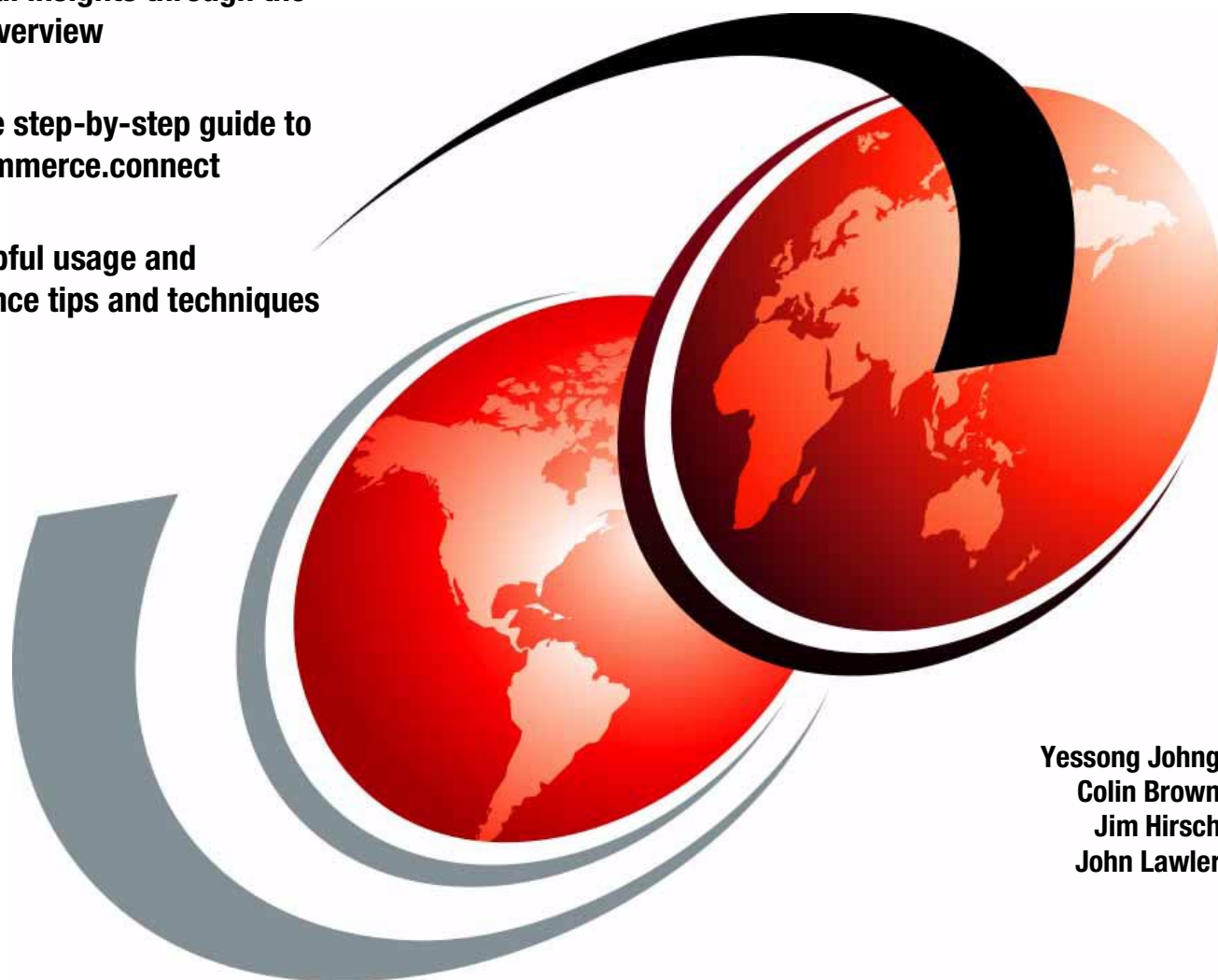# IBM

Geac System21 commerce.connect

Implementation on the IBM *e*server iSeries Server

Gain useful insights through the product overview

Follow the step-by-step guide to install commerce.connect

Learn helpful usage and performance tips and techniques

Yessong Johng
Colin Brown
Jim Hirsch
John Lawler

Redbooks

IBM

International Technical Support Organization

**Geac System21 commerce.connect:
Implementation on the IBM** @server **iSeries Server**

December 2002

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

First Edition (December 2002)

This edition applies to Geac call.connect Version 1.1, vendor.connect 1.1, and System21 3.5.2b SP4 or SP5.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks(logo)™	iSeries™	S/390®
AS/400®	MQSeries®	SecureWay®
DB2®	OS/390®	SP™
DB2 Universal Database™	OS/400®	TCS®
IBM®	Perform™	VisualAge®
IBM eServer™	Redbooks™	WebSphere®

The following terms are trademarks of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both:

Lotus®	Notes®	Word Pro®

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM Redbook introduces the new Geac commerce platform *.connect* applications – the *call.connect* and *vendor.connect* applications. These applications extend and enhance the capabilities of Geac System21 into the intranet and Internet.

This redbook targets consultants or customers who work with these .connect applications. It explains how to install, maintain, integrate, and manage these applications on the IBM @server iSeries server. It also helps you to understand the architecture and middleware used by the applications.

Prior to reading this book, you must be familiar with the basic, traditional use of the iSeries or AS/400 and System21. For example, you should know how to enter simple commands and understand such concepts as the library list. Similarly for System21, you should be familiar with the menus and such tasks as defining a System21 user.

As necessary throughout the book, detail is provided about the newer, less traditional features of the iSeries such as the integrated files system (IFS), Qshell, Java, and WebSphere.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Rochester Center.

Yessong Johng is an IBM Certified IT Specialist at the IBM International Technical Support Organization, Rochester Center. He specializes in WebSphere and Domino implementation on iSeries, with a focus on their integration. Recently Yessong expanded his expertise to include Linux and its solutions on the iSeries server.

Colin Brown is a Senior Software Architect at Geac United Kingdom (UK). He has 15 years of experience in software design and implementation. He holds a degree in computer science. His area of expertise includes Enterprise JavaBean (EJB) component development.

Jim Hirsch is a Test Manager at Geac UK. He has 20 years of experience in various IT disciplines. He holds a degree in math from London University. His areas of expertise include AS/400, iSeries, and Geac System21.

John Lawler is a Technical Consultant at Geac UK. He has 18 years of experience in IT. He holds a degree in mathematics from Oxford University. His areas of expertise include AS/400, iSeries, UNIX, Windows NT, RPG, C, Java, and WebSphere.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

ix

Find out more about the residency program, browse the residency index, and apply online at:

`ibm.com`/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

► Use the online **Contact us** review redbook form found at:

`ibm.com`/redbooks

► Send your comments in an Internet note to:

redbook@us.ibm.com

► Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. JLU Building 107-2
3605 Highway 52N
Rochester, Minnesota 55901-7829

1

The .connect applications

This redbook covers the two Geac .connect applications:

► call.connect
► vendor.connect

This chapter introduces you to these two applications.

Geac also offers the following applications, which provide security, process control and messaging, and integration with System21:

► secure.connect
► process.connect
► inter.connect

These are part of the commerce platform. *commerce.platform* is the element of *commerce.connect* that supports the integration of applications and processes with System21. commerce.platform includes:

► The process.connect business modeling tool and workflow engine to define and automate business processes

► Secure.connect to manage and control user access to information and processes

► A series of components that contain the business rules and connections needed to integrate System21 with external applications

To learn how vendor.connect uses process.connect and inter.connect, see Chapter 2, "Architecture of the commerce.connect products" on page 7.

1

1.1 call.connect

Geac call.connect fills two roles. First, it is a telesales-oriented product. It is intended to help call center personnel actively sell to the customer and create and foster personal relationships.

To support this type of active sales, call center personnel need instant access to all relevant information for the calling customer. Typical information includes order history, account information, and product information. The ability to quickly enter an order (complete with stock allocation, customer pricing, and credit checking) while the customer is still on the telephone is paramount. This should be backed up by the ability to script the conversation to highlight selling and promotion opportunities.

