, are good candidates for pooling. Creating an efficient and effective pooling framework can be difficult. The overhead involved in creating and managing a pool can offset the benefits gained in some JVMs. The IBM JVM will typically produce a greater benefit from this technique than JVMs that use a generational collector. The Apache Commons Pool project and WebSphere Application Server v6, however, will each provide enhanced pooling capabilities to simplify this process.

INCREASING OBJECT SCOPE

Increasing the object scope involves moving objects that are created in a narrow scope outward. A broader-scoped object that is reused, however, reduces object allocation and collection – and provides better performance than constantly creating and discarding new objects. This would include such simple tasks as moving object instantiation outside of a loop and then reusing an object, as shown in Listings 1 and 2.

In Listing 2, the MyObject class had to be modified to perform initialization in the init method rather than in a constructor. Other examples of this pattern could include moving a variable to class or static scope, then reusing it across multiple activities. These moves can be counterintuitive for developers who have been coached to scope variables to the narrowest possible visibility to help minimize application defects. For this reason, class or static scoping of objects for purely performance reasons is discouraged because it sacrifices code maintainability for performance. Also, scoping objects beyond the method level can introduce the need to synchronize access to these objects. In this case, performance gains are somewhat dubious because concurrency bottlenecks can cost more than the optimizations yield.

SINGLETONS

This pattern increases the object's scope and lifetime by creating a single instance of the object. Developers can use a "singleton factory" to get a reference to this single instance. Note that singletons must be written to be thread-safe. The potential for concurrency bottlenecks must be considered when evaluating this approach. The J2EE multithreaded servlet is an example of a singleton pattern. Creating thread-safe objects is not a simple task because it is possible to create synchronization bottlenecks that offset performance gains. Creating thread-safe objects

can be very difficult. However, this process can impede performance by allocating and discarding these variables on each invocation. As has been discovered with servlets, high-use singletons usually yield higher performance when minimizing synchronization, instead of object allocation and collection.

FLATTENING OBJECT ARCHITECTURE

This technique strives to minimize object allocation and collection by defining fewer objects. Developers who have recently moved from non–object-oriented languages, such as COBOL or C, can find this especially challenging. They have often gone through a difficult transition that included learning how to separate a group of related operations into a new class. The justification for this increased level of granularity is a reduction in costly application defects and development effort by code reuse. It is important to be cautious when flattening the object architecture because it can compromise application maintainability or quality for performance. At the same time, overly complex frameworks can introduce a huge overhead in object allocation/collection and call stack processing.

USE IMMUTABLE OBJECTS CAREFULLY

The most obvious example of this is the String class. Because strings are immutable, performing operations like concatenation (e.g., s1 += s2;) will actually allocate a new string (s1) and discard the old string (s1). It would appear that a simple solution would be to use the StringBuffer class. However, most container classes, including the StringBuffer class, use arrays in the underlying implementation. Arrays, while not immutable objects, are static in size. The StringBuffer class and other containers allow objects to grow in size by creating larger arrays as needed and copying the contents of the old array into the new one. This has performance implications in terms of collection of the old array, as well as the processing overhead of the array copy. While the use of the StringBuffer class in place of the String class is sound performance advice, it is not the single key to optimization. The construction of any container object should specify an initial size in order to avoid the overhead associated with the growth and copying of the underlying array.

Summary

It is important for developers embarking on performance optimization to be aware of several key issues. First, understand the underlying JVM characteristics that influence performance optimizations. Development teams should also have a clear understanding of the environment in which the application will be deployed, including the platforms and JVMs that will be used. Above all, exercise caution with the selection and application of optimization techniques. Remember, your code should outlast the deployment environment and must have the flexibility to adapt to new technologies and environments.

Resources

- Borman, Sam (2002). *Sensible Sanitation – Understanding the IBM Java Garbage Collector, Part 1: Object Allocation (Series)*, IBM developerWorks: www-106.ibm.com/developerworks/java/library/i-garbage1/
- Goetz, Brian (2004). *Java Theory and Practice: Garbage Collection and Performance (Series)*, IBM developerWorks: www-106.ibm.com/developerworks/java/library/j-jtp01274.html
- Sun Microsystems Inc. (1999). *Tuning Garbage Collection with the 1.3.1 Java Virtual Machine:* http://java.sun.com/docs/hotspot/gc/
- IBM (2003). *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.3.1, Diagnostics Guide:* www-106.ibm.com/developerworks/java/jdk/diagnosis/diag131rev1.pdf
- IBM (2003). *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.1, Diagnostics Guide:* www-106.ibm.com/developerworks/java/jdk/diagnosis/diag141sr1.pdf

Mike Storey is a solutions architect with Candle's Application Infrastructure Management consultancy. He has more than 19 years of experience in IT with an emphasis on complex systems architecture and design. His areas of focus include service-oriented architectures, J2EE, and message-oriented middleware.
mike_storey@candle.com

LISTING 1: ALLOCATES AND COLLECTS 11 INTEGERS AND 100 MYOBJECTS.

The integer y is allocated on line 02 and discarded on line 06 each time the outer x loop is executed. The object ("obj") is allocated on line 03 and discarded on line 05 each time the inner loop y is executed.

```
01 for (int x = 0; x < 10; x++) {
02        for (int y = 0; y < 10; y++)
{
03                   MyObject obj = new
MyObject(x,y);
04                   // use obj in some
way
05        }
06 }
```

LISTING 2: ALLOCATES AND COLLECTS TWO INTEGERS AND ONE MYOBJECT

This code is functionally equivalent to that shown in Listing 1. However, the object instantiation is moved out of the loops to lines 01, 02 and 03, eliminating the churn of object creation.

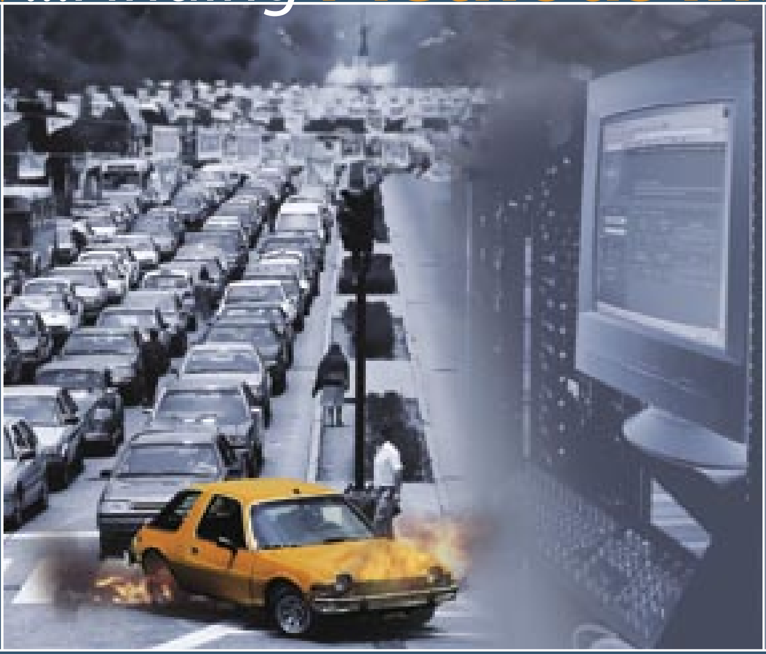
```
01 int x;
02 int y;
03 MyObject obj = new MyObject();
04 for (x = 0; x < 10; x++) {
05        for (y = 0; y < 10; y++) {
06                   obj.init(x,y);
07                   // use obj in some
way
```

Cyanea™

What's Obstructing Your Application Flow?

With Cyanea/One™, your company will gain valuable insight into its J2EE applications. With a few mouse clicks, you can quickly drill down into your applications, all the way down to the method level. Isolate, diagnose and proactively resolve bottlenecks and resource consumption issues. Accomplish all these without touching your code or even requiring knowledge of your application.

...Finding **Methods in the Madness.**

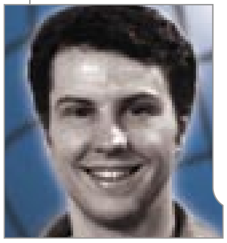


Find methods at:
www.cyanea.com/wsdj/nomoremadness.html
1-877-CYANEA8 | sales@cyanea.com

Securing Access to the WebSphere Application Server Namespace

Define roles to protect the contents of a namespace

BY STEPHEN **PIPES**



IBM's WebSphere Application Server (WAS), v5.1 extends the Java 2 Enterprise Edition (J2EE) role-based authorization concept to protect its CosNaming service, which provides access to a namespace. Client programs typically make use of this service indirectly when using the Java Naming and Directory (JNDI) interfaces or when directly invoking the CosNaming methods.

WebSphere Application Server, v5.1 provides a distributed Interoperable Naming Service (INS) that is federated among all servers in a cell. On each application server, a name service provides the same logical view of the cell-wide namespace.

According to the Sun Microsystems Java Web site, "the CORBA COS (Common Object Services) Naming Service provides a tree-like directory for object references much like a file system provides a directory structure for files" and "the Interoperable Naming Service (INS) is a URL-based naming system on top of the CORBA Naming Service, as well as a common bootstrap mechanism that lets applications share a common initial naming context."

The J2EE platform specification justifies the existence of a naming service by defining the following two requirements:

- The Application Assembler and Deployer should be able to customize an application's business logic without accessing its source code.
- The Application must be able to access resources and external information in its operational environment without knowledge of how the external information is named and organized in that environment.

This article uses the dumpnamespace tool to display the contents of the namespace (at the server root) accessed through the name server running on the specified host and port. The first section of a namespace dump is shown below.

```
Getting the initial context
Getting the starting context

====================================
Namespace Dump
  Provider URL: corbaloc:iiop:localhost:2809
  Context factory: com.ibm.websphere.naming.
WsnInitialContextFactory
  Requested root context: cell
  Starting context: (top)=hurper00
  Formatting rules: jndi
  Time of dump: Sat Jan 24 22:55:28 GMT 2004
====================================


====================================
Beginning of Namespace Dump
====================================

  1 (top)
  2 (top)/cellname                         java.lang.String
  3 (top)/nodes                     javax.naming.Context
  4 (top)/nodes/hurper00            javax.naming.Context
  5 (top)/nodes/hurper00/node       javax.naming.Context
  5    Linked to context: hurper00/nodes/hurper00
  6 (top)/nodes/hurper00/domain     javax.naming.Context
  6    Linked to context: hurper00
  7 (top)/nodes/hurper00/nodename          java.lang.String
  8 (top)/nodes/hurper00/cell       javax.naming.Context
  8    Linked to context: hurper00
  9 (top)/nodes/hurper00/persistent javax.naming.Context
 10 (top)/nodes/hurper00/servers    javax.naming.Context
 11 (top)/nodes/hurper00/servers/server1
                                    javax.naming.Context
 ...
```

The WAS administrative console may be used to add arbitrary values to the namespace. This article assumes the reader is familiar with the operation of the namespace provided by WAS and the administrative console, although certain tasks relating to the name server are covered in detail. For information beyond the scope of this article, refer to the resources section at the end of this article. The reader should also be familiar with the Java programming language.

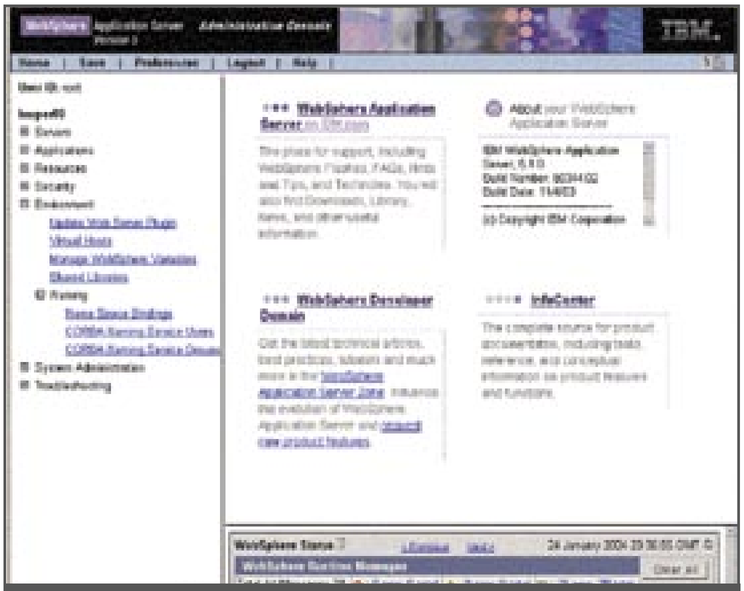
CosNaming Security

Access to the CosNaming service is protected by role-based security. This improves the granularity of control over which CosNaming functions a user may perform. Four roles exist in WAS v5.1:

- ***CosNamingRead:*** Users assigned to this role may query the namespace. This function is provided by the JNDI method lookup. This role is mapped to the group Everyone by default. The predefined groups are discussed later in this article.
- ***CosNamingWrite:*** Users assigned to this role can write to the namespace using the JNDI methods bind, rebind, and unbind, as well as all CosNamingRead operations. This role is mapped to the group AllAuthenticated by default.
- ***CosNamingCreate:*** Users assigned to this role may create new objects in the namespace using the JNDI method createSubcontext, plus all CosNamingWrite operations. This role is mapped to the group AllAuthenticated by default.
- ***CosNamingDelete:*** Users assigned to this role have the ability to destroy objects in the namespace using the JNDI method destroySubcontext, as well as any of the CosNamingCreate operations. This role is mapped to the group AllAuthenticated by default.

These roles are defined in naming-authz.xml, which is a schema document that can be found in the <WAS_HOME>/config/cells/<CELL_NAME> directory. The default content of this file is shown in Listing 1.

The roles are defined with the roleName tag and are assigned an ID that is used elsewhere to refer to these roles. There are three predefined groups, defined by the specialSubjects tag, which are AllAuthenticatedUsers, Everyone, and Server. The first two groups are visible to administrators of the application server using tools such as the administrative console. The third group is reserved and allows the application server, running under the system identity, to perform all CosNaming functions.