Companies need to classify their markets and customers and determine their policy in satisfying conflicting priorities and supplies. Geac call.connect is a customer service and order taking application that is designed to be deployed in this fast moving and complex environment to meet these requirements.

call.connect is a component-based order capture application. This component-based approach allows the Geac Professional Services Organization to build solutions that optimize the order capture process for individual customers. Indeed, this component-based approach allows Geac consultants to build new order capture solutions for sales force automation or mobile computing, for example.

Figure 1-1 shows a typical call.connect window. The top two-thirds of the window show the Reactive Sales page. This contains products that are relevant to the particular customer to which the operator is selling. These products may be ones that the customer buys on a regular basis, ones that are on special promotion, or ones that are on the special price list for the customer. The bottom third of this window shows a configurable set of tabs. These pages contain information about the current customer, which may help the operator in the telesales environment.
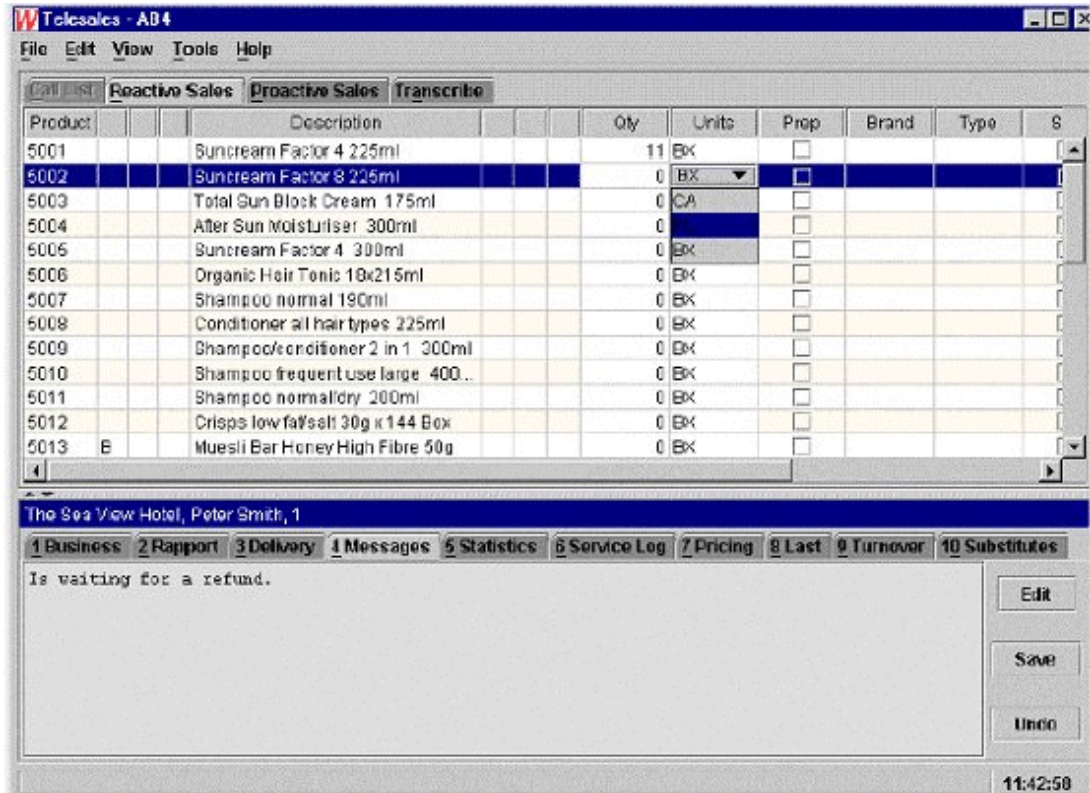
Figure 1-1   The call.connect buying list

call.connect is a flexible order capture process that includes the following facilities:

- ► **Dynamic buying lists**: The lists of products that the customer is likely to purchase is created by combining a fixed list of products with a dynamic list, based on rules applied to previous purchase history.

- ► **Call management**: Supervisors have access to a call-management application that allows them to track the progress of calls and re-assign them to operators to ensure adequate throughput. Operators can be assigned a skill level to aid the allocation of customer calls to operators.

- ► **Stock allocation**: Rule-based stock allocation and sourcing engine.

- ► **Promotions**: A rule-based promotions engine supports a variety of promotions (for example, buy one get one free), special prices, discounts, and loyalty points.

- ► **Active selling**: Includes support for up-sell and cross-sell. *Up-sell* prompts the operator to suggest a higher value product to appropriate customers. For example, when the operator selects a 21-inch television, call.connect prompts an up-sell opportunity to sell a 25-inch television. *Cross-sell* prompts the operator to offer associated products. For example, when the operator selects a video recorder, call.connect suggests some blank tapes to go with it.

- ► **Order management**: Includes standing orders, order copy, and the ability to "park" an order for later completion. call.connect is a "front-office" order capture and customer service application that integrates with System21 for order fulfilment processing (picking, despatch, etc.). The component-based approach allows the application to be extended to support, for example, integration with call-center telephone exchanges to provide efficient call routing and improved customer responsiveness.

1.2 vendor.connect

Geac vendor.connect is a supplier self-service application. It is designed to support a cooperative relationship between customers using System21 and their suppliers. It makes information available, exchanges business documents, and allows controlled direct update facilities.

vendor.connect improves supplier communication, aids planning, and reduces inventory investment in the supply chain. Figure 1-2 shows an example of the vendor.connect Replenishment page.

Figure 1-2   The vendor.connect Replenishment page

vendor.connect supports a number of supplier relationships. This includes vendor-managed inventory, direct delivery to the end customer, and normal purchase orders. vendor.connect provides:

► *Web-based interfaces to System21* to support:

  – Enquiries on orders, returns, and receipts
  – Displays of relevant System21 information to the supplier
  – New orders and changes to orders since the supplier last visited the site are highlighted for convenience
  – A search facility and rule-based exception notification

► *Order transmission*: In addition to the ability to print or fax orders, you can send orders to the supplier as e-mail attachments. A Web page allows the supplier to acknowledge receipt of the order and update the System21 purchase order appropriately.

- *Supplier planning*: An enquiry allows the supplier to view the stock status and demand for products for which they are the preferred supplier. The demand enquiry includes relevant information from System21 – unallocated sales orders, expected demand from Material Requirements Planning (MRP), expected demand from Distributed Requirements Planning (DRP), and a historical demand element based on sales and customer-specific filters (for example, sales current year to date or previous year to date).

- *Maintain blanket order delivery schedule*: Allows the supplier to amend a blanket purchase order to specify planned delivery dates and quantities.

- *Promise date update*: Allows the supplier to specify the date they plan to make the delivery. For direct delivery orders, the supplier updates the promise date for the associated sales order and e-mails the customer with the delivery notification.

- *Create Advanced Shipping Notification (ASN)*: Allows the supplier to construct a shipment from current order lines and amend quantities on the delivery as required.

- *Support for entering packaging details to allow shipping documentation to be created*. Allows the customer to receive against the ASN. This reduces the time it takes to get products into the warehouse.

- *Delivery of direct customer orders*: Allows the supplier to build the details of a direct shipment to an end customer. They can use this as the start of an @ctive process to initiate customer billing.

- *Documentation print*: Allows the supplier to print standard shipping documentation (barcode labels, quality reports, etc.) to be sent with the shipment.

Architecture of the commerce.connect products

The products that make up the commerce.connect platform originate from a diverse background. The challenge and vision is to bring these applications together into a single coherent, product strategy. This requires an architecture that can encompass the entire technology spectrum from the legacy applications to the leading-edge Enterprise Java applications.

This chapter explains the key parts of the architecture, specifically the architecture of call.connect and vendor.connect, which are the two applications that were designed and developed from the ground up around an Enterprise JavaBean (EJB)/Java 2 Platform, Enterprise Edition (J2EE) architecture. Both of these applications integrate with System21. This chapter also highlights the key integration points.

2.1 The need for an architecture

Why is an architecture necessary? Can't we simply write programs that deliver the function that is required? The reasons for having an architecture are:

► Ever increasing demands are placed on systems in terms of security and availability.

► The need to extend the system to both customers and suppliers across the Internet is growing rapidly. Applications need to have this capability "architected in".

► The need to reduce the product development life-cycle, while delivering more complex systems at the same time, means that Geac simply cannot develop its systems from scratch or the infrastructure required.

► There is a need to connect different systems (both Geac and external systems) to provide a viable, reliable, and robust solution.

To meet these requirements, Geac has to rely more and more on infrastructure or middleware to provide these services. In turn, this means that Geac should clearly architect, develop, and deploy its software to maximize the benefits that the chosen middleware offers.

Over two years ago, as part of the strategic alliance with IBM, Geac chose to base its new e-business enterprise applications around WebSphere.

The WebSphere suite of products enables Geac to develop, deploy, and integrate next-generation e-business applications. This includes such applications for business-to-business e-commerce. Geac also supports business applications from simple Web publishing through enterprise-scale transaction processing, extending applications to incorporate mobile devices, etc.

The entire WebSphere philosophy allows Geac, as an Enterprise Application Developer, to build, integrate, and deliver solutions more timely to market using WebSphere. WebSphere is the cornerstone of IBM's enterprise development strategy. There is little functional or time availability differences between the release of WebSphere on the iSeries server and WebSphere on Windows 2000. This allows Geac to deploy its J2EE applications on the platform that is best suited to the particular customer.

Many of the middleware services provided by WebSphere form part of IBM's implementation of the J2EE specification. J2EE defines the standard for developing multi-tier enterprise applications. J2EE simplifies enterprise applications by basing them on standardized, modular components, providing a complete set of services to those components, and handling many details of application behavior automatically, without complex programming.

However, it is true (from practical experience) that both the EJB specification and WebSphere do not completely remove or absolve the implementor of the responsibility of using the above services appropriately. Performance needs to be "designed" into the application. Key architectural decisions still need to be made within the constraints and goals of the project and these decisions need to be well documented and understood by everyone involved in the project. This ensures a consistent and high quality approach when designing and implementing a large-scale project.

2.1.1 Key Enterprise JavaBeans and WebSphere Application Server benefits

The Architectural Specification Geac follows is Enterprise JavaBeans 1.0 (EJB 1.0). As discussed, the Application Server that Geac uses to implement this specification is IBM's WebSphere Application Server Advanced Edition. If the applications that Geac were implementing were entirely PC-based, WebSphere Advanced Edition would not be necessary, but this is not the case.

The EJB standard is a server-side Java-based component architecture for building multi-tier, distributed, enterprise applications. WebSphere Application Server provides such services as:

- **Authentication and security services**: Are provided either via the host operating system or through Lightweight Directory Access Protocol (LDAP).

- **Transaction management**: Ensures integrity of data, not just within a single database, but across the enterprise.

- **Resource pooling**: Allows more efficient use of valuable system resources. This is especially important in a large Internet deployment scenario, where there may be thousands rather than tens of users apparently concurrently using the system.

- **Clustering and high availability**: Enable scalability and ease the implementation of a fault tolerant high availability system.

Developing EJBs and deploying within WebSphere offers other key benefits that are often overlooked in the context of a single project, but that *must* be considered within the technical strategy of the company.

For the first time in Geac's System21 development history, they can deploy the WebSphere-based Business Application (in the form of EJB components) that they develop on almost any Enterprise Server where there is a business benefit. Currently Geac's requirements are restricted to iSeries and Windows NT or Windows 2000. But in the wider Geac worldwide context, they could deploy onto OS/390 and most UNIX variants, including Linux.

As the e-marketplace develops, a significant number of third-party EJB components will become available. A simple example of this is be Secure Credit Card Authorization. Using a common architecture across multiple components reduces the number of possible failure points, simplifies deployment, and in this example, can actually guarantee true transaction integrity.

2.1.2 The architecture moving forward

Java standards relating to the enterprise have evolved over the past couple of years. The biggest push on standards by IBM, Sun, and others has been around J2EE. J2EE encompasses all of the Java standards relating to middleware and enterprise application development. The EJB specification is now part of the J2EE standard.

Geac's architectural strategy is to follow, comply with, and implement J2EE solutions. For example, in the short to medium term, Geac will:

- Use JMS as its messaging subsystem with MQSeries as the message transport layer wherever possible. This allows for maximum flexibility and reliability between Java- and non-Java-based systems.

- Ensure that the Geac's infrastructure software, such as process.connect and inter.connect, not only coexist on the same server as one of its WebSphere EJB-based applications, but will take full advantage of the facilities offered by WebSphere.

- Implement and support the next generation of Web Technologies such as Universal Description, Discovery, and Integration (UDDI). For more information, see:

  http://www.uddi.org

2.1.3 The development process

Geac uses a development process based around the *Rational Unified Process*, but heavily modifies it. Development process refers to a process that starts at project inception and goes all the way through deployment onto the customer's site. Rational Software is the prime contributor to the Rational Unified Process.

A key process difference to the normal Geac approach is that the process is iteration-based, rather than traditional waterfall based. The process splits the project into several smaller deliverables rather than one large deliverable at the end of the project. Therefore, a project can consist of several smaller iterations each with identifiable, measurable objectives. Another key difference is that the process focuses on risk elimination. The idea is that identified high risks (for example, potential technical problems) are addressed early on in the process to avoid large surprises later on. This reduces the risk of costly failures late in the project. For more information, see:

http://www.rational.com

The architecture has a key role to play in the development process. Early in the process, any requirements that may impact the architecture, such as the "user interface is going to be Web-based and PC client-based", are taken into account and the risk is assessed. When there are high technical risks to a project, these risks are mitigated in an "elaboration phase", which attempts to develop a "proof of concept" deliverable that focuses on the specific risks.

A risk that Geac identified was the need to re-use the existing System21 RPG code base for a specific business function. Geac saw this as a high risk area, because of the different technologies involved and potential performance considerations. If a viable solution could not be found, this would greatly increase the cost of the project because of the need to rewrite large parts of existing business logic in Java.

2.1.4 Implementation

All software development is carried out under VisualAge for Java 3.5.4. Developers test and deploy locally in the WebSphere Test Environment before testing on the iSeries server. The database is always DB2 Universal Database (UDB) for iSeries. Geac uses a shared VisualAge Repository for Source Code control.

2.1.5 The design methodology: Using Unified Modelling Language

Geac uses the Unified Modelling Language (UML) for both high-level and detailed design. It uses Rational Rose as the design tool that supports UML.

Geac's design methodology is more a component than strict object-oriented (OO)-based design. It designs and implements course grained components based around business subsystems. Each of these subsystems has one or more defined interfaces.

Figure 2-1 shows an example of the primary relationship between the main SalesOrder subsystem and its dependencies for call.connect.
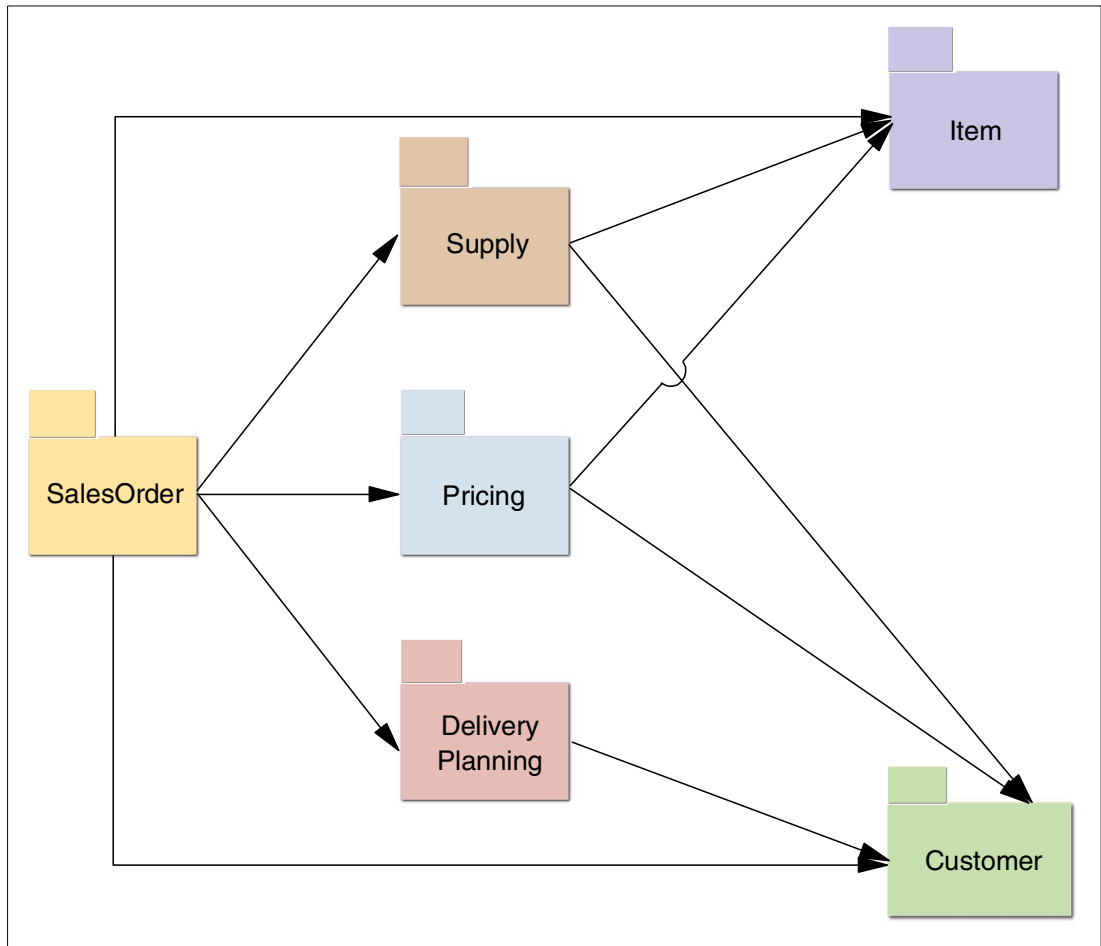
Figure 2-1   SalesOrder subsystem dependencies