WebSphere Application Server Administrative Console
IBM.
Home | Save | Preferences | Logout | Help
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variables
Shared Libraries
Naming
Name Space Bindings
CORBA Naming Service Users
CORBA Naming Service Groups
System Administration
Troubleshooting
WebSphere Application Server
WebSphere Developer Domain
InfoCenter
WebSphere Status
Clear all
FIG 1: COSNAMING SERVICE IN ADMINISTRATIVE CONSOLE

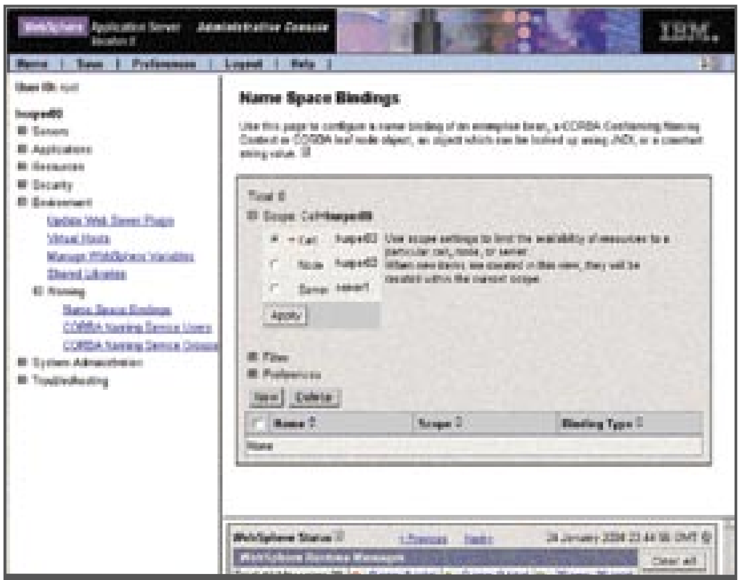


FIG 2: NAME SPACE BINDINGS ADMINISTRATION

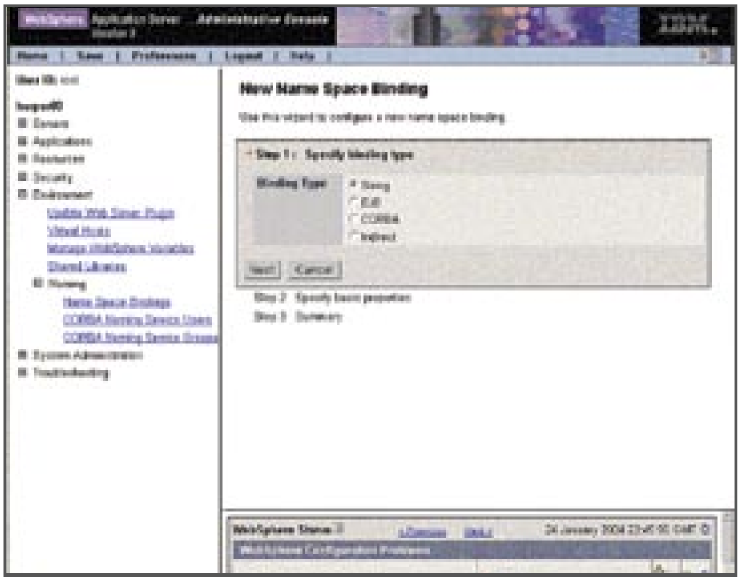
The mappings of users and groups to roles are clear in this file. Each role has zero or more associated users and groups, which are defined by the authorizations tag. For instance, the role CosNamingCreate, which is referred internally as SecurityRoleExt_3, is associated with two groups as defined in RoleAssignmentExt_3. The roles are the special subjects AllAuthenticated and Server.

The administrative console is used to maintain mappings between users, groups, and CosNaming roles. If Global Security is enabled, a valid user name and password will be required to log in. Then, in the left-hand navigation pane, expand the Environment tab and then expand Naming, as shown in Figure 1.

Click CORBA Naming Service Users to display a list of user-to-role mappings. By default, this list is empty. Click CORBA Naming Service Groups to display a list of group-to-role mappings. By default, this list contains the predefined groups All Authenticated and Everyone.

Users and groups, including the special subjects, may be added and removed using the administrative console. It is necessary to restart the application server for the changes to take effect.

The final option is Namespace Bindings, which allows an administrator to add name bindings of an enterprise bean, another CosNaming context, or leaf node


New Name Space Binding

Use this wizard to configure a new name space binding

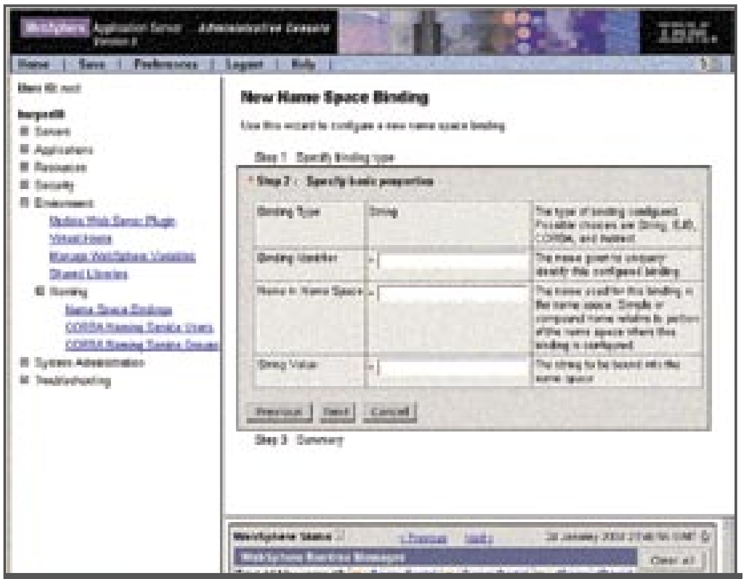
Step 1 : Specify binding type

Binding Type: String, EJB, CORBA, Indirect

Step 2 Specify basic properties

Step 3 Summary

FIG 3: ADDING A NEW NAMESPACE BINDING


New Name Space Binding

Use this wizard to configure a new name space binding

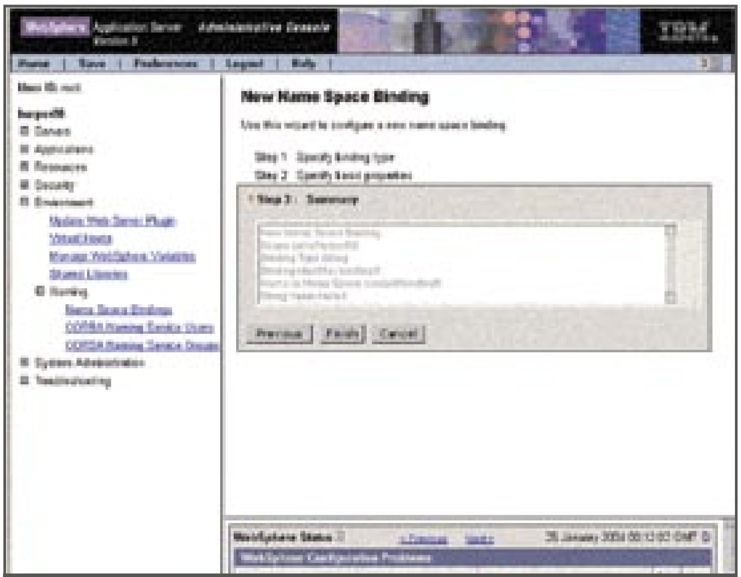
Step 1 Specify binding type

Step 2 : Specify basic properties

Binding Type	String	The type of binding configured. Possible choices are String, EJB, CORBA, and Indirect.
Binding Identifier		The name given to uniquely identify this configured binding.
Name in Name Space		The name used for this binding in the name space. Simple or compound name relative to portion of the name space where this binding is configured.
String Value		The string to be bound into the name space.

Step 3 Summary

FIG 4: NAMESPACE BINDING INFORMATION


New Name Space Binding

Use this wizard to configure a new name space binding

Step 1 Specify binding type

Step 2 Specify basic properties

Step 3 : Summary

FIG 5: NEW BINDING SUMMARY

object – any object that can be referenced through JNDI or a constant string value. The binding's scope can be limited to one of the three following levels:

- ***Cell:*** The binding is created under the cell persistent root context.
- ***Node:*** The binding is created under the node persistent root context for the node.
- ***Server:*** The binding is created under the server persistent root context for the server. If the server is a member of a cluster, this binding will be created under the server root for each server in that cluster.

The advantage of configuring a binding, rather than programmatically using a CosNaming or JNDI application, is that the application server will recreate the binding when it is started and so the binding persists between server restarts. This is achieved because the configured binding is stored in a file called namebindings.xml in the <WAS_HOME>/config/cells/<CELL_NAME> directory. Note, this file may not exist by default – it is created when a new binding is added.

Adding Configured Bindings

To add a persistent binding to the namespace, click Namespace Bindings. The page shown in Figure 2 will appear.

Select the scope for the binding by clicking the appropriate radio button; in this example it will be left as cell, which is the default. Click New, which will display the New Namespace Binding page (see Figure 3).

Select the appropriate type of binding to be added. The example will select String. Click Next. The next page requests information appropriate to the type of binding (see Figure 4).

Enter a unique identifier in the Binding Identifier field, such as bindtest1. Enter a namespace reference in the Name in Namespace field. The binding identifier and namespace identifier can be the same. Intermediate contexts can also be included in this field, for instance, icontext/bindtest1. Enter a string value that will be associated with this binding. This value can be any arbitrary string. Click Next to move to the summary screen (see Figure 5).

Click Finish to create the new binding. If this is successful, the master configuration must be updated to reflect the changes, so click Save to update the box shown in Figure 6 and then Save again to confirm.

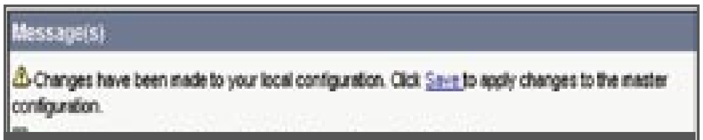
Return to the Namespace Bindings page to confirm that the binding has been correctly saved.

INTERMEDIATE CONTEXTS

The following binding may be defined:

```
cell1/node1/server1/ejbHome
```

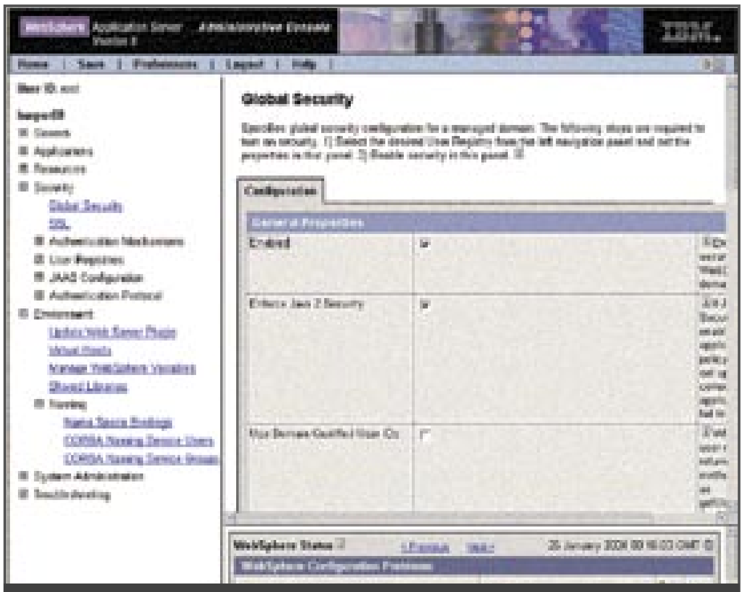
The intermediate contexts, cell1, node1, and server1 are read-only, regardless of the CosNaming role associations and they cannot be modified. However, additional bindings may be added to intermediate contexts and



Message(s)

Changes have been made to your local configuration. Click Save to apply changes to the master configuration.

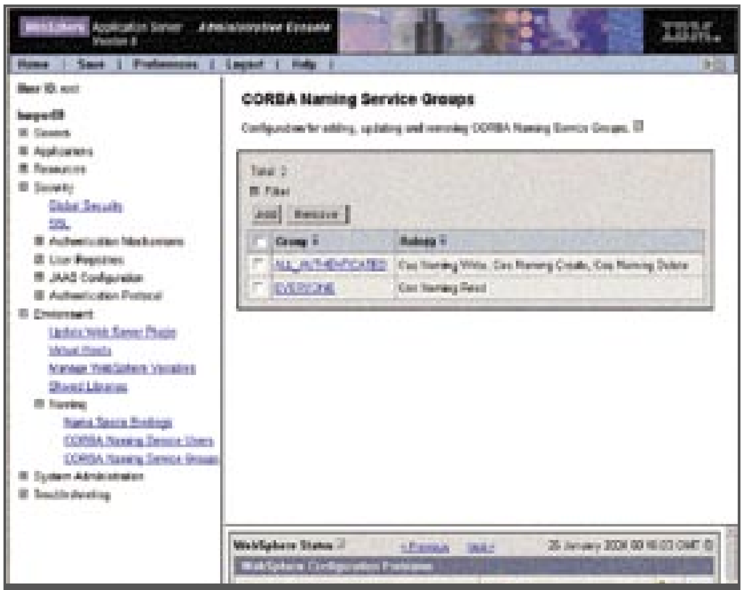
FIG 6: UPDATE THE MASTER CONFIGURATION



Global Security

General Properties	
Enabled	✓
Enforce Java 2 Security	✓
Use Domain Qualified User IDs	

FIG 7: GLOBAL SECURITY SETTINGS



CORBA Naming Service Groups

Group	Roles
ALL_AUTHENTICATED	Cos Naming Write, Cos Naming Create, Cos Naming Delete
EVERYONE	Cos Naming Read

FIG 8: COSNAMING GROUPS

bindings that currently exist may be removed. A user will have appropriate access granted to these bindings, as determined by the CosNaming access-control lists. There is more information on the structure and configuration of namespaces in the WAS Infocenter.

Adding Bindings Programmatically

The namespace may be accessed programmatically for purposes of viewing, adding, and removing bindings. Two methods are commonly used to make calls to the CosNaming service.

The first approach uses a CORBA application and can make calls to the Java 2 CosNaming API, which provides the necessary methods to view and manipulate the namespace.

The second approach is to call the CosNaming service from a JNDI application. A JNDI interface is provided to satisfy the J2EE specification that states J2EE application clients, enterprise beans, and Web components are required to have access to a JNDI naming environment.

A key difference with programmatically binding to the namespace is that any modifications made will not be reflected in the master configuration repository. Only changes made using the administrative console will be reflected.

Sample JNDI Application

The details of how this JNDI client application works is largely beyond the scope of this article and will not be included here. Advanced issues such as JNDI cache management are also not covered here. The full source code for this sample application is provided with this article for the reader to implement as an application client. To run the application client from the command line, use the tool called launchclient, which is provided by WAS. For example,

```
launchclient <EAR_FILENAME> <JNDI_PATH>
```

where <EAR_FILENAME> is the filename of the enterprise archive (including the directory in which it is located) and <JNDI_PATH> is the location of the bound object in the namespace.

The first step for the JNDI client application is to get the default initial context. The initial context returned depends on the type of client. If it is a J2EE application client, the context is the server root context.

Typically, the application client environment is set appropriately and no further action is needed to obtain a default root context. However, if a different root context is required, then the java.naming.provider.url URL property may be used. Refer to the Infocenter documentation for detailed information. For the time being, this property will be set to the default root context (see Listing 2).

Using the initial context object, the client can perform lookups on the CosNaming server. To test this, the following code attempts to locate the binding that was added to the namespace earlier in this article.

```
  private String getMessage(String name) throws
NamingException {
         // Look up the bound String using the JNDI
name
         if (this.ctx == null)
                return null;
         return (String) this.ctx.lookup(name);
  }
```

Because a string was bound, it can be cast back to its original type in this method. Finally, a main method is required before this application can be run with the launchclient tool, as shown in Listing 3.

Because the entire JNDI client application has not been reproduced here, consult the source code file included with this article for the complete listing.