For vendor.connect, Figure 2-2 shows the relationship between the controlling servlet and the vendor.connect subsystems.
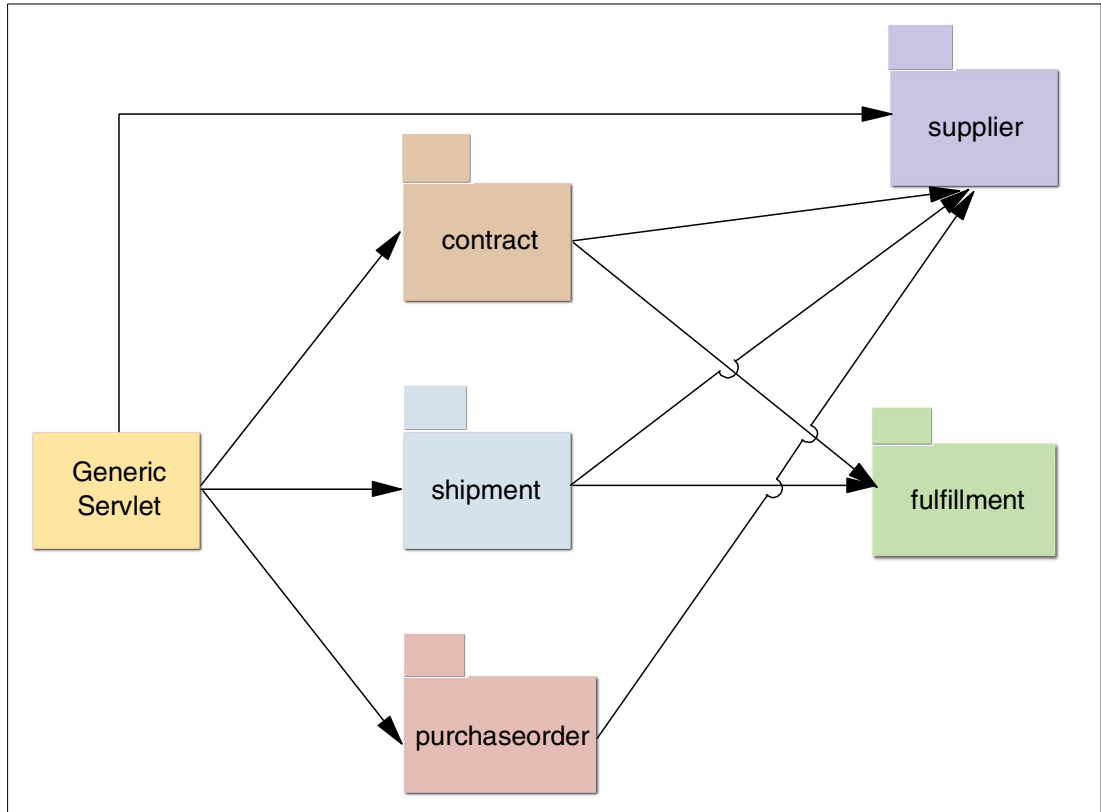
Figure 2-2 vendor.connect dependencies

call.connect components are designed to be extensible, reusable, and replaceable. This means that the business functionality can be extended without needing to cut and paste entire programs. This helps to reduce maintenance overhead and allows existing code to be reused.

2.2 Messaging: Java Message Service and IBM WebSphere MQ

The term *messaging* refers to the sending of structured information between two or more subsystems. Typically the message is sent asynchronously. That is that the sender can continue to run without waiting for a reply from the receiver. Also, the sender and receiver typically run as separate processes and possibly on physically separate machines.

2.2.1 call.connect

Both call.connect and vendor.connect use asynchronous messaging to meet specific requirements:

call.connect uses messaging to meet two requirements:

► To send information back to the client user interface (UI) such as "the order has been put on hold due to credit problems", or "a given item is out of stock"

► To allow for time-consuming steps to be completed asynchronously, for example, order completion when the order is actually committed into System21

call.connect messaging is implemented using IBM's implementation of the Java Message Service (JMS), which itself uses MQSeries as the underlying messaging facility. You can learn more about JMS at:

http://java.sun.com/products/jms/index.html

Figure 2-3 shows how call.connect uses messaging.
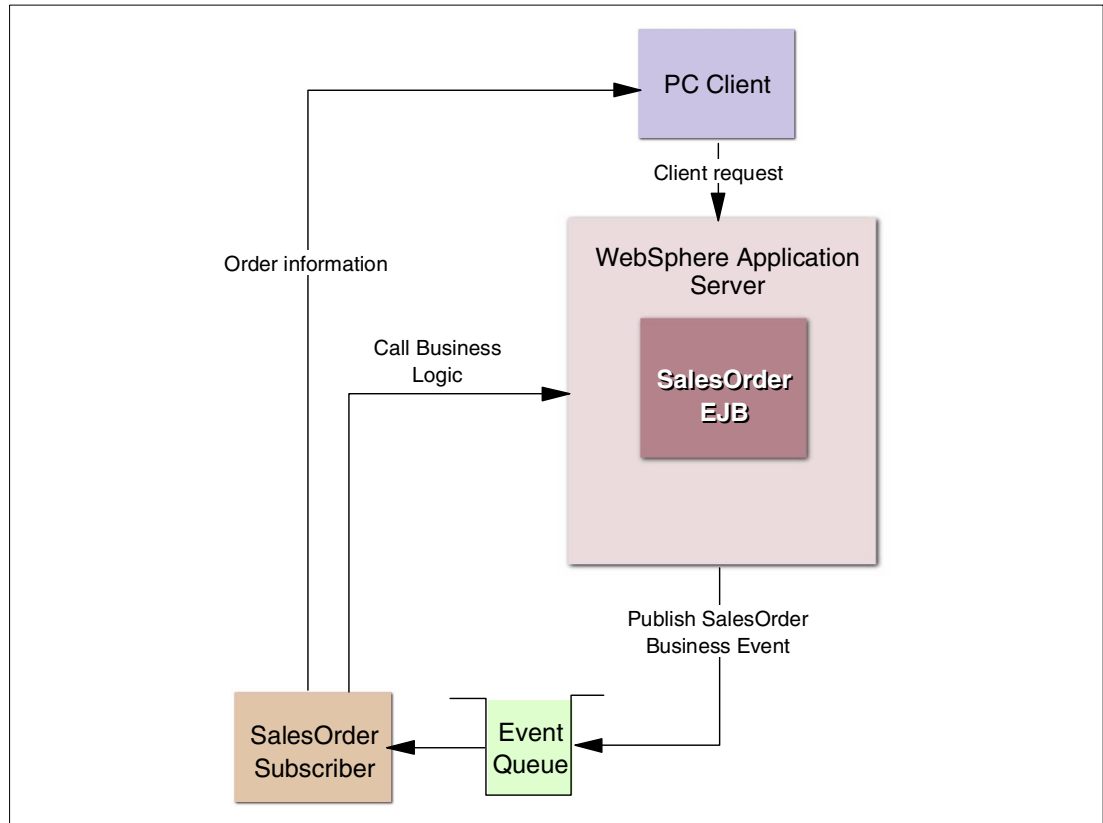
Figure 2-3   call.connect messaging usage

Adding a Sales Order Line scenario

This example explains what happens when an order line is added to a Sales Order by an operator:

1. The SalesOrder EJB processes the addline in the normal way.

2. The Business Event Addline is published.

3. Control returns back to the client ready for further input.

4. The SalesOrder subscriber is a separate process that listens for SalesOrder events and receives the Addline message.

5. The subscriber calls back into the SalesOrder EJB to perform a "check for unusual quantity" request to see if the customer normally orders this amount of stock for the item concerned.

6. If the quantity ordered is unusual, the SalesOrder subscriber notifies the PC client by making a call back into the client. The client displays the notification text in the UI status bar.

2.2.2 vendor.connect

vendor.connect uses messaging in conjunction with process.connect. The main requirement is to synchronize and respond to changes in System21 data such as purchase order updates. Figure 2-4 shows how this is achieved.
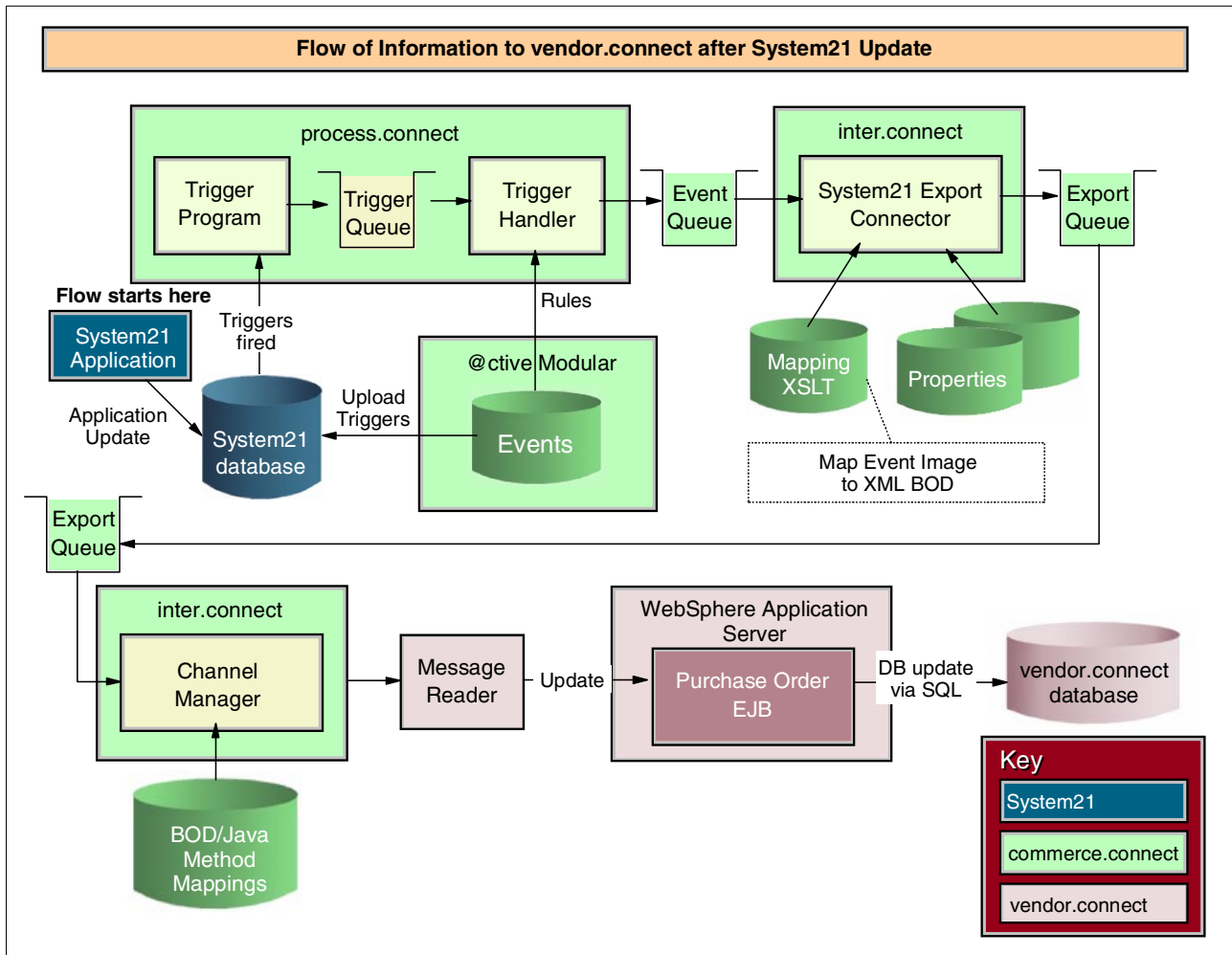
Figure 2-4   vendor.connect messaging usage

MQSeries is used as the underlying message transport facility by process.connect.

Figure 2-5 shows the relationship between the .connect applications, the commerce.connect platform, System21, and the IBM middleware-WebSphere and MQSeries application server.

PC Client Rich
Java Swing UI

Web Browser
UI

.connect application

Middleware

commerce.connect

System21

HTML Web Pages

Web server

vendor.
connect
Servlets/JSPs

Call
Business
Function

Call Business Function

EJB Container

Authenticate/
Authorize
User

call.connect
EJBs

vendor.connect
EJBs

secure.
connect

Send XML Document

Send XML Document

inter.
connect

WebSphere MQ

SQL via JDBC

Via Jacada Server

System21 Business
Function

process.
connect

Database Triggers

System21
Database

Product Interaction Overview

Figure 2-5   Relationship between the .connect applications

2.3 Overview of process.connect

process.connect provides a business modelling tool and workflow engine to define and automate business processes. It delivers optimal business processes that electronically improve the performance of mid-market companies. It integrates the commerce.platform and the commerce.connect applications, @ctive Processes and @ctive Modeler, to deliver rapid modelling, implementation, and execution of critical-business activities.

process.connect can also integrate the processes of the System21 organization with those of its suppliers or customers. A process can interact with customers (or suppliers) by sending an

e-mail requesting information. The e-mail reply is then used to control the next step in the process. For example, in an order fulfilment process, if an order cannot be fully allocated, the process can e-mail the customer and ask if a part shipment is acceptable. Links in the e-mail send a yes or no response back to the process, which continues appropriately.

2.4 Architectural representation

The deployment requirements (for example, must be able to run on a single iSeries machine) considerably affect the overall architecture in terms of physical location of database, code, etc. However, Geac ensures that the architecture is not constrained by these physical limitations.

There is a logical view of the architecture that identifies each significant layer of the system and shows how that layer maps onto a design and implementation tiered architecture. For example, Geac can deploy the Business Logic for vendor.connect onto either the iSeries server or Windows 2000.

2.4.1 Architectural goals and constraints

In an ideal world, the architecture allows and assists the product team to build the ultimate system with infinite flexibility, scalability, ease of use, and installation. In the real world, each interested stakeholder has their priorities for the system.

For a sales person, the system should be seen as infinitely flexible and expandable. In reality, meeting these aspirations may lead to an architecture that is overly complex and cumbersome for the main day-to-day running of the system. Therefore the architecture is a trade-off between many requirements.

2.4.2 Non-functional architectural considerations

This section details the key requirements of the system that may significantly impact the architecture:

► **Compatibility with System21**: The architecture needs to consider that all .connect applications have a constraint that they must be compatible with System21.

► **Performance and scalability**: The system must meet specific volume and response time requirements for the project where the product will be deployed.

► **Re-usability**: Both call.connect and vendor.connect were originally developed as part of individual specific customer projects. The justification for the development of is that Geac expects to have a significant amount of design and implementation re-use for future projects. Therefore, the ability to re-use and extend the core business logic is vital to the success of the product.

► **The follow on to re-usability is supportability**: Geac must be able to enhance and add functionality to the product while supporting existing installations from the common design and code base.