The binding's message should appear on the console when this application is run. To ensure that the binding will survive an application server restart, it will be created within the persistent context. Run the following command on the command line.

```
  launchclient <EAR_FILENAME> cell/persistent/icon-
text/bindtest1
```

The output should look something like this:

```
IBM WebSphere Application Server, Release 5.1
J2EE Application Client Tool
Copyright IBM Corp., 1997-2003
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application Client
Environment.
WSCL0035I: Initialization of the J2EE Application
Client Environment has completed.
WSCL0014I: Invoking the Application Client class
<EAR_FILENAME>
Hello!
```

If problems are encountered accessing the CosNaming, it may be that Global Security is enabled on the server but not on the client. Check the sas.client.props file in the <WAS_HOME>/properties directory and ensure that the property com.ibm.CORBA.securityEnabled equals true. It is necessary to enable Global Security to protect the CosNaming server. The IBM WebSphere v5.0 Security Redbook has some useful information on the subject of Global Security. Ensure that the correct user details are being entered when the application client runs, so the server can authenticate the user successfully.

So far, the JNDI client is simply performing a lookup. As discussed earlier, the application server allows all users of the system to perform this operation. Although access to read from the namespace can be restricted by modifying which operations members of the Everyone can perform, this is not recommended. Any J2EE application installed in an application server may require read access to the namespace. If the default Everyone group was removed from the CosNaming read role, these applications may no longer perform correctly. One symptom of such a problem is the appearance of org.omg.CORBA.NO_PERMISSION exceptions at application runtime. On the other hand, a policy may be defined that insists that all applications formally log on before performing such lookups.

A security policy may, however, consider making changes to a namespace a lot more seriously. The default configuration allows any authenticated user to write

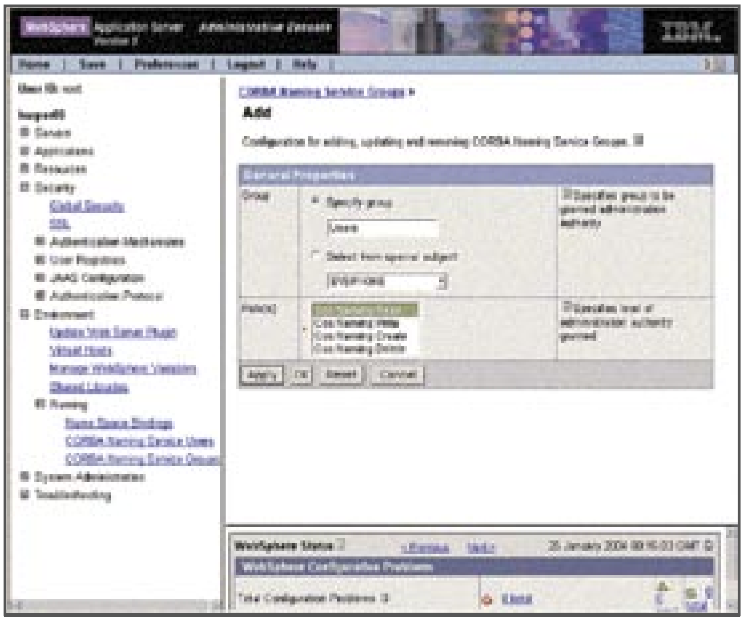

WebSphere Application Server Administrative Console
Home | Save | Preferences | Logout | Help
CORBA Naming Service Groups
Add
General Properties
Apply
OK
Reset
Cancel
WebSphere Status

FIG 9: ADDING A NEW GROUP

to the namespace and modify its structure. This article assumes that such behavior is considered unacceptable and should be restricted, regardless of the number of applications installed on an application server. Only those with identities that are members of the appropriate CosNaming groups will be able to make changes to the namespace.

In fact, there are significantly fewer changes made to the namespace, at least as a member of the AllAuthenticated group, than there are read operations. Testing shows that removing this group from the default CosNaming roles and assigning it to the CosNamingRead role only does not affect the starting of the application server.

The next block of code allows the JNDI application to write some information to the JNDI namespace.

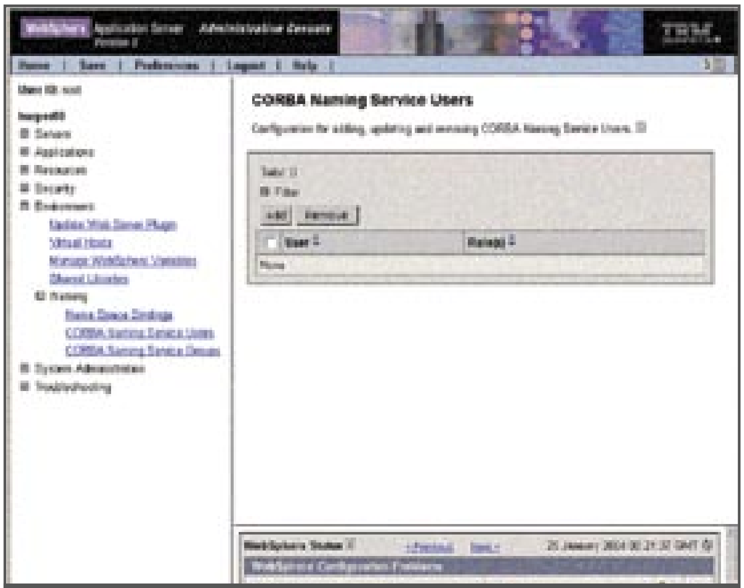
```
  private String bindMessage(String name, String
message)
         throws NamingException {
         // Bind a String to the namespace - inter-
mediates must exist
         if (this.ctx == null)
                 return null;
         this.ctx.bind(name,message);
         return name;
  }
```

To test this code, enter the following on the command line.

```
  launchclient <EAR_FILENAME> cell/persistent/icon-
text/bindtest2 Goodbye!
```

An exception should be thrown stating:

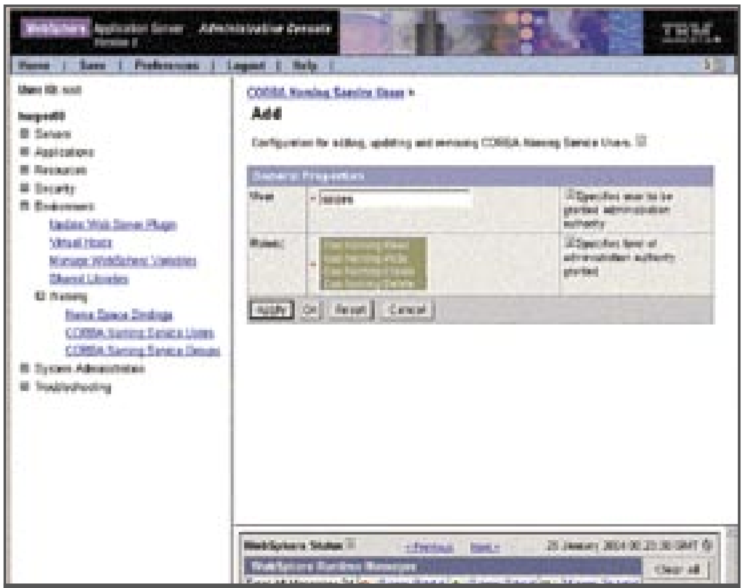
```
  org.omg.CORBA.IMP_LIMIT: Context is Read Only,
Primary Name=<NODE_NAME>/persistent/icontext
```



CORBA Naming Service Users

Configuration for adding, updating and removing CORBA Naming Service Users.

FIG 10: COSNAMING USERS



Add

Configuration for adding, updating and removing CORBA Naming Service Users.

FIG 11: ADDING A NEW USER

The bind has failed because the context is read-only and will not allow additional bindings to be made at this point in the namespace. Intermediate contexts that are created using the administrative console are set to read-only by default. However, it is perfectly acceptable to perform a bind elsewhere in the namespace. Run the following command:

```
  launchclient <EAR_FILENAME> cell/persistent/
bindtest2 Goodbye!
```

The context has changed from the previous attempt and this time the bind should be successful. The message should appear on the command line without exceptions being thrown, as shown below:

```
IBM WebSphere Application Server, Release 5.1
J2EE Application Client Tool
Copyright IBM Corp., 1997-2003
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application
Client Environment.
WSCL0035I: Initialization of the J2EE Application
Client Environment has completed.
WSCL0014I: Invoking the Application Client class
<EAR_FILENAME>
Goodbye!
```

Running the dumpnamespace tool will confirm that this bind has been successful.
The next stage is to modify the CosNaming server's access control lists to ensure that the namespace is protected from unauthorized user access.

Mapping Users and Groups to CosNaming Roles

GLOBAL SECURITY

Global Security must be enabled for the CosNaming server to protect the namespace. There are numerous documents on the subject of enabling and using Global Security, so only the basic process is covered here.

Global Security is typically controlled from the administrative console. In the left-hand pane, expand the Security tab and select Global Security (see Figure 7).

Ensure that the Enabled box is ticked and if any changes have been made, click OK to confirm and then save the changes. It is necessary to restart the application server to enable Global Security.

ADDING A GROUP

It is generally recommended to add groups to the access control lists rather than individual users. This should result in fewer entries in the lists and removes any chance of the lists becoming outdated as individuals are added and removed. Because making changes to the CosNaming server access control lists requires the application server to be restarted, maintaining a list of groups is preferable to a list of users, as fewer changes should be required.

The administrative console is, once again, the primary tool for adding and removing groups. In the left-hand pane, expand the Environment tab and then Naming. Select CORBA Naming Service Groups. The page shown in Figure 8 should appear.

Click Add. A page will appear requesting the name of the new group. In the Specify group text box, enter a group name that exists in the user registry, such as Users (see Figure 9).

Select appropriate CosNaming roles from the Role(s) list (multiple entries can be selected by holding down the CTRL key and clicking). Click OK.

If the group was added successfully, it will appear in the list with the other groups. The changes should be saved to the master configuration in the usual fashion and the application server should be restarted.

ADDING A USER

It may be necessary to grant individuals access to the CosNaming server rather than groups of users. For this reason, it is possible to add users individually. Select the CORBA Naming Service Users link in the left-hand pane of the administrative console (see Figure 10).

Click Add. In the user text box that appears, enter the name of a user in the user registry and select the CosNaming roles that this user will be permitted to perform (see Figure 11).

Click OK to confirm. If the addition was successful, the user will be added to the list. In this case, the changes should be saved to the master configuration and the application server should be restarted.

Note: If a user is assigned to a particular set of roles and the user is also a member of a group that is assigned to a different set of roles, then the user will be granted access to both sets of roles.

Differences Between WAS v4 and v5

The names of the CosNaming roles have not changed from WAS v4. However, how access is granted, by default, differs. In WAS v4.0.2, each of the CosNaming functions mentioned earlier in this article was mapped to a single role.

Taking WAS v4 as an example, assume that a user has been assigned the role CosNamingCreate. It would not be possible for that user to perform a CosNaming lookup because the role CosNamingRead had not been assigned. However, in WAS v5, the CosNamingCreate includes both CosNamingRead and CosNamingWrite roles and in this case, the user would be able to perform a lookup successfully.

STATUS MESSAGES

The WAS documentation refers to several messages that the reader may find useful:

NMSV0730I: Name server registering as a Security Service listener.
Explanation: This is an informational message logged during name server startup indicating that the name server is registering as a Security service listener. This means that name server security will be enforced.
User Response: This message is informational only. No action is required.

NMSV0731I: Security Service not available. Name server cannot register as a Security Service listener.
Explanation: This is an informational message logged during name server startup indicating that the Naming component is not registering as a Security service listener. This means that name server security will not be enforced.
User Response: This message is informational only. No action is required.

SECJ0139E: Error to get initial naming context
Explanation: This is an unexpected exception. The cause can not be immediately determined.
User Response: For further information on resolving this error, please consult the IBM WAS Support Web site available at: http://www-3.ibm.com/software/webservers/appserv/support.html. The site provides searchable databases of technotes, solutions, and e-fixes. Information on contacting the WebSphere Support team is also provided.

"Global Security must be enabled for the CosNaming server to protect the namespace"

Conclusion

The CosNaming service provides a convenient namespace mechanism for J2EE applications to programmatically view and alter. WAS v5.1 provides two types of interfaces for easy manipulation of the namespace and to ensure compatibility with the J2EE 1.3 specification.

Administrators are granted full rights to make appropriate changes to the namespace using the administrative console. However, access to non-administrative users may be restricted through the use of CosNaming roles and Global Security. Roles may be assigned on a group basis or on an individual user basis. Maintenance of these roles has been improved from WAS v4, making administration easier and less prone to error.

Resources

- *J2EE 1.3 specification (see chapter 5)*: http://java.sun.com/j2ee/1.3/download.html#platformspec
- *CORBA FAQ:* www.omg.org/gettingstarted/corbafaq.htm
- *IBM WebSphere v5.0 Security Redbook (PDF document):* www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246573.pdf

Stephen Pipes is an IT specialist at IBM's Hursley Park Labs in the UK, where he is a member of the Pervasive Computing group, providing consultancy services for customers in Europe, the Middle East, and Africa. Stephen is an expert in WebSphere Application Server security and regularly presents at technical conferences on the subjects of pervasive computing, security, and Java.
pipessd@uk.ibm.com

LISTING 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:rolebasedauthz="http://www.ibm.com/websphere/appserver/
schemas/5.0/rolebasedauthz.xmi">
  <xmi:Documentation>
    <contact>WebSphere Security</contact>
  </xmi:Documentation>
  <rolebasedauthz:AuthorizationTableExt xmi:id="Authorization
TableExt_1" context="domain">
    <authorizations xmi:id="RoleAssignmentExt_1"
role="SecurityRoleExt_1">
  <specialSubjects xmi:type="rolebasedauthz:EveryoneExt" xmi:
id="EveryoneExt_1"/>
        <specialSubjects xmi:type="rolebasedauthz:ServerExt"
xmi:id="ServerExt_5"/>
    </authorizations>
    <authorizations xmi:id="RoleAssignmentExt_2"
role="SecurityRoleExt_2">
  <specialSubjects xmi:type="rolebasedauthz:AllAuthenticatedU
sersExt" xmi:id="AllAuthenticatedUsersExt_2"/>
        <specialSubjects xmi:type="rolebasedauthz:ServerExt"
xmi:id="ServerExt_6"/>
    </authorizations>
    <authorizations xmi:id="RoleAssignmentExt_3"
role="SecurityRoleExt_3">
  <specialSubjects xmi:type="rolebasedauthz:AllAuthenticatedU
sersExt" xmi:id="AllAuthenticatedUsersExt_3"/>
        <specialSubjects xmi:type="rolebasedauthz:ServerExt"
xmi:id="ServerExt_7"/>
    </authorizations>
    <authorizations xmi:id="RoleAssignmentExt_4"
role="SecurityRoleExt_4">
  <specialSubjects xmi:type="rolebasedauthz:AllAuthenticatedU
sersExt" xmi:id="AllAuthenticatedUsersExt_4"/>
        <specialSubjects xmi:type="rolebasedauthz:ServerExt"
xmi:id="ServerExt_8"/>
    </authorizations>
    <roles xmi:id="SecurityRoleExt_1" roleName="CosNamingRe
ad"/>
    <roles xmi:id="SecurityRoleExt_2" roleName="CosNamingWr
ite"/>
    <roles xmi:id="SecurityRoleExt_3" roleName="CosNamingCr
eate"/>
    <roles xmi:id="SecurityRoleExt_4" roleName="CosNamingDe
lete"/>
  </rolebasedauthz:AuthorizationTableExt>
</xmi:XMI>
```

LISTING 2