► **e-commerce support**: The call.connect EJB Business Logic components form the back-end order entry system for the Geac e-commerce solution, web.connect. Therefore, the architecture must support or be extendible to support the Web deployment model of servlets, JavaServer Pages (JSPs), etc.

► **Deployability**: In the short term, where possible, both call.connect and vendor.connect should be deployable on a single iSeries running alongside System21 on the same physical hardware. Due to hardware and system software constraints, it may be necessary

to upgrade the hardware, software, or both. The minimum hardware requirements obviously depend on the usage profile of the customer as well as the base WebSphere Application Server requirements.

2.4.3 Functional architectural considerations

This section details current known functional requirements that may likely have a major impact on the architecture of the system.

The deployment influence

Certain aspects of the architecture are strongly influenced by the targeted deployment implementation platform of Java, WebSphere, and EJB.

What? No entity beans?

From previous architectural prototypes, and call.connect Release 1, and the current level of EJB support within WebSphere, Geac chose not to use EJB entity beans for database access. This was primarily on the grounds of performance, which was found to be significantly slower than the equivalent hand-crafted Java Database Connectivity (JDBC) database access.

There were also concerns over issues to do with legacy integration, specifically RPG programs potentially accessing the same tables as those mapped by an entity bean. This was an indication that there could be further performance problems.

Geac intends to re-evaluate the use of entity beans for the next release of call.connect. The main benefits are of improved productivity and maintainability.

Session beans

The EJBs that contain the call.connect and vendor.connect Business Logic are designed and deployed as *stateless session beans*. This means that they do not (directly) maintain any state specific data (for example, Order Number) across method calls.

In theory (and proven in practice by using the WebSphere Performance Monitor), this allows the EJB container (WebSphere) to pool session beans, thereby reducing the resource usage on the system. Even if there are 75 telesales operators all entering orders into the system, there may be only 10 instances of the Sales Order Session Bean actively running (and consuming system resources) on the system.

2.5 Reusing and extending System21 business logic

This section details the options that are available when considering how to maximize the reuse of existing legacy application logic.

2.5.1 Accessing System21 RPG business logic

The need to reuse existing System21 logic was identified early on in the original call.connect project. The following technical options were available.

IBM Toolbox for Java (previously AS/400 Toolbox for Java)

This is an obvious candidate. This has many classes for accessing iSeries objects including calling programs.

There are two disadvantages:

► The first is primarily stylistic. Using Toolbox for Java would make Java look more iSeries specific.

► The second is that Toolbox for Java would form its own connection independently of any JDBC connection. This would mean that updates made by iSeries programs called in this way could not easily be part of the same commit transaction as updates made through JDBC.

The first objection is more apparent than real. Toolbox for Java is 100% Pure Java. Therefore, its use does not restrict the portability of the Java code. The second objection is more significant.

Java Native Interface (JNI)

Sun defines a technique for calling between Java and native code called the *Java Native Interface*. By native, Sun means non-Java code supported by the underlying platform. All of the Sun documentation discusses C and C++, but it is also possible with RPG on the iSeries.

At V4R4 and V4R5, it was quite difficult to integrate Java and RPG. At V5R1, it became much easier. If V4R5 had to be supported, then the coding complexity would be an issue.

At V5R1, there was no longer an issue but another objection remained. JNI is a direct call and the native code would have to be on the same system. This would force an element of Java onto the iSeries and restrict the deployment options for the application. The entire application would not have to be on the iSeries since some form of remote invocation could be used, but would add a considerable degree of complexity.

Stored procedures

There are two forms of stored procedure on the iSeries server. They may be written in Structured Query Language (SQL) or in most other supported languages for example, RPG and CL.

Geac does the reverse of typical usage. It does not have a stored procedure and then decide to write it in RPG. Instead, Geac already has RPG and creates a stored procedure, which becomes the program. This stored procedure can then easily be called using JDBC from Java. For a programmer who has learned basic SQL and JDBC, this is easy.

The choice

JNI is probably the best performing option. However, its restrictions on deployment options eliminated it. Also V5R1 did not exist at the time that Geac made the choice, and the complexity of JNI prior to V5R1 counted against it.

The performance of the IBM Toolbox for Java and stored procedure options is similar. The coding complexity is similar but stored procedures have the advantage of similarity with other SQL. The final decision was made based on the ability of stored procedures to share connections and partake in transactions.

3

Installing and setting up call.connect

This chapter explains how to install and set up call.connect and vendor.connect. Prior to reading this chapter, you must be familiar with the skills listed for each product and have knowledge about System21 and its configuration.

Installing call.connect involves a number of steps on both the server and on each client. Before starting, be sure to verify that all prerequisites are in place. Use care in following the steps that are provided. Failure to do so can result in problems with the installation of these products. Once these products are configured, they require little or no maintenance.

Throughout the installation instructions, you will see variables in *italics*. For each of these variables, you need to provide a real name. For example, for the variable *<iSeries>*, you must change it to the name of the machine you are using.

19

3.1 Skills and prerequisites for installing, running WebSphere Application Server

This section outlines the skills you need for the installation and the prerequisites that must be in place before you start.

3.1.1 Skills

You must be familiar with the workstation, particularly with the operation of the keyboard and mouse. You should also be familiar with the following applications, which are used in the call.connect configuration. Although instructions are provided at each stage, a complete beginner may find the installation to be difficult and may need help from someone with more experience.

- ► Windows NT Explorer
- ► Notepad and Wordpad or other tool to edit files (for example, .bat and .xml files)
- ► Client Access
- ► MS-DOS
- ► OS/400 commands
- ► A Web browser (for example, Internet Explorer)
- ► System21

In addition, you should also be familiar with:

- ► WebSphere Application Server
- ► CL programming on the iSeries
- ► SQL

For straightforward configurations, you may be able to proceed regardless of your familiarity with these items.

3.1.2 Prerequisites

Before you install WebSphere Application Server, make sure that you have met all hardware and software prerequisites. The following sections list the iSeries hardware requirements, iSeries software requirements, workstation hardware requirements, and workstation software requirements.

For more information, see:

http://www-1.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/docs/doc.htm

iSeries hardware requirements

Any of the following servers (recommended minimums) are required:

- ► iSeries Model 170 with processor feature 2385
- ► iSeries Model 720 with processor feature 2062
- ► iSeries Model 270 with processor feature 2252
- ► iSeries Model 820 with processor feature 2396
- ► 1 GB of memory (recommended minimum)

> **Note:** You may use earlier systems than these recommended minimums in environments that support a limited number of users and where longer server initialization times can be tolerated.

Table 3-1 lists the disk requirements.

Table 3-1   Disk space requirements

Installation option	During installation	After installation
*BASE (client application development software only)	500 MB	250 MB
Option 1 (includes *BASE and WebSphere Application Server environment)	650 MB	450 MB

iSeries software requirements

The minimum software required for your iSeries includes:

► **OS/400 Version 4 Release 4, or later** (in an unrestricted state): To install and run WebSphere Application Server Advanced Edition for iSeries on your iSeries server.

► **iSeries user profile with *ALLOBJ authority**: To install WebSphere Application Server Advanced Edition for iSeries on your iSeries server.

► **iSeries Developer Kit for Java (5769-JV1), Version 1.2 (option 3)**

► **OS/400 Qshell Interpreter (5769-SS1, option 30)**: For using the scripts that are included with the product and for local installation (installing to your iSeries server from the CD-ROM of your iSeries server).

► **OS/400 Host Servers (5769-SS1, option 12)**: For remote installation (installing to your iSeries server from the CD-ROM of another workstation). You can start the host servers by using the Start Host Server (STRHOSTSVR) command. On the iSeries command line, type:

```
STRHOSTSVR *ALL
```

The QSERVER subsystem must be running on iSeries.

► **TCP/IP Connectivity Utilities for iSeries (5769-TC1)**: To configure and run WebSphere Application Server. It is also needed if you are using remote installation (installing WebSphere Application Server on your iSeries server from the CD-ROM of another workstation). To start TCP/IP on iSeries, enter the Start TCP/IP (STRTCP) command on the iSeries command line.

► **HTTP Server for iSeries (5769-DG1)**: To support requests for servlets and JavaServer Pages resources managed by WebSphere Application Server. It is also needed if you plan to use Secure Sockets Layer (SSL) protocol. If you plan to deploy only enterprise beans, HTTP Server for iSeries is not needed. call.connect optionally uses Sun's Java Web Start technology to allow dynamic updates of client applications. Java WebStart requires an HTTP Server to service update requests. However, call.connect can be used without enabling these features.

► **DB2 Universal Database (UDB) for iSeries**: Must be configured to work with WebSphere Application Server for iSeries if you plan to connect to the local database. Optionally, this database may be on a different machine than the iSeries server that is running WebSphere Application Server. We recommend that you use the local iSeries database during the initial WebSphere Administration Server setup.

► **Entry in the relational database directory that points to *LOCAL**: To run WebSphere Application Server on iSeries.

To view the current settings, enter the Work with Relational Database Directory Entry (WRKRDBDIRE) command. You can add a directory entry by entering the Add Relational Database Directory Entry (ADDRDBDIRE) command.

The maximum number of jobs allowed for the iSeries SQL server jobs should be set to *NOMAX. Use the Change Prestart Job Entry (CHGPJE) command to change the prestart job entry for the SQL server jobs. For example, you would enter:

```
CHGPJE SBSD(QSYSWRK) PGM(QSQSRVR) MAXJOBS(*NOMAX)
```

- ► **DB2 Query Manager and SQL Development Kit for iSeries (5769-ST1):** This is an optional product that can be helpful in developing client applications.

- ► **All necessary fixes**: For a current list of fixes, see:

  http://www.iSeries.ibm.com/WebSphere

  When you reach this site, click **PTFs**.

Workstation hardware requirements

Workstations and software configurations, other than those in the following list, are capable of running WebSphere. However, they are not included here because they have not been tested running call.connect.

The capable workstations include any Intel-based personal computer capable of running any of the following operating systems:

- ► Windows NT Server or Workstation V4.0
- ► Windows 2000 Server or Advanced Server
- ► Support for a communications adapter or an appropriate network interface
- ► 40 MB of free disk space (minimum)
- ► 96 MB of memory (minimum)
- ► CD-ROM drive

Workstation software requirements

The minimum software required an each workstation running call.connect includes:

- ► One of the following operating systems:
    - – Windows NT Server V4.0, SP 6A or later
    - – Windows 2000 Server or Advanced Server

- ► IBM Development Kit for Java: Windows NT IBM enhanced Java Development Kit, Version 1.2.2

> **Note:** These products are included on the WebSphere Application Server Advanced Edition workstation CD-ROM:
>
> - ► TCP/IP must be installed and running on your workstation
> - ► A Web browser that supports HTML 4 and cascading style sheets (CSS)
> - ► Client Access
>
> You can install these products by selecting the IBM JDK 1.2.2 install option when you install the Administrative Console.

3.1.3 System21 authorization code for Order Management (OM)

You must obtain an authorization code, from your Geac distributor, for the System21 Order Management application (and all other System21 applications) before you can use it. You must apply the code in the usual way.

3.2 Standard installation procedures

This section provides detailed instructions for installing call.connect. It includes sections on a basic standard setup using the configuration supplied on the CD and a section on setting up the new System21 application Order Management.

Checklist of the basic steps

Table 3-2 summarizes the steps required to install and configure call.connect. You can find additional information on each step in the following sections. For full details, see the *Geac call.connect Installation Guide*.

Table 3-2   Summary of the call.connect installation and configuration

Step	Action	Completed
1	Load System21, including standard applications and the Order Management application	
2	Load Java components and configuration files for call.connect	
3	Install and configure WebSphere Application Sever	
4	Set up journaling	
5	Create stored procedures and run SQL scripts	
6	Set up Java Message Service (JMS)	
7	Create user profiles	
8	Set up System21 data	
9	Install Java Web Start	
10	Back up the installation (for example, the entire configuration)	

3.2.1 Installing Order Management and call.connect

This section explains how to load the Order Management System21 application and the call.connect IFS objects.

> **Note:** This redbook does not provide instructions for installing System21. You can find information on installing System 21 and current details regarding your installation scenario in the *Geac System21 Installation and Setup Guide*.

System21 base

To run call.connect, you need to load at least the following System21 applications at V3.5.2b Service Pack 5:

▶ Application Manager
▶ Geac System21
▶ Advanced Order Entry (AO)
▶ Distribution Requirements Planning (DR)
▶ Inventory (IN)
▶ Order Entry (OE)
▶ Cash Management (CS)
▶ General Ledger (GL)
▶ Sales Ledger (SL)

You can obtain these applications from your Geac distributor.

System21 Order Management (OM)

In addition to the System21 applications, you must load the Order Management System21 module. This module is supplied on the call.connect CD. Follow these instructions:

1. From the CD, load the Order Management module as explained in the readme file of the CD. This installs the following libraries:
   – OSLOMF3: Contains data files
   – OSLOMD3: Contains all other Order Management objects

   The source is available in the OSLOMS3 library, but this requires a special order and is not normally shipped.

2. After you install the libraries, initialize the application with the following command:

   ```
   AMINSAPP APSR(0) APPL(OM) RLSL(03) LIB(OSLOMD3)
   ```

3. Apply the authorization code for OM with the command:

   ```
   STRIPGCF
   ```

4. Select option 4 and enter the codes that are provided next to the Order Management application.

Java components and configuration files

All other components of call.connect are on the call.connect CD. You can load them by following the instructions in the readme file.

The components are then loaded onto the iSeries in the /OrderManagement folder. This folder contains the following subfolders:

► Cfg
► Log
► Deployed
► Client
► Mgr
► Stored procedures
► SQL scripts

3.2.2 WebSphere installation and configuration

This section explains how to start the WebSphere administration server, import the configuration file, and start the application.

You can find complete instructions about how to install and configure WebSphere Application Server on the WebSphere Application Server Web site at:

http://www-1.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/docs/doc.htm

Starting the administration server

To start the administration server, follow these steps:

1. On the iSeries, enter the command:

   ```
   STRSBS SBSD(QEJB/QEJBSBS)
   ```

   This starts the QEJB Subsystem and two auto-start jobs QEJBADMIN and QEJBMNTR.

   WebSphere cannot start until the administration server is ready.

2. To find out the status of the administration server, check the job log for job QEJBADMIN. Enter the following command:

   ```
   WRKACTJOB SBS(QEJBSBS)
   ```

3. Type 5 next to QEJBADMIN to view the job.

4. Select option `10` to view the job log. Wait until you see the message `WebSphere administration server QEJBADMIN ready`. This may take several minutes. The first start after the installation may be even slower.

Importing the configuration file

Important: To import the configuration file, the WebSphere instance *must* be running, but the console *must not* be running.

Run the import tool inside Qshell. Follow these steps:

1. Run the Start Qshell (STRQSH) command. After each command, wait for the dollar ($) signs to appear.

2. Switch to the current directory within Qshell. Enter the following command (Figure 3-1):

   `cd /QIBM/ProdData/WebASAdv/bin`

```
                             QSH Command Entry

     $
   > cd /QIBM/ProdData/WebASAdv/bin
     $













  ===> XMLConfig -adminNodeName homer -import OrderManagement/Config/Config.xml
  -substitute "nodename=homer;dir=/OrderManagement;appname=SalesOrder;container=Sa
  lesOrder"

  F3=Exit    F6=Print F9=Retrieve F12=Disconnect
  F13=Clear F17=Top  F18=Bottom  F21=CL command entry


MA    a                    MW                                  20/010
```

Figure 3-1   Importing the WebSphere configuration to an iSeries called homer

3. Import the Config.xml configuration file from OrderManagement/Config.

4. To use the delivered file to configure the default instance on an iSeries server, enter the following command. Substitute your server name for *iSeries*. Enter:

   ```
   XMLConfig -adminNodeName <iSeries> -import /OrderManagement/Config/Config.xml
   -substitute
   "nodename=<iSeries>;dir=/OrderManagement;appname=SalesOrder;container=SalesOrder"
   ```

   The system may run for several minutes.

   Notes:

   ► Leave a space before each - sign, and leave a space after each keyword, for example, `-adminNodeName` , `-import` , and `-substitute` .

   ► There is only one pair of quotes around the entire substitution string.

The import includes:

► The node
► The SalesOrder application and all contents
► The JDBC driver required
► The datasource required.