```
private boolean getCtx(Hashtable h) throws NamingException {
          this.ctx = new InitialContext(h);
          return true;
  }

  private Hashtable getEnv() {
          Hashtable env = new Hashtable();
          env.put(
                    Context.INITIAL_CONTEXT_FACTORY,
                    "com.ibm.websphere.naming.WsnInitialConte
xtFactory");
          env.put(Context.PROVIDER_URL, "corbaloc:iiop:local-
host:2809");
          return env;
  }
```

LISTING 3

```
public static void main(String[] args) {
          if (args.length == 0)
                    System.out.println("Enter a JNDI path.");
          else {
                    try {
                              JNDIClient jc = new
JNDIClient();
                              Hashtable h = jc.getEnv();
                              if (jc.getCtx(h)) {
                                        // context is set
correctly
                                        if (args.length == 2)

jc.bindMessage(args[0], args[1]);
                                        System.out.
println(jc.getMessage(args[0]));
                              }
                    } catch (Exception e) {
                              System.out.println(
                                        "Exception thrown in
application client. Exiting process.");
                              e.printStackTrace();
                    }
          }
          System.exit(0);
  }
```

Interoperating between disparate platforms

Best Practices in Java Database Connectivity

BY DAVID MURPHY

The Java Database Connectivity (JDBC) specification was created to bridge application interoperability gaps by enabling programs written in Java to access information in any type of database. To build an interoperable environment that meets the needs of users in terms of data accessibility and interoperability, developers must choose each piece of the JDBC solution carefully, while keeping in mind the full data environment – from database platform to end user data needs.

Database drivers are an important piece in the JDBC puzzle because they contain the special knowledge of the device or special software interface that is not visible to typical application code. With numerous driver choices on the market, it is important to remember that not all database drivers are created equal. Price, while tempting with free drivers on the market, should not be the determining factor in choosing your driver. Enterprises must closely examine needs in terms of data volume processed, security, and interoperability, as well as speed of queries and updates. This article outlines the areas to examine and questions to ask in determining which driver will best meet your needs.

Driver Type

First, you must determine the type of driver your application will require. This means looking at the environment, both in a technical and business sense, in which the data will be shared and transmitted. If users place a high priority on speed – for example, if you are running a transaction system with a large number of concurrent users – a native driver would best serve the need for performance. If the security of data is a prime concern – as in a distributed financial application – a server-oriented, “type three” driver may be the best bet to ensure secure data transmission.

Once the type is determined, you need to take a close look at the driver features to ensure your choice will support your applications.

Keeping Up with the Standards

It would seem that, in the category of standards and certification, it would be a wash across all driver choices – wouldn’t all JDBC drivers be certified on the latest standards? Not necessarily. First, look for a commitment to the Java platform. A stated commitment shows that support for Java will continue through updates and through time, eliminating the uncertainty of a sudden cutoff of Java support.

Next, look for Sun J2EE certification to guarantee that the product will interoperate with other J2EE-certified products like IBM’s WebSphere Applicaton Server. Certification alone does not necessarily mean the product is up-to-date in terms of the latest standards. The JDBC driver should be compliant with the latest standard, JDBC 3.0. This compliance shows that the vendor stands behind the Sun JDBC specification and will continue to release new versions as the JDBC specification evolves – for example, support for the forthcoming JDBC 4.0 specification. JDBC 3.0 compliance means the driver will provide:

- Standard row key retrieval
- Savepoints
- Holdable cursors
- Parameter metadata
- Named parameters
- Statement pooling

Without JDBC 3.0, applications have to revert to proprietary techniques to retrieve new row keys, which hinders the application from running on different database platforms and causes vendor lock-in.

Security

Depending on your security infrastructure and the application’s security requirement, you may need to investigate the driver’s security abilities. Drivers with data encryption and firewall tunneling enable JDBC connectivity and data sharing within secure wide and local area networks.

Another security concern is trusted authentication, which is important in environments that include a large number of clients where database security administration needs to be centralized. Effective JDBC security solutions will appear seamless to the end user as part of a smooth data exchange process.

Scalability

The best JDBC drivers can be installed and can begin running without changing any lines of the application's code. This will save time and money because your application benefits from all the advantages of a high-quality JDBC driver by simply deploying it with the application.

To determine how well your application will scale to many users once it is deployed, you should have a sense of any capacity limits the driver imposes. Take a look at the vendor's stated limit and capacity for:

- Multiple statements per connection
- Multiple result sets per connection
- Multiple statements per transaction

Limits on these characteristics can severely impact the performance and scalability of your application. For example, a driver may require new connections to support additional statements, which causes major scalability problems for data intensive transactional applications.

Reputation and Support

Similar to hiring a new DBA or developer, you need to check references on the "candidate" for your JDBC driver. Look for reviews in your favorite technology publication or on Web sites. Post questions on bulletin boards and list servs you frequent. Inquire as to the product's reputation for performance, scalability, and reliability.

Look into the product's stated support policy. More than just a promise of immediate support, the vendor should be able to promise that their support representatives are engineers, qualified and trained in Java applications and JDBC.

In-house or OEM?

Will the end application be used in-house or be part of your company's OEM offerings? If it will be OEM-ed, the driver should be able to be bundled and installed with your OEM applications. Bundling benefits both you as the vendor as well as the customer by allowing preconfiguration of an application before it is installed on customer site.

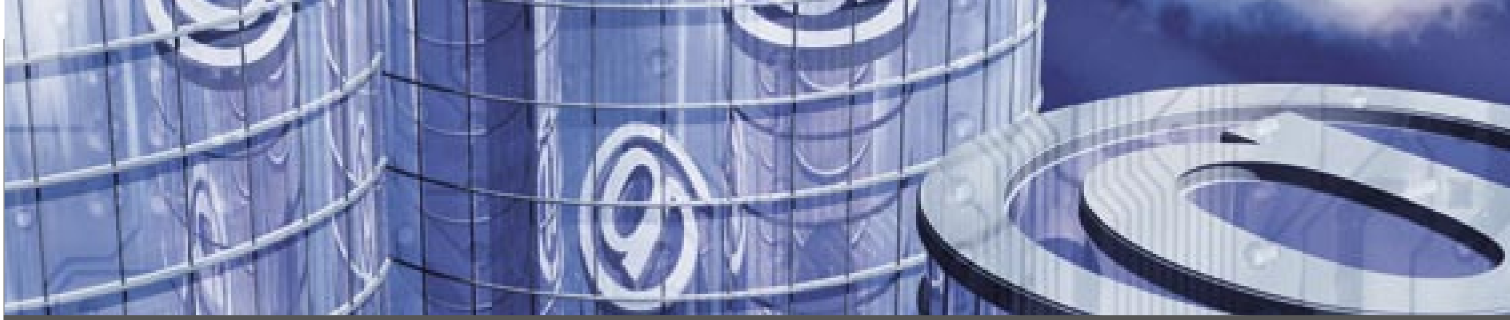
In an OEM environment, the driver vendor's redistribution policies are of crucial importance for determining/maintaining pricing. Scalability issues, as discussed above, should also be taken into account in an OEM situation – does the driver's scalability match your application's promise of scalability?

Conclusion

JDBC driver selection should be approached with the same consideration as database selection. In a transaction-oriented solution, the performance characteristics of the database driver can have a crippling effect on application stability. A relatively small investment in a high-performance database driver can bring significant savings by improving the performance and scalability of applications and reducing the need to expend financial resources for hardware improvements.

The adage "a chain is only as strong as its weakest link" is particularly appropriate to the connectivity issues regarding JDBC driver selection. An enterprise-capable driver, selected with consideration for the issues raised in this article, may possibly be the missing link to the ultimate performance and scalability of your Java applications.

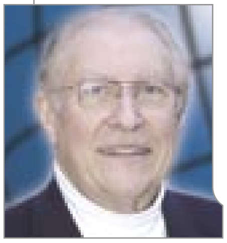
David Murphy is the executive vice president of corporate strategy at JNETDirect.
David.Murphy@jnetdirect.com



IBM's WebSphere Application Server vs. Microsoft's .NET Framework

Part 1

BY DWIGHT PELTZER



The data evolution has completely changed how businesses conduct transactions on the Web. The business environment has evolved rapidly from presenting static data to consumers and businesses alike, to a new world, something we can only describe as the era of application integration and data evolution.

Interoperability, today's buzz word, is meaningful because it describes a process in which legacy systems are being integrated with new, interactive business software solutions. Both IBM's WSAD and Microsoft's.NET Framework v1.1 are component based and contain a large collection of interfaces.

This article focuses on the internal structure of .NET, whereas the second article will examine WSAD's layered architecture. The remaining articles will demonstrate how both infrastructures interact with their respective Web components, i.e., ASP.NET, ADO.NET, Java servlets, JavaServer Pages, and Enterprise JavaBeans.

Web Services Provide the Glue that Binds Disparate Systems

Web services represent the main force driving the Internet today. Recently, Sun Microsystems decided to provide support for Web services. Their release of the .NET Framework represents the company's attempt to enter the enterprise distributed application arena.

The Framework consists of several components, all supporting Web services. WebSphere Application Server's architecture also supports Web services, something we'll discuss in the next article.

A dizzying array of integrated business solutions is available on both platforms. Learning new technologies is a major challenge facing developers who are accustomed to building enterprise applications exclusively in one programming language or another, but not both. Rarely does a developer know both platforms, especially since Microsoft .NET technology is relatively new.

Understanding .NET's Web Services and Functionality

The Internet has leveled the business playing field. Small companies can appear to be much larger than they are. Large corporations can cut costs significantly by conducting business transactions on the Web.

So, how does the .NET Framework and its components meet the needs of consumers today? The same question can be asked of IBM's WSAD container and its diverse technologies. The answer to both questions requires detailed explanations.

Let's begin by exploring the .NET Framework and its layered architecture, followed by an examination of IBM's WSAD and how it supports Web services. Cross-platform interoperability is the key to providing insights on how to achieve application integration.

.NET's Infrastructure: Component Based

The .NET platform consists of several components:

- Common Language Runtime (CLR)
- Common Type Specification (CTS)
- Common Language Specification (CLS)
- Web Forms
- ASP.NET
- ADO.NET

At the lowest layer lies the operating system, which can be a variety of Windows platforms, ranging from Windows XP to Windows Server 2003. Residing on top of the OS is a series of Enterprise Server products such as Application Center 2000, BizTalk Server 2000, Host Integration Server 2000, and SQL Server 2000. The top layer comprises the Web services layer, namely the .NET Framework v1.1. The Framework is a new development and runtime structure that includes the CLR, the CTS, the CLS, and a common set of libraries and namespaces providing language independence and multi-language interoperability.

You can think of the CLR, the most important component of the .NET Framework, as the equivalent of Java's JVM. The main difference is that the JVM supports only the Java language, while the CLR supports all languages targeting the .NET Framework. This interoperability is provided courtesy of the Common Intermediate Language (CIL). Java's JVM executes bytecode by interpreting it, so it can support many different languages. However, unlike bytecode, the Microsoft Intermediate Language (MSIL) is compiled rather than interpreted.

The layer residing on top of the CLR contains a set of classes that support fundamental input and output functionality, network communications, thread management, text management, reflection, collections functionality, and security management.

On top of the Framework base classes is a collection of classes that support data management and XML functionality. They also include the Structured Query Language (SQL) interface to SQL Server 2000. .NET Component ADO.NET allows you to manipulate persistent data in a disconnected mode. We will talk more about this later.

The three Web-oriented technologies – Web services, Web Forms, and Windows Forms – extend the Framework. Web services play a major role in the development of distributed components. Web Forms include a set of classes that allow the developer to create Graphical User Interface (GUI) applications, which provide a drag-and-drop experience similar to building GUIs in Visual Basic.NET. You can drag a control onto the Web form, double-click on the control, and respond to the related event. Windows Forms offer a supporting collection of classes that facilitate development of native Windows GUI applications.

Understanding CLR Management's Functionality

The CLR, similar to WSAD, provides low-level services such as automatic memory management, garbage collection, connection pooling, and security. This means you can inherit from classes, catch exceptions, and support object-oriented polymorphisms across numerous programming languages and platforms.

The Common Type Specification facilitates the common concept of interfaces, classes, reference types, value types, and delegates that support callbacks. The Framework base classes offer support for the base system types such as integers, strings, and file manipulation. Last but not least, the Framework designates Simple Object Access Protocol (SOAP) as its primary method of transport using the ubiquitous HTTP protocol.

Exploring the CLR's Environment

The CLR manages all aspects of code execution within the Framework. At this stage, we introduce two key terms, the Manifest and the Assembly. The Manifest provides metadata about the Assembly, including a list of externally referenced assemblies required for application execution. The CLR utilizes a tool included within Visual Studio.NET called ildasm.exe to display the Manifest and the Assembly. Consider the code shown in Listing 1.

The Assembly is the basic unit containing its name, version, and culture. A public key provides assembly verification. This permits assemblies to reside side-by-side without conflict. Significantly, .NET no longer registers DLLs in the system registry, thereby eliminating a lot of problems we have all experienced with DLL versioning. The CLR examines the assembly for its version number. All assemblies shared by more than one assembly must be constructed with a public/private key. At build time, the compiler generates a hash of the assembly, then signs the hash with a private key and subsequently stores the digital signature in a specified section of the Program Executable file (PE). We will examine the PE file in a moment. The CLR verifies the assembly's public key by decrypting the assembly's digital signature. Finally, the CLR uses the assembly's information to generate a hash and compare it with the original hash. If the comparison is valid, the assembly is loaded by the Class Loader, another important CLR component. Its sole task is locating and loading .NET classes using Reflection. However, before the loader loads the assembly, the CLR examines the PE file for application validity.

"Web services represent the main force driving the Internet today"

Examining the Program Executable File (PE)

Let's examine a PE file using a tool called dumpbin.exe, also included within Visual Studio.NET. The command line format is dumpbin.exe filename; its type is either .exe or .dll. (see Listing 2).

The PE file lists the MS-DOS and Common Object File Format (COFF) headers. The overall file structure is the same for all Windows files. Microsoft has extended the PE file by adding headers to accommodate the Framework. Notice that it supports 32-bit Windows programs. The FILE HEADER values segment indicates there are three sections in this file. The SECTION HEADER #1 stores the CLR header and data. The next segment, Code and Execute Read, informs the OS Loader the file contains code to be executed by the CLR at runtime. The rest of the PE file holds individual segments for .rdata, .rscr, and .text, as demonstrated above in the header file. When the CLR locates the header, it executes 0_-CorDLLMain.

The CLR header and data section include both metadata and MSIL code. These describe in detail how CustomerClient executes. MSIL code is similar to Java's bytecode, but MSIL is compiled rather than interpreted. If the file is an executable, it executes o_CorEXEMain.

The PE file hosts the application code whereas the Manifest (metadata about the assembly) describes all references to external libraries, methods, classes, and types in binary format.

Examining the CLR Core Entities

The CLR contains two core entities: runtime engine mscoree.dll and base class library mscorlib.dll. Notice how the binary example listed above begins with assembly extern mscorlib.dll. Since the Assembly was generated in Visual Basic, the manifest reflects this as well. All assemblies have a public key token indicating that the assembly can be shared by other assemblies. We notice that the manifest references other external assemblies such as System, System.Data, System.xml, System.CLSCompliantAttribute, and System.Reflection. Reflection allows you to examine MSIL Code and assemblies. It also supports dynamic creation of assemblies. System.Reflection facilitates late binding and permits you to load the assemblies at runtime.

The second section of the Manifest enumerates each module within the assembly:

```
.custom instance void [mscorlib] System.Reflection.
AssemblyDescriptionAttribute::.ctor (String) = ( 01
00 00 00 00 ).
```

Two kinds of assemblies exist: static assemblies and dynamic assemblies. Static assemblies are stored somewhere on your hard drive, while a dynamic assembly is generated by System.Reflection.Emit. This namespace creates the dynamic assembly in memory at runtime. Once the assembly exists, it reverts back to a static status. Furthermore, it is possible to add new types dynamically to the runtime assembly. For example, Web-enabled languages generate raw code, emit IL code, and store it dynamically to the assembly.

The CLR can generate both single- and multi-file assemblies. The first type represents a single binary. The second contains more than one binary, called a module. Modules allow you to partition a multi-file assembly, thereby facilitating the creation of an easy deployment model.

Managed code is the term used to describe instances of objects you create managed by the runtime. The container holding the IL code is the assembly.

Understanding the Verification Process

Before running MSIL code, you must convert it utilizing the just-in-time (JIT) compiler, called affectionately the Jitter. MSIL is CPU-specific code that runs on the same computer as the JIT compiler. The CLR provides a JIT compiler for each CPU-specific architecture.

The JIT compiles only code required for execution rather than loading all MSIL code existing in the PE file. Once the code is loaded, JIT stores the native code for subsequent calls. The loader creates a stub and attaches it to each of a type's methods. With the initial call, the stub relinquishes control to the JIT compiler. The compiler then modifies the stub to provide CPU-specific execution instructions. Subsequent calls go directly to the identified native code rather than repeating the process just described here. This substantially reduces the time required to run native code.

Another mode of compilation is called install-time code generation. This compilation method converts the MSIL code the same way as the standard mode previously described above. However, the JIT converts larger chunks of code at compile time and stores the native code with consideration for what it already knows about other already installed assemblies. This file loads and starts more quickly than the standard JIT method.

The verification process validates the code before the JIT compiles the MSIL code. It employs reflection to examine both the metadata and MSIL to ensure the code is type safe. The compiler subsequently accesses only the portions of memory containing the type-safe code. Additionally, the verification process guarantees and enforces security restrictions defined by the CLS.

The CLR depends on the following criteria to ensure it is processing type safe code:

- Identities must be validated.
- Only type-safe operations are invoked on an object.

If the JIT receives nonverified code, it generates an exception.

Conclusion

Now that we have explored the .NET Framework, my next article will examine IBM's WSAD infrastructure. We will expose similarities and differences between the two technologies. Once that task is accomplished, we can discuss how both the .NET's CLR and WSAD manage their respective technologies.

Dwight Peltzer is a professor of computer science, developer, lecturer, and author on Microsoft and J2EE software solutions. Presently, he is a faculty member at the School of Engineering and Computer Science at the C.W. Post campus of Long Island University. He teaches seminars on C++, Visual Basic, .C#, .NET, SQL, ASP.NET, and J2EE-related technologies such as JAXP, JavaServer Pages, and XML.
dpeltzer@optonline.net

LISTING 1: A MANIFEST FRAGMENT

```
.assembly extern mscorlib
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
            // .z\V.4..
  .ver 1:0:3300:0
}
.assembly extern Microsoft.VisualBasic
{
  .publickeytoken = (B0 3F 5F 7F 11 D5 0A 3A )
            // .?_....:
  .ver 7:0:3300:0
}
.assembly extern System
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
            // .z\V.4..
  .ver 1:0:3300:0
}
.assembly extern System.Data
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
            // .z\V.4..
  .ver 1:0:3300:0
}
.assembly extern System.XML
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
            // .z\V.4..
  .ver 1:0:3300:0
}
.assembly ViewAssembly
  .custom instance void [mscorlib] System.
CLSCompliantAttribute::.ctor(bool) = ( 01 00 01 00 00 )
  .custom instance void [mscorlib] System.Reflection.Assembly
ProductAttribute::.ctor(string) = ( 01 00 00 00 00 )
::.ctor(string) = ( 01 00 00 00 00 )
  .hash algorithm 0x00008004
  .ver 1:0:907:22791
}
.module ViewAssembly.exe
// MVID: {6E7682F1-0E71-4169-A9D9-2CC80CBD0545}
.imagebase 0x11000000
.subsystem 0x00000003
.file alignment 511
.corflags 0x00000001
// Image base: 0x03190000
```

LISTING 2: THE PE FILE

```
Dumbpin.exe Customerclient.dll / all

Microsoft <R> COFF/PE Dumper Version 7.10 3077
Copyright <C> Microsoft Corporation. All rights reserved.
 Dump of file Customer.dll
PE signature found
File Type: DLL
File Header values [MS-DOS/COFF HEADERS]
14C machine (x86)
OPTIONAL HEADER VALUES
 10B magic #(PE32)
 Section Header #1  [SECTION DATA]
      …
 .text name
 Code  Execute Read
RAW DATA #1
      …
clr Header
//Section contains the following imports:
 Mscoree.dll
 402000 Import Address Table
 4025F8 Import Name Table
      …
 0_CorDLLMain
```

Excerpts from Chapter 4: Using the JMS API

Enterprise Messaging Using JMS and IBM WebSphere

BY KAREEM **YUSUF**, PHD

Having established an understanding of our unit of exchange – the message – we are now ready to delve into the mechanics of messaging using the JMS API. In this chapter we begin with a discussion on implementation choices, examining our options regarding the software components with which we might implement a JMS client. We then undertake a detailed review of the JMS API, using code snippets to illustrate usage and implementation approaches. We do this by examining in turn each set of interfaces: point-to-point, publish-subscribe, and the unified or common interface. This will round out our knowledge of JMS and prepare us to consider a number of real-world scenarios based on a specific product family: IBM WebSphere.

ABOUT THE BOOK
Enterprise Messaging Using JMS and IBM WebSphere
by Kareem Yusuf
ISBN: 0131468634
Publisher: Prentice Hall Professional Technical Reference
Reprinted with the permission of Prentice Hall Professional Technical Reference, in alliance with IBM Press, copyright 2004

The JMS client is the entity that utilizes the JMS API to interact with the JMS provider. It can be any Java artifact, and depending on the application environment, a number of choices exist as to how the JMS client is implemented. The JMS client could be a standalone Java application, serving as part of a desktop client, or acting as the connectivity module (adapter) for a business application. The JMS client could also be encapsulated in a J2EE component such as an applet, servlet or portlet, or Enterprise JavaBean (EJB). That the JMS client can be any Java artifact offers a great deal of flexibility in implementation. However, this comes at a price, as the decision-making process regarding your implementation choice becomes more involved.

The choice between a standalone Java application or J2EE component is one we need not address, as that decision is driven more by the nature of the application and its target runtime environment than by the JMS connectivity requirement. However, if the application is J2EE-based, then we still have the basic choices of applet, servlet, or EJB.

Servlets and EJBs are server-side components that run in containers in the J2EE server. They consequently have access to a host of services, such as security and transaction support (see Chapter 2, "Java Message Service"). Applets, on the other hand, are client-side (presentation layer) components that run in a browser and are most often used as user interface constructs. Current best practice recommends that J2EE applications adopt a lightweight presentation layer with business logic and connectivity to enterprise resources (such as messaging providers) residing in server-side components. This enables transaction and security services to be readily invoked if required and offers an important advantage in that it is much easier to scale and modify serverside components. Furthermore, it greatly simplifies the implementation of the presentation layer.

With this in mind, it is fair to say that applets are probably least suited to the role of JMS client. Besides the design considerations, there are practical implications to be considered as well. The applet needs access to the JMS libraries (standard and provider implementation), increasing its size and potentially impacting the time it takes it to download to the browser. In addition, the still existing inconsistencies in browser Java Virtual Machines (JVMs) and their support for running JMS must be addressed. Such inconsistencies can complicate the deploy-ment of the solution and force you to have to specify browser versions or runtime plug-ins that will support your application. In solutions where applets are used (their current popularity is questionable), a better pattern is for the applet to call a servlet. The servlet then either calls an EJB, which implements the JMS client, or it implements the JMS client itself.

In deciding whether to implement the JMS client in a servlet/portlet or EJB, consider the following key questions:

- Does your programming model currently use EJBs?
- Is your application Web based?
- Based on your application design, where does business logic reside?
- Is the interaction with the JMS provider going to be transactional, potentially involving other resources?
- If it is transactional, do you want to use container managed or client-demarcated transactions?
- What is your considered approach to scaling your solution, particularly access to the JMS provider?
- Which interaction patterns will your JMS client implement?

J2EE best practice recommends that connectivity logic – that is, access to enterprise resources – reside in the business logic layer, which in J2EE is the EJB layer. This suggests that EJBs should be the first choice when implementing the JMS client. However, for Web-based applications, it is not uncommon to find the JMS client being implemented by a servlet or portlet, since they offer a simpler programming framework when compared to EJBs. If transactions are required, then the only choice open to the servlet developer is the use of client-demarcated transactions. This requires that the developer explicitly handle the transactions, making appropriate Java Transaction API (JTA) calls such as commit or rollback at the right time. In contrast, the EJB developer has the option of having transactions managed by the container, which simply involves specifying the scope of the transaction as part of the deployment properties and requires no explicit programming. This greatly simplifies building transactional components and is considered a best practice.

In comparison to servlets, EJBs offer considerable tuning options and efficiency savings in terms of how the EJB container manages its EJBs and, by extension, access to enterprise resources. The EJB container can be expected to provide pooling optimizations, load-balancing configurations, and other scalability and performance enhancements, which can have consider-able benefits in terms of the scalability and reliability of the implemented solution. In addition, access to EJBs often tends to be more secure.

Recall from our discussion in Chapter 1, "Enterprise Messaging," that the interaction between the JMS client and the provider is composed of three base patterns: message producer, message consumer, and request-reply.

The message producer pattern involves simply sending a message; interaction with the provider ends once the message has been accepted by the provider. The session EJB readily lends itself to the role of message producer, being for the most part a stateless bean that can be reused by client applications such as servlets for sending messages. The same can be said for servlets, and here the choice is influenced by additional factors previously discussed, such as transaction requirements. For example, if the sending of the message is to occur as part of a global transaction, such as updating a database within the same transaction, then the container-managed transaction services offered by the EJB container further facilitate the implementation of this pattern.

The message consumer pattern is initiated by the arrival of a message. As discussed in Chapter 1, this initiation could occur in either a pull or push mode. In the pull mode, the application checks (polls) the messaging provider at suitable time intervals for a message. In the push mode, the messaging provider invokes the application when a message arrives, passing it the message. EJB 2.0 (included in J2EE 1.3) introduced support for implementing the message consumer pattern using EJBs by defining the message-driven bean (MDB). MDB support implements the message consumer pattern in push mode. The EJB container monitors the JMS destination using Application Server Facilities, or ASF (see Chapter 2) and invokes the MDB with a retrieved message as messages arrive.

In support of this facility, JMS defines a MessageListener interface, which the MDB implements. The MessageListener interface (discussed in the next section) was available before the advent of MDBs and can be used by servlets or other Java entities to implement the message consumer pattern. When the MessageListener interface is used by components other than MDBs, the JMS provider monitors the destination and invokes the component with the message on arrival. The use of MDBs, however, offers a number of distinct advantages. Using ASF, the EJB container can process multiple incoming messages from multiple destinations concurrently, providing a significant performance benefit. In contrast, the JMS provider can provide only serialized access to messages and destinations for other components using the MessageListener interface. In addition, the EJB container provides tuning, transaction, and message redelivery configuration options, which would have to be explicitly programmed into non-MDB entities.

It is useful to remember, however, that prior to J2EE 1.3 and EJB 2.0, none of this support existed, and the only option available to implement a push-mode message consumer was the MessageListener interface. The push-mode (some might say asynchronous) nature of the MessageListener interface does not readily lend itself to implementation using session EJBs, and thus prior to EJB 2.0, it was particularly difficult to implement the message consumer pattern using EJBs. A common workaround at the time was to have another component, typically a standalone application or servlet, monitor the JMS destination for a message and either retrieve and pass the message to the session bean or notify the session bean that a message was available for retrieval (used if retrieval needed to occur as part of a transaction managed by the EJB container).

The request-reply pattern combines the message producer and message consumer patterns to implement a conversation between the sending and receiving applications. There are two basic variations for this pattern (see Chapter 1). In the first, the requester (acting as a message producer) sends the request, then reverts to the role of message consumer and waits for its reply (pull mode). In the second variation, the requester sends the request, but a different component receives (consumes) the replies, usually operating in a push mode.

With the first variation, a critical factor is the length of time that the requester will wait for a reply to its request. JMS provides a receive method (more on this later in the chapter) that can have a wait time specified. The method returns with a message if it is available or with nothing after the specified wait time if no message has arrived. During the wait interval, the calling thread is blocked, and this is of particular importance if a session EJB is used to implement this pattern. Certain schools of thought totally abhor the concept of a blocked thread within the EJB container, particularly when it is considered that a large number of instances (blocked threads) of the session EJB can potentially exist. Consequently, when a session EJB is used in this manner, careful consideration must be given to the amount of time (specified in milliseconds) the session EJB will be allowed to wait and the impact that may have on the J2EE Server environment.

The second variation avoids this issue, as the requester session EJB doesn't wait for the reply and thus never blocks. The MDB readily lends itself to acting as the consumer of replies and, when paired with the requesting session EJB, provides a straightforward implementation for this variation.

We have detailed a number of considerations that should influence our choice of whether to use servlets or EJBs for JMS client implementations. All things considered, I recommend that JMS clients be implemented using EJBs. On the other hand, in cases where EJBs are not part of the adopted programming model and transactions (particularly global transactions involving other resources) are not required, servlets may prove attractive. However, as with all design decisions, the choice of what software component to use, is further constrained by factors directly associated with the nature of the project. Consequently, it is important to note that regardless of the entity adopted to implement the JMS client, the JMS API is used in the same way, and we examine its use in the following sections.

Point-to-Point Interface

As discussed in Chapter 1, point-to-point messaging is adopted when there is a one-to-one relationship between sender and receiver. In this message distribution pattern, the sender sends the message to a known destination, from where it is retrieved. Common usage scenarios include submitting an order or registration request to a processing application, effecting an account inquiry or update, and exchanging data between two systems that are being synchronized. JMS supports this message distribution pattern with a specific subset of the API that supports the semantics of point-to-point messaging. We examine the API by considering the three basic phases our JMS client will experience: connecting to a provider, producing a message, and consuming a message.

CONNECTING TO A PROVIDER

Connectivity to a provider is based on a QueueConnectionFactory, which contains the configuration details required to connect to the provider. As discussed in Chapter 2, the QueueConnectionFactory, a JMS-administered object, should be retrieved from a JNDI namespace. It is of course possible to create a provider-specific QueueConnectionFactory in code, but as this compromises the JMS client's portability and is not a recommended JMS practice, we will not explore that option further. Rather, we assume that a QueueCon-nectionFactory has been defined and stored in a given JNDI namespace, using a provider-specific administration tool (we explore how this is achieved in Chapter 6, "IBM JMS–Administered Objects"). Our first task is thus to establish a context to the JNDI namespace.

CREATING AN INITIALCONTEXT

JNDI offers a rich API, but from the JMS client's perspective, we are interested in creating an InitialContext object with which we can look up named objects, specifically the Queue-ConnectionFactory. The InitialContext class acts as a starting point for accessing the naming system (namespace) and is created thus:

```
import javax.jms.*; // JMS classes
import java.io.*;// Standard Java
imports import java.util.*;
import javax.naming.*; // JNDI
imports ……
String jmsICF = "com.sun.jndi.
fscontext.RefFSContextFactory";
String jmsURL = "file:/c:/
JNDINamespace";
//connect to JNDI Namespace
Hashtable environment = new
Hashtable();
environment.put(Context.INITIAL_
CONTEXT_FACTORY, jmsICF);
environment.put(Context.PROVIDER_
URL, jmsURL);
Context ctx = new InitialContext(
environment);
```

Subscribe Today!

– INCLUDES –
FREE
DIGITAL EDITION!
(WITH PAID SUBSCRIPTION)
GET YOUR ACCESS CODE INSTANTLY!

The major infosecurity issues of the day... ***identity theft, cyber-terrorism, encryption, perimeter defense, and more*** *come to the forefront in* **ISSJ** *the storage and security magazine targeted at IT professionals, managers, and decision makers*

Editorial Mission

Greater Collaboration and Efficiency Through Education

✓ ***ISSJ*'s editorial mission is to showcase proven solutions** that will guide, motivate, and inspire senior IT and business management leaders in the planning, development, deployment, and management of successful enterprise-wide security and storage solutions.

✓ ***ISSJ*** brings together all key elements of data storage and protection, and presents compelling insight into the benefits, efficiencies, and effectiveness gained by focusing on these two critical areas of IT simultaneously.

✓ ***ISSJ*** is an objective, critical source of information that helps storage and security managers make informed management decisions about what is working today, and what they need to plan for tomorrow, and is the only publication that focuses exclusively on the needs of IT professionals who are driving the enterprise storage architecture/infrastructure while pursuing and incorporating the latest security technologies.

✓ ***ISSJ*** achieves our mission by delivering in-depth features, practical "how-to" solutions, authoritative commentary, hard-hitting product reviews, comprehensive real-world case studies, and successful business models, written by and for professional IT storage and security practitioners.

SAVE 50% OFF!
(REGULAR NEWSSTAND PRICE)

Only $39.99 ONE YEAR 12 ISSUES

www.ISSJournal.com
or
1-888-303-5282

SYS-CON MEDIA
The World's Leading i-Technology Publisher

Speech-enabled applications enhance customer self-service

Extending Your Investment with Voice

BY DUNCAN ROSS

You have built a self-service application on the Web – now, extend that to a telephone. Don't rebuild, extend. Extend the reach, extend the value, give customers flexibility, give them information on demand.

There is a growing understanding from the office of the CIO that a company no longer needs to abandon previously installed infrastructure every time a new innovation comes along. In fact, most companies are finding that they can move to the next level of functionality while retaining the value of past investments. One critical way to leverage development work and training is to voice-enable existing business applications. This is especially true in the case of customer service–oriented businesses like insurance, banking, transportation, and travel.

One of the often overlooked or misunderstood customer service tools is speech technology. Thanks to a number of successful implementations by well-known transportation companies and various financial services organizations, the perception is changing and we can safely say that good customer self-service by voice is possible today.

There are two key developments in speech technology that are making this type of self-service through conversational access a much more viable option for all companies looking to improve the customer experience. First, the old "black art" of telephony system development is being opened. More specifically, applications that were previously developed only by those with skills specific to proprietary IVRs, switches, and so on, can now be created by people with standard IT development skills. Significant improvements in the development environment and the integration of speech into the WebSphere portfolio are empowering mainstream developers to work with speech.

The second development is directly tied to the emergence of self-service over the Web. If you think about the way people are served over the Internet, for example in banking, there are a number of self-service applications that create a "dialogue" between the Web site and the customer. "Dialogue" applications are built to call up and display personalized data on command – or more specifically, on demand. The widespread acceptance and comfort that has developed around online self-service can now be transferred to a telephone interface. So, rather than building very basic touchtone, or DTMF, systems we can reverse the development process and start by transforming Web applications into voice applications. Previously, these two interfaces were completely separate (phone and Web) – now, a business can merge the information and the customer experience and use it over the phone, leveraging the same developers and business logic. In this model, systems are synchronized by using the same information and same application development. As a result, an organization does not have to build a customer service system for different channels of interaction from scratch but instead can use one architecture.

So, This Is Easy Now...?

Yes, ROI cases for automation of customer service by speech are eye-catching. The current systems can be user-friendly and offer a pleasant experience but speech technology is not the only salve. Executives in customer service often state that one of the biggest process challenges is that, despite all their efforts, they do not provide a consistent customer-centric experience across all channels. No amount of speech technology will create this interaction on its own. It has been estimated that as much as 40% of IS cost in major enterprises can be expended on achieving integration, or, more bluntly, just making stuff work together. To avoid wasting resources, businesses need to seek to create an on-demand operating environment that facilitates horizontal process integration and is flexible to change. If they do not take this approach, they will find that they are constantly dealing with the integration challenge instead of improving the customer experience.

To date, two of the key barriers

Duncan Ross has global responsibility for the sales and marketing of IBM's Voice Solutions portfolio. Part of the company's Pervasive Computing Division, the portfolio includes software products that facilitate self-service and assisted-service access to information and transactions, via telephones and other consumer appliances.
duncan_ross@uk.ibm.com

to implementing valuable speech applications have been skepticism about the value of an automated voice solution and poor implementation due to a lack of integration with existing systems. However, the benefits of exploring speech and potential opportunities within the enterprise have outweighed the challenges and provided an opportunity for IBM and its partners to adapt WebSphere offerings to overcome both customer misconceptions as well as business process challenges.

Overcoming Skepticism

Originally, voice applications meant dealing with your bank or credit card company over the telephone by responding to commands, such as “Please press or say one.” These interactive voice response (IVR) systems evolved over time from one word or discrete digit recognition systems to allow a few basic commands, such as “Please say, ‘Operator,’” or “Call Mom.” While somewhat effective, these systems often alienated customers and either led to dropped calls or longer wait times due to complex directions and misunderstood commands.

Over time, speech has evolved to enable more dynamic interactions between customers and conversational access systems, specifically, the ability to use names, codes, and commands. The flexibility of an application to understand a broad range of utterances including personal names, place names, products, ZIP codes, account codes, and commands tends to be a challenge for legacy IVR solutions. However, it is a significant strength of IBM’s current WebSphere Voice offerings.

The fundamental idea behind a voice application is conversation – one in which the user converses with a system, either in a structured dialog or menu (directed dialog application) or a more natural, freeform conversation (natural language understanding (NLU)). Voice applications have now gone well beyond one-word systems to provide a more natural system interface than pressing buttons on a telephone.

Lengthy menu navigation is no longer necessary and the new system is much more flexible, allowing callers to express the same request in different ways, such as “How much money do I have in Investment A?” or “Tell me the balance I have in Investment A.” Each call becomes more natural – similar to talking to a “live” customer support representative. Retirement account management becomes quicker and easier, with customers learning how to interact with the system faster, and making them happier with the results.

Improving Integration and Process Flow

A second barrier to adding speech to extend the enterprise has been the lack of a development environment that offers easy customization and integration with existing systems. By infusing three decades of speech experience into WebSphere, both of these challenges were addressed.

At the core, WebSphere allows businesses to tie together all of their data and existing applications and put a Web front end on them. It is composed of server software and development products bundled into packages and can be broken down into components. The WebSphere Voice Server then takes integration one step further by enhancing e-business applications with natural voice input and output, extending Web information to virtually anyone with a telephone and using existing industry standards, making it easy for companies to create speech-enabled applications.

WebSphere Journal Advertiser Index...

COMPANY	URL	PHONE	PG
CANDLE CORPORATION	WWW.CANDLE.COM/WWW1/PATHWAI	800-488-4246	2,3
CYANEA	WWW.CYANEA.COM/WSDJ/NOMOREMADNESS/HTML	877-CYANEA8	15
H&W COMPUTER SYSTEMS	WWW.HWCS.COM/WSD1.ASP	800-338-6692	51
KENETIKS	WWW.KENETIKS.COM	888-KENETIKS	19
MERCURY INTERACTIVE	WWW.MERCURYINTERACTIVE.COM/OPTIMIZEJ2EE	800-837-8911	9
MQSOFTWARE	WWW.MQSOFTWARE.COM/QNAMI	800-998-2858	21
NETIQ	WWW.NETIQ.COM/SOLUTIONS/WEB	408-856-3000	11
PROLIFICS	WWW.PROLIFICS.COM/WEBSERVICES	800-675-5419	29
QUEST SOFTWARE	HTTP://JAVA.QUEST.COM/JPROBE.WDJ	800-663-4723	39
SYS-CON MEDIA	WWW.SYS-CON.COM/2001/SUB.CFM	888-303-5282	41
LINUXWORLD CONFERENCE & EXPO	WWW.LINUXWORLDEXPO.COM	800-657-1474	34-35
TRILOG GROUP	WWW.FLOWBUILDER.COM	800-818-2199	52
WEB SERVICES EDGE EAST 2004	WWW.SYS-CON.COM	201-802-3069	42-49
WILY TECHNOLOGY	WWW.WILYTECH.COM	888-GET-WILY	5

As we eliminate the technical barriers to success with integrating speech and enhancing natural language interaction, we improve operational efficiencies and enhance the customer's experience. Over time, acceptance of speech technology within the call center will drive adoption in other areas of computing and communications and lead to the eventual development and demand for multimodal devices and applications.

Looking to the future, if enterprises already consider today's multichannel environment challenging for the IS department to support, we need to accept the likelihood (if not certainty) that customers will seek to interact with their service or product suppliers via a variety of new intelligent mobile devices. To avoid wasting resources on integration, businesses need to seek to create an on-demand operating environment that will sustain change. By integrating these two areas of customer self-service, they will go a long way in surpassing that hurdle.

Prudential Life Insurance Gets Conversational

BY RICK **SOULER**

Rick Souler is responsible for planning and leading corporate marketing, communications, and branding efforts at Viecore. Prior to Viecore, he was director of marketing for EG&G Sealol, a leading industrial supplier. Mr. Souler holds a BS in industrial engineering from Roger Williams University.
rick.souler@viecore.com

Viecore, Inc., a New Jersey-based systems integration firm specializing in customer self-service automation, recently implemented a speech solution utilizing the WebSphere product suite for Prudential Life Insurance. As an IBM business partner of over 13 years, Viecore has grown with the WebSphere family of products and is an integrator of choice for many complex deployments at mid- to large-size enterprises.

Viecore was selected as systems integrator for a self-service speech solution for Prudential Life Insurance based on the WebSphere platform. Prudential receives 3.3 million calls a year requesting information about individual life insurance. Before the implementation, approximately half of all incoming calls were handled by touchtone, or DTMF in the IVR, as it is known in the industry.

In addition to providing a sub-optimal customer experience, the system did not manage resources well. Many calls in the touchtone system were transferred out of the IVR to a customer service representative, which caused significant inefficiencies across the enterprise. In addition, all calls handled through the network router (both incoming and transfers from the IVR) incurred "per-click" charges from the outside network provider. In contrast, enterprises today are looking for solutions that improve operating efficiencies by reducing outside network charges, increase IVR retention rates, and improve call tracking. Call centers also seek to save money by decreasing the length of a call with a service representative. Streamlining can be achieved by using computer telephony interaction (CTI) to pass authentication information collected in the IVR to the agent – for example, a customer is asked up front for an account number, last few digits of a social security number, or date of birth so that the call can be processed faster.

Following the Viecore integration process, Prudential now utilizes WebSphere Voice Response (WVR), WebSphere Voice Server (WVS), and WebSphere Application Server (WAS) to deliver an enhanced customer experience. Prudential customers now use voice commands to conduct transactions such as payment information, service request status, policy value, loan information, tax information, ordering duplicate 1099s, and ordering duplicate statements.

The IBM WebSphere product suite provides a fully integrated solution platform for developing and sustaining high-quality, enterprise-level applications. The WebSphere family of products not only offers a scalable and reliable production environment, it also provides a host of integrated development tools that increases time-to-market.

Viecore implemented the Prudential solution on a 144-port WVR platform. WVS was used as the speech recognition engine because of its ease of integration with the suite of WebSphere products. WAS was used to interpret the Java/JSP code, connect to the back end (MQ Series), and serve up the VoiceXML. WebSphere Studio Application Developer and the WebSphere Voice toolkit accelerated development of the speech applications for quick deployment.

Prudential leveraged IBM's WebSphere product suite and Viecore's extensive speech and industry expertise to provide their clients with a state-of-the-art automated customer contact solution. Their new self-service speech solution is aimed at helping to reduce operating costs and increase their visibility in the insurance market as an industry and technology leader. Viecore performed the technical design, implementation, and testing of the new solution's functionality, working closely with both IBM and Prudential to ensure a high-quality, user-friendly, customer self-service solution. Together, Viecore and IBM aim to help Prudential realize maximum operational efficiencies and optimal customer service levels.

–continued from page 4

gration with WebSphere to support Web services in a service oriented architecture."

I am not, by any stretch of the imagination, a database expert, so I asked how Stinger stacked up to the latest Oracle and Microsoft offerings and he told me with confidence, "The degree to which DB2 is tackling automated performance improvement is light years ahead of the competition and DB2 has the lowest cost per transaction as measured by the TPC-C benchmark of $1.68 per tpmC (with 18,318 tpmC, or transactions per minute).

So there you have it – IBM has done it again. I can't wait to see what happens with WebSphere Studio and Application Server this year.

Reach Over 100,000

Enterprise Development Managers & Decision Makers with...

>Linux >Java >Web Services >.NET >XML >Wireless >Storage >Security
The Premier Resource for Today's Corporate & IT Decision Makers
IT solutions >GUIDE
WWW.SYS-CON.COM/IT
Top IT Solution Providers!
> Candle
> ClearNova
> Computer Associates
> Creative Science
> DralaSoft
> GraphOn
> InterSystems
> ITKO
> Merant
> O'Reilly
> Parasoft
> QuadBase
> SAP
> Wily Technology
> Zero G
TECHNOLOGIES YOU NEED NOW!
How to Manage Your Ideas Using Today's i-Technologies

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

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

Get Listed as a Top 20* Solutions Provider

For Advertising Details Call 201 802-3021 Today!

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

SYS-CON MEDIA

The World's Leading i-Technology Publisher

Goverments come up to speed with technology

Governments as On-Demand Enterprises

Without the make-or-break competition of the private sector, it might seem that governments have less need to invest in information technology to maximize their efficiency, cut costs, and improve collaboration and service to constituencies.

But the pressures on governments to more efficiently and effectively take timely action are intense. Many are looking to IT, specifically Web-based e-solutions and integrated systems, to help meet these demands.

One of the pressures on government is the need to increase collaboration across agencies and departments so they can securely share information that is needed to take quick and appropriate action. This is critical with today's safety and security issues and the need to manage crises. Yet government employees-must also balance safety issues with privacy considerations. Furthermore, increased cross-government collaboration is needed to improve overall service.

Governments are also under pressure to maximize operational efficiency so that they can meet the expectations of citizens and businesses for more responsiveness. Many constituencies prefer Web access to government and increasingly want to be able to use it to apply for permits and pay taxes. Governments also need to improve efficiency to meet a range of increasing regulatory requirements.

In addition, pressure to cut costs can be as intense in government today as in the private sector. The worldwide economic slowdown of recent years has cut revenue to governments just as it has private entities. Increasingly, governments are faced with balancing their budgets while providing continuous service to citizens.

To meet these challenges, governments need to integrate their systems across agencies and beyond. They have to focus outside of their organizations while maintaining safe and secure information. It is important that they become on-demand enterprises – organizations in which processes are integrated across all groups and partner institutions, allowing them to respond with speed to business and regulatory needs, changes, opportunities, and threats.

While IBM is working with all types of entities to become on-demand enterprises, the process becomes most valuable in an industry context. "That's because it's about the customer's needs, not speeds and feeds. That's why we seek out the industry issues that are important to our clients. We want a partnership role," said Wayne Janzen, the government market segment manager for IBM's software group.

Being an on-demand enterprise – for governments – means having an empowered workforce that is able to deliver higher-value productivity for its constituents. This allows them to better manage costs as well as to provide fast, high-quality responses to both routine and unexpected events.

IBM is working closely with many governments to help them become on-demand e-governments. A provider of IT solutions to governments since 1932, IBM has the #1 market share for IT solutions to government and both the Gartner Group and the International Data Corporation rank IBM as the #1 e-government solutions provider worldwide. Government is the #2 industry within IBM in terms of expenditure with the company.

IBM's experience with government customers is the foundation for its five middleware solutions designed specifically to address today's most pressing challenges for governments. The solutions are part of its effort to deliver industry middleware solutions – a strategy based on customer buying behavior that indicates they prefer to buy solutions designed for their industry.

Each industry middleware solution contains functions from IBM's WebSphere, Lotus, Tivoli, DB2, and Rational middleware brands combined with industry-specific middleware, applications from independent software vendors (ISVs), and industry-expert services. "By combining a common set of data and integration capabilities with application function from best-of-breed, industry-expert partners, IBM provides solutions that help customers meet constituents' needs and statutory requirements," said Janzen.

The five government solutions are IBM Middleware Solution for On Demand Workplace, IBM Middleware Solution for Government Access, IBM Middleware Solution for Emergency Response, IBM Middleware Solution for Government Collaboration, and IBM Middleware Solution for e-Forms/Records Management.

The on-demand workplace solution enables government employees to access critical applications and find and share information to work together more efficiently to provide services to constituents. "This improves the quality and speed of responses to routine and unexpected events," said Janzen. "It increases productivity and improves morale. It reduces costs."

The on-demand workplace solution decreases workplace complexity and empowers employees by simplifying their access to information and people. It helps governments improve organizational performance with existing staff. The solution reduces risk by providing timely collaboration with experts and more use of best practices.

IBM software in this solution includes Lotus Workplace Team Collaboration, Lotus Workplace Messaging, Lotus Workplace Collaborative Learning, WebSphere Portal, WebSphere Host Integration, Rational Rapid Developer, DB2 Information Integrator, and Tivoli Monitoring for Messaging. The IBM industry-specific middleware includes Global e-Business Solutions Center, WebSphere "As Is" and "To Be" Process Models for Human Resources, Dynamic Team Management Component Capability, WebSphere Portal Catalog, Communities of Practice, and Blended Learning. Industry-specific middleware from business partners includes CSC SmartWorkplace and Pinnacor Government News portlet.

"The on-demand workplace solution can be delivered with consulting and implementation services, as well as the software and hardware, which can be tailored to client needs," said Janzen. "But they don't have to buy a behemoth solution. It builds on existing IT investments. They can buy pieces, then build on them, and phase in on-demand over time."

This phase-in approach is possible with all the government solutions, including the IBM Middleware Solution for Government Access. "The access solution helps governments be more responsive by improving citizen and business access to information and services," said Janzen. "This can also help businesses meet their regulatory and compliance obligations."

The government access solution is designed to support multiple levels of government by providing constituents with continuous, user-friendly access to information, services, and benefits. It can also improve citizen access to critical knowledge with personalization features. It is designed to provide better, faster service via collaboration tools that allow government employees to help citizens through complex online transactions.

"It streamlines processes and improves efficiency by eliminating manual and redundant processes," said Janzen. "The result is a more responsive government, reduced regulatory/compliance burdens, and easy and available self-service." The improved efficiency cuts costs and enables employees to focus on critical issues.

Clients who have implemented the solution successfully include the Arizona Department of Motor Vehicles (DMV) and the State of Michigan.

The Arizona DMV had experienced an 8% increase in transactions for vehicle registration. The staff could not keep up with the increase and citizens were unhappy with long waits for DMV services. IBM helped the Arizona DMV implement a Web application that citizens can use 24 hours a day, seven days a week. Today, nearly 50% of DMV transactions flow through the ServiceArizona Web application, cutting the cost of processing renewals by 75%.

In Michigan, the state's e-government director wanted a way for citizens, businesses, and employees to access multiple state government functions from a single Web site. IBM helped build a centralized portal without alienating individual state agencies. Similar to the application used in Arizona, it is available 24 hours a day, seven days a week, significantly improving service and efficiency.

Providing around-the-clock access to information is also a key part of IBM's Middleware Solution for Government Emergency Response. Emergency responders (law enforcement, emergency response, and transportation management agencies) often use proprietary and incompatible messaging applications and database access systems. This solution provides a complete view of timely information – any time, anywhere – from across government and private organizations so it can be leveraged in response to emergency situations.

The emergency response solution is most valuable in metropolitan regions with multiple law enforcement, emergency response, and transportation management agencies that operate agency-based mobile data communications systems. Agency-based systems generally cannot communicate with systems operated by other agencies.

The solution also securely integrates critical data and enables more rigorous analysis and resource allocation through spatial analysis and visualization capabilities.

"The increased threat of terrorist attacks raises the importance of timely and effective communication among emergency response agencies," said Janzen. "Without it, emergency responses may not be appropriately coordinated and the safety of the public could be unnecessarily jeopardized. With this solution, emergency decision-making can be undertaken with a single view of the best available information in real time."

Use of the best available information is also one of the benefits of the IBM Middleware Solution for Government Collaboration. It enables the integration of processes across governments and government departments to create secure, real-time collaboration.

"The result is faster, higher-integrity, multi-department processes and a flexible enterprise that allows governments to provide new services and improve decision-making, coordination, and response to events," said Janzen. "The collaboration solution reduces risks by providing better information from expert employees, allowing rapid responses to changing policies, and reducing the cost of new services via the flexible architecture," he added.

For example, the United States Army is undertaking a pilot project with this solution. The project will help them gather intelligence information from various sources or formats and quickly distribute critical, sensitive data to key people. It would eliminate inflexible architectures and integrate information and processes that are largely manual and "stove-piped" by department, which prevents collaboration.

The project would create a system that allows faster decision-making and quicker reactions to existing and potential threats. It would be secure and always available, providing real-time communication, coordination, and collaboration among analysts using aggregated data from across organizations.

Management of data across government can also be improved with the IBM Middleware Solution for Government e-Forms/Records Management. Government agencies must comply with numerous laws regarding freedom of information, privacy, and the maintenance of historical and archival records. The volume of electronic and paper forms and records generated each day is tremendous. Governments must track all the e-mails, invoices, and other documentation dispersed throughout their network so they can be accessed when needed and disposed of when they are no longer necessary. Government regulations, as well as industry and legal standards, require this capability.

The e-forms/records management solution allows government employees to control document management and access, reduce paper-based processes and redundant data entry, and manage document retention. It is designed to improve operational efficiencies and help provide compliance with regulatory and legal obligations.

It does so by providing formal, rules-based management for processing, retention, and disposition of records and by enabling the secure electronic completion, review, verification, routing, and approval of forms and records. The electronic processing of forms is not only faster, it reduces costs and storage needs and decreases errors.

"This results in faster, higher-quality responses to information requests, reduced operating costs, and the ability to manage a large amount of records and comply with regulations and standards," said Janzen.

The California Franchise Tax Board, which monitors tax law compliance, had 260 million records across 50 federal, state, and local systems. With IBM, they built a Web-based, self-service site for constituents to submit tax forms electronically, download forms, get information, learn refund status, and respond to nonfiler notices.

The result was that an additional $200 million was generated in state tax revenues, with 100% payback within one year. In addition, the Tax Board saw a 55% reduction in erroneous taxpayer contacts and combining records enabled the Board to make 50,000 fewer calls.

Most governments would love to see such benefits. Many started building e-government functions in the 1990s, however, most were just simple transactions that automated pre-existing processes. They did not break down obsolete and bureaucratic divisions of information and processes.

"Because of all the pressures facing governments, the next phase of their e-Government efforts has to be the transformation into a flexible and collaborative civic service and governance model," said Janzen. "The first phase of e-Government transformation was to provide online services from nonintegrated departments to citizens and businesses. The second phase is to become the foundation for delivering integrated services to employees and consolidated operations across agencies."

The IT solutions needed to do this must tie into existing investments, be open so as to work well in the future, provide a choice of platforms, and allow entities to buy-and-build at their own pace, according to Janzen. Industry expertise and industry-specific product capabilities are also essential, he added, which is why IBM is working with a strong group of federal and regional systems integrators and independent software vendors.

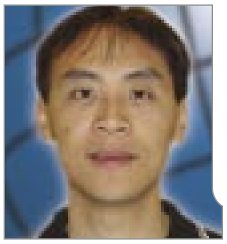
With these capabilities, Janzen said, "Governments can meet their challenges and become the responsive, flexible, and effective entities needed by their business and citizen constituents around the world."

Wayne Janzen is the government market segment manager for IBM's software group. He works with governments across the globe that are interested in understanding how existing and emerging technologies can be used to restructure programs, services, and operations. His work focuses on the development of solutions to improve citizen and business services, enhance economic development, provide for more efficient operations, and redefine how governments and their constituents interact.

Create and implement JMS applications with various support methods

JMS Support on the WebSphere Platform

BY COLIN YU



WebSphere Application Server (WAS) v5 is a J2EE 1.3-compliant application server. WebSphere Studio is a power development tool that allows users to develop J2EE applications and test the applications within a WebSphere test environment without deployment to a real server. Both WAS and WebSphere Studio have strong support for Java Message Service (JMS) and message driven beans (MDB).

This article discusses your options within WAS and WebSphere Studio for JMS applications, which include using MQ for Java Developers, WebSphere Embedded JMS Provider, and WebSphere MQ (formerly MQ Series). This article also focuses on setting up your deployment environment to support various JMS scenarios.

JMS Options on the WebSphere Platform

WAS supports asynchronous messaging based on the Java Messaging Service (JMS) of a JMS provider that conforms to the JMS specification v1.0.2 and supports the Application Server Facility (ASF) function defined within that specification. The following four options for the JMS provider are supported on the WebSphere platform:

- MQ for Java Developers (MQJD)
- Embedded JMS providers
- WebSphere MQ
- Third-party generic JMS providers

Colin Yu is a technical designer on the Business Scenarios team of the IBM Software Group System House at the IBM Toronto Lab. Colin helps define integration requirements for IBM Software Group products through the development of scenarios.
coliny@ca.ibm.com

MQ for Java Developers

By default, WebSphere Studio is configured to run a special JMS provider called MQ for Java Developers (MQJD). Unlike the normal Embedded Messaging Server, this MQJD Simulator is a pure Java JMS provider, so it avoids installation issues and the overhead of separate processes running in the development environment.

MQJD provides both point-to-point as well as pub/sub messaging support. Point-to-point messaging employs a queue as the JMS destination. One JMS client will retrieve the message from this queue. The public/subscribe service employs a topic as the JMS destination. A message is published to a topic. Multiple JMS clients may subscribe to this topic, and each subscribing client will receive the message.

MQJD implements all of the JMS APIs, but it does not support security checks, persistence, or JMS communications across process boundaries. As a result, it does not work with application clients. This means that an application client or a standalone Java application cannot communicate with an MQJD that is using a JMS API.

Embedded JMS Provider

In J2EE, Java Messaging Service becomes an integral part of the platform. WAS v5 satisfies this requirement with the embedded WebSphere JMS provider. The implementation of the embedded WebSphere JMS provider is based on the integration of IBM WebSphere MQSeries product and the JMS Client (MA88) support pack into WAS. The versions of these embedded products are reduced-footprint and reduced-function versions of the independently shipped MQ product.

MQJD provides both point-to-point and pub/sub messaging support. Each JMS server is responsible for managing the WebSphere MQ Queue Manager and the internal provider of the publish/subscribe service.

The embedded JMS provider may be used by server-side components such as EJBs, JSPs, and servlets running in the WAS, and the Java applications running as J2EE application clients. However, it doesn't support APIs other than JMS API.

Because the embedded JMS provider is a reduced footprint of MQSeries, there are some limitations to using it. It is not possible to exchange messages with queue managers outside WAS, including WebSphere MQ. Therefore, it is not useful for heterogeneous communication between WAS and other environments. Also, some advanced options such as QMgr/QMgr channels, message flows, and message transformation are not supported.

WebSphere MQ

WebSphere MQ, formerly known as MQSeries, is IBM's foundation product for business integration and has been the market leader in message-oriented middleware for nearly 10 years, providing not only the de facto standard for messaging, but also supporting the fast growing language of today's business applications that want to communicate using JMS.

WebSphere MQ supports additional qualities of service that are not present in the embedded JMS provider, such as remote queuing, QMgr channel configuration, message flows, encrypted client-server communications, and greater performance tuning flexibility. Options also exist in WebSphere MQ for clustering that do not exist in the embedded JMS provider. The key addition that makes these topologies possible is that WAS v5 ships with the embedded messaging client, which is the same as the WMQ 5.3.0.1 client. This supports XA connections to a remote queue manager on the distributed platforms.

WebSphere MQ allows you to exchange information across different platforms and integrate existing business applications in the process and offers comprehensive security options using secure sockets layer (SSL), the Internet standard for secure communication.

Besides JMS API, WebSphere MQ provides support for different programming languages including C, C++, Visual Basic, COBOL, Assembler, RPR, PL/I, and TAL. WebSphere MQ provides the built-in point-to-point support. For publish/subscribe support, one of the following publish/subscribe brokers is required:

- *WebSphere MQ MA0C Pub/Sub Support Pack:* www-1.ibm.com/support/docview.wss?rs=203&uid=swg24000643&loc=en_US&cs=utf-8&lang=en
- *WebSphere Business Integration Event Broker:* www-306.ibm.com/software/integration/wbieventbroker
- *WebSphere Business Integration Message Broker:* http://www-306.ibm.com/software/integration/wbimessagebroker/

JMS Deployment Topologies

With the JMS options outlined above, the following deplyment topologies for JMS server on the WebSphere platform use avaiable to users.

MQJD comes with WebSphere Studio. You can use MQJD as the JMS server for server-side components running on the WebSphere test environment, a testing instance of WAS (see Figure 1).

This option is only available in the WebSphere test environment of WebSphere Studio. The benefit of this option is its simplicity. You do not need to do any further configuration before you can test your JMS application with WebSphere Studio.

Because MQJD does not listen on a socket for requests from other processes, this configuration does not support JMS communications with other servers or processes. It can only receive messages sent by applications within the same unit test server process. If your application doesn't require application clients or standalone Java applications to communicate with the JMS server, this is the right choice for testing purposes.

Because MQJD does not support security checks or persistence, the security constraints on the JMS objects will not take effect. Once the server restarts, all the messages will be gone.

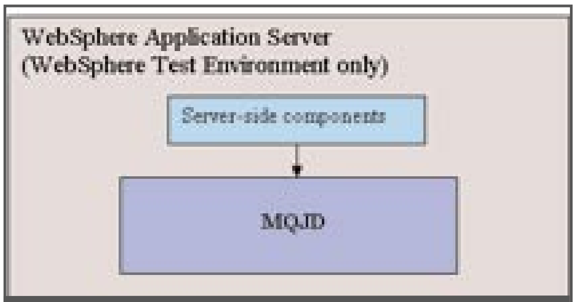

WebSphere Application Server
(WebSphere Test Environment only)
Server-side components
MQJD

FIG 1: USING MQJD AS THE JMS SERVER

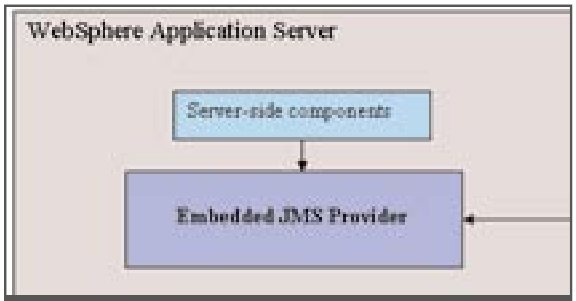


FIG 2: EXTERNAL APPLICATION CLIENTS COMMUNICATE WITH THE JMS SERVER

DEPLOYMENT TOPOLOGY WITH EMBEDDED JMS PROVIDER

WAS and WebSphere Studio come with a fully functioning embedded JMS provider, which is the default JMS provider for WAS. Similarly, you can configure WebSphere Studio to support the embedded messaging server instead of MQJD.

The benefit of using an embedded JMS provider is that you do not need to pay extra for it. By default, when you

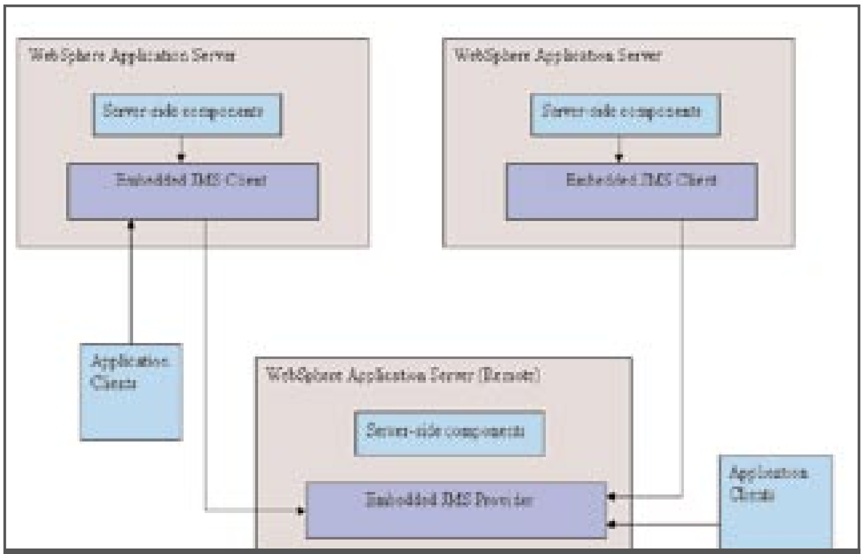


FIG 3: SHARING A REMOTE EMBEDDED JMS PROVIDER THROUGH THE EMBEDDED JMS CLIENT

install WAS, the embedded JMS provider including JMS Client and JMS Server will be also installed on the same box.

With an embedded JMS provider, you have the necessary listeners that allow external application clients to communicate with destinations in the JMS server (see Figure 2).

This environment also gives you the added benefit of setting up different scenarios, such as multiple WAS instances sharing a remote embedded JMS provider through the embedded JMS client (see Figure 3).

If your application does not require programming language support other than JMS, and you do not need the JMS server to communicate with other JMS servers, an embedded JMS provider is the right choice for you.

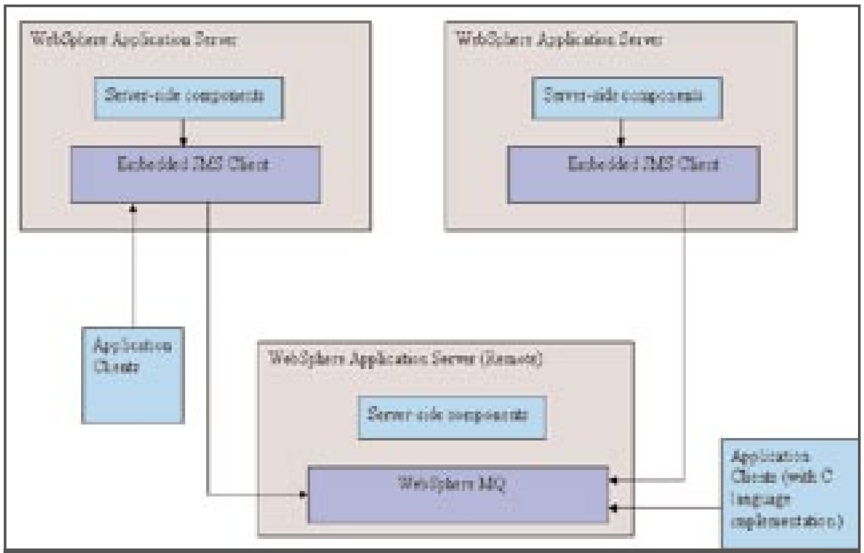

WebSphere Application Server
Server-side components
Embedded JMS Client
WebSphere Application Server
Server-side components
Embedded JMS Client
Application Clients
WebSphere Application Server (Remote)
Server side components
WebSphere MQ
Application Clients (with C language implementation)

FIG 4: **WEBSPHERE MQ AS THE JMS SERVER**

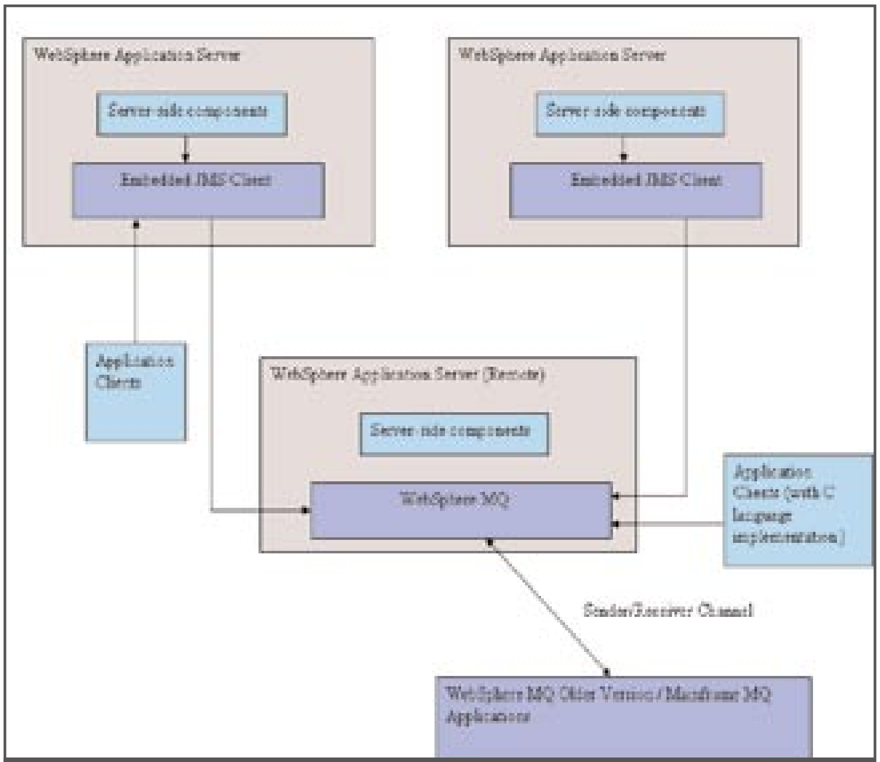

WebSphere Application Server
Server-side components
Embedded JMS Client
WebSphere Application Server
Server-side components
Embedded JMS Client
Application Clients
WebSphere Application Server (Remote)
Server-side components
WebSphere MQ
Application Clients (with C language implementation)
Sender/Receiver Channel
WebSphere MQ Older Version / Mainframe MQ Applications

FIG 5: **COMMUNICATING WITH AN OLDER VERSION OF MQ**

DEPLOYMENT TOPOLOGIES WITH WEBSPHERE MQ

Finally, you can configure WebSphere MQ as the JMS server. WebSphere MQ supports the topologies similar to the one for the embedded JMS provider (see Figure 4).

One of the benefits of using WebSphere MQ is its support for different programming language such as C, C++, Visual Basic, and COBOL, besides the JMS API.

If you need to communicate with an older version of MQ, perhaps mainframe applications, and support XA, you can setup a JMS server with WebSphere MQ 5 that communicates to older versions of MQ using sender and receiver channels (see Figure 5).

If you need to support different message clients or interchange with legacy MQ applications, WebSphere MQ is the right choice for you. If you are looking for the additional features such as clustering, message flows, encrypted client-server communications, and greater performance tuning flexibility, then upgrade to WebSphere MQ.

Conclusion

This article illustrated how WAS supports JMS using MQJD, WebSphere Embedded Messaging, and WebSphere MQ. It was intended to help you make decisions on how to choose the appropriate JMS product that fulfills the requirements of your applications.

A LIMITED TIME SAVINGS OFFER FROM SYS-CON MEDIA

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

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

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

CALL TODAY! 888-303-5282

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

☐ 3-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 6-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 9-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.

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

☐ **MX Developer's Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $167 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $29.99 /	Save: $60
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $137 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $40
Int'l - Two Years (24) $216	You Pay: $89.99 /	Save: $127 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $59.99 /	Save: $30
Digital Edition - One Year (12)	You Pay: $19.99	

☐ **Linux World Magazine**

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

☐ **Java Developer's Journal**

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

☐ **Web Services Journal**

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

☐ **.NET Developer's Journal**

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

☐ **XML-Journal**

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

☐ **WebLogic Developer's Journal**

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

☐ **ColdFusion Developer's Journal**

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

☐ **Wireless Business & Technology**

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

☐ **WebSphere Developer's Journal**

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

☐ **PowerBuilder Developer's Journal**

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

3-Pack

Pick any 3 of our magazines and save up to $275^00

Pay only $175 for a 1 year subscription plus a FREE CD

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

6-Pack

Pick any 6 of our magazines and save up to $350^00

Pay only $395 for a 1 year subscription plus 2 FREE CDs

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

9-Pack

Pick 9 of our magazines and save up to $400^00

Pay only $495 for a 1 year subscription plus 3 FREE CDs

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

*WHILE SUPPILES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Subscribe Online Today www.sys-con.com/2001/sub.cfm

SYS-CON MEDIA

Speech in the Call Center

BY BRIAN GARR

Streamlining the call-flow process

When combining self-service and traditional call center components, it is important to clearly define the call-flow process so that all interactions work seamlessly. The solution should incorporate built-in contingency plans that connect customers with a live agent as soon as additional assistance is required. Savvy companies achieve the best of both worlds by enhancing the customer experience while reducing the overall cost of customer care. Lowering the unit cost of customer interaction is a complex process that involves weighing the pros and cons of numerous interconnected variables.

In the 1970s and 1980s, most speech technologies concentrated on routing calls through IVR systems, handling simple inquiries, and generally yielding very low customer satisfaction results. Over the past three years, advancements in conversational technologies have enabled the deployment of sophisticated and engaging personas to handle more complex transactions with an improved customer experience. Conversational access applications have moved from the research lab to the IT center as a valued solution.

The next step is to develop and refine processes, methods, and technologies in speech applications. Voice is the most natural and standard interface for communications. Voice applications represent a powerful medium in a global economy for improving the dynamics in channeling care.

In 2003, with the broad acceptance of VoiceXML the industry saw a major shift in the way conversational applications were deployed. Business logic became separated from the user interface, creating an environment where back-end processes could be leveraged and reused across multiple modalities. VoiceXML dramatically reduces the time and cost of deploying new conversational solutions, and IBM has augmented those benefits by creating graphical tools based on the open standard Eclipse framework. Using these advanced VoiceXML generating tools, developers can visually map out the call flow of their applications.

Speech has gone through all of the phases of a startup as it moves from a lab experiment, to a few good deployments, to a mature technology that can have a significant impact on the call center, generating greater customer satisfaction, customer retention, and additional revenue streams. Teams in IBM Research have made tremendous progress in natural language understanding (NLU), the study of spoken word recognition and unstructured dialogue interactions. The most common use of NLU today is for call-routing systems that use the opening dialog "How can I help you?" and then use NLU to parse the users' utterances and transfer them to the appropriate person or conversational application. More sophisticated NLU applications, such as the one deployed at T. Rowe Price, allow you to complete a broad set of transactions, such as buying or selling mutual funds, through a conversational interface.

Users expect to be able to customize their experience when they are dealing with a conversational interface with multiple applications. Call center managers want to be able to add and delete services without having to rewrite their entire applications. As both audiences begin to embrace new technologies in speech, we will witness the transition from analysis to implementation, resulting in optimum solutions for driving down costs, while shifting investments to the strategic dimensions of the care program.

"Conversational access applications have moved from the research lab to the IT center as a valued solution"

Brian Garr is the segment lead for the call center and voice portal segment for IBM pervasive computing. He is an evangelist and speaker worldwide on machine translation, text to speech, and speech recognition. He received the Smithsonian Institute's "Heros of Technology" designation in 1998 for his work in machine translation. bgarr@us.ibm.com