The config.xml file contains the following variables, which are substituted on the import command and replaced with real names:

► **$NodeName$**: This is the WebSphere node name. The node name in WebSphere is case sensitive. This is relevant when you start the console with AdminClient and when you specify an Internet Inter-ORB Protocol (IIOP) address for a Java Naming and Directory Interface (JNDI) lookup (for example, within the batch files that start client programs). It is also relevant to tools such as the XML export/import utility.

Usually the name is entirely in lowercase. On some systems, it is entirely in uppercase and could, in theory, be a mixture of both.

If you find that the console fails to start with such messages as "`Could not get attributes`" (similar to when the service/subsystem is not started), but the service is started, then the problem may be that you are using the incorrect case on the name.

You can verify the required name by using the following SQL to look at a WebSphere table:

```
select NAME
from EJSADMIN/NODE_TABLE
```

This shows the node name exactly as WebSphere wants it to appear. Do not be tempted to change the contents of this table. It is liable to make things worse rather than better.

Note: You must have QSECOFR user authority to read this table.

► **$AppName$**: This is the application name that is typically SalesOrder for the default instance.

► **Dir**: This is the root directory for the installation. This is the directory that contains the subdirectories cfg, deployed, and log. For the default instance, this is usually /OrderManagement.

Note: The application name should be less than 11 characters in length so that you can clearly see it on the iSeries server when you use the WRKACTJOB SBS(QEJBSBS) command. Longer names are allowed but must be truncated when viewed this way.

► **$ContainerName$**: This is the container name that is typically the same as the application name, but can be different.

Starting the application

To start the application, follow these steps:

1. Start a WebSphere console. In a command prompt, change the directory to:

   `..\WebSphere\appserver\bin`

   Note: This is a standard directory for a default installation of the console. However, you can install the console in a different directory if necessary.

2. To start the console for the specified machine in the default environment, enter the command:

```
adminclient <iSeries> <Port Number>
```

This may be quite slow. If an instance other than the default instance is required, enter the port number. Otherwise, you may leave this blank.

3. Wait for the message "`Console Ready`" to appear in the console. The topology view appears by default.

4. Open the node, application server, and container to display the beans. Click the **Application Server**. Click **Start** and wait. The beans turn blue as they start. This may take several minutes.

3.2.3 Journaling

Because WebSphere applications run under commitment control, files used by call.connect must be journaled. There are many options to provide different levels of security.

Since files from different libraries must be journaled, the journal receiver and the journal must be created in a new library rather than an existing System21 library. The following files must be journaled to operate call.connect:

► All physical files in library OSLOMF3
► The following files in library OSLD1F3:
    – INP40
    – INP60
    – OEP05
    – OEP40
    – OEP41
    – OEP45
    – OEP45E
    – OEP55
    – OEP56

However, merely journaling these files does not give the user the full advantages of journaling such as extra security.

Performance improves if files are journaled to an auxiliary storage pool (ASP). You can use the commands in the following sections with or without an ASP.

Journaling the files without an ASP

To journal these files without an ASP, complete the following steps. These are sample commands only and may be varied as required.

1. Sign on as `QSECOFR`.

2. Create a new library for the journal receiver and journal, for example, OSLF3:

```
CRTLIB LIB(OSLF3) TEXT('OSL journal library')
```

3. Create a journal receiver:

```
CRTJRNRCV JRNRCV(OSLF3/OSL0001) THRESHOLD(50000) TEXT('System21 journal receiver.')
```

4. Create a journal:

```
CRTJRN JRN(OSLF3/OSL) JRNRCV(OSLF3/OSL0001) MNGRCV(*SYSTEM) DLTRCV(*YES) TEXT('System21 journal')
```

Journaling the files with an ASP

To journal the files using an ASP, follow these steps:

1. Create a separate library for the journal receiver:

   ```
   CRTLIB LIB(OSLF3R) TEXT('OSL journal receiver library')
   ```

2. Create a journal receiver:

   ```
   CRTJRNRCV JRNRCV(OSLF3R/OSL0001) ASP(2) THRESHOLD(50000) TEXT('System21 journal
   receiver.')
   ```

3. Create the journal:

   ```
   CRTJRN JRN(OSLF3/OSL) JRNRCV(OSLF3/OSL0001) MNGRCV(*SYSTEM) DLTRCV(*YES) TEXT('System21
   journal')
   ```

Journaling a single file

To journal a single file, such as OMP00, use the following command:

```
STRJRNPF FILE(OSLOMF3/OMP00) JRN(OSLF3/OSL) OMTJRNE(*OPNCLO)
```

Journaling multiple files within a library

To journal multiple files in a library, follow these steps:

1. Start the Programming Development Manager (PDM):

   ```
   STRPDM
   ```

2. Select option 9 (Work with user-defined options).

3. You see the Specify Option File to Work With display (Figure 3-2). Press Enter to accept the defaults (File QAUOOPT, Library QGPL, and Member QAUOOPT).

```
                    Specify Option File to Work With

  Type choices, press Enter.

    File  . . . . . . . . . .    QAUOOPT      Name, F4 for list

       Library . . . . . . . .     QGPL       *LIBL, *CURLIB, name

    Member  . . . . . . . . .    QAUOOPT      Name

















   F3=Exit     F5=Refresh     F12=Cancel
```

Figure 3-2   Working with user-defined options

4. On the next display, press F6 to create a new user-defined option.

5. On the Create User-Defined Option display (Figure 3-3), enter the following information for the parameters listed:

- Option = `SJ`
  - Command = `STRJRNPF FILE(&L/&N) JRN(OSLF3/OSL) OMTJRNE(*OPNCLO)`

Press Enter.

```
                     Create User-Defined Option

 Type option and command, press Enter.

   Option  . . . . . . . . .   SJ   Option to create

   Command . . . . . . . . .   STRJRNPF FILE(&L/&N) JRN(OSLF3/OSL) OMTJRNE(*OPNC
 LO)
 _____
 _____




 F3=Exit      F4=Prompt     F12=Cancel
```
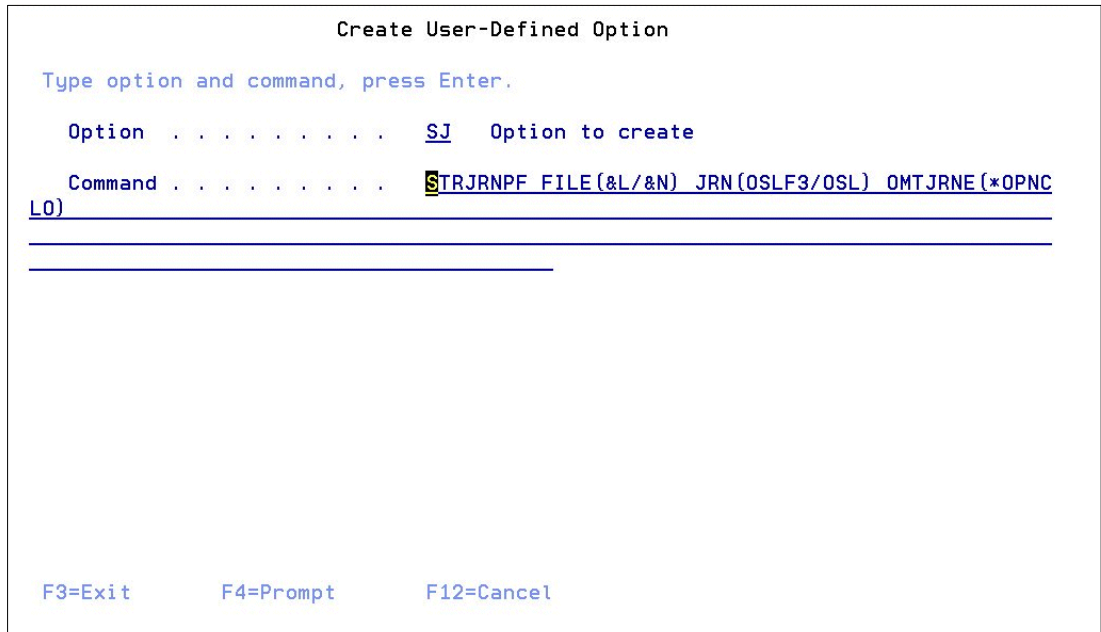
Figure 3-3   Creating a user-defined option

6. You return to the list of user-defined options. Press F3.

7. You return to the PDM menu. Select option `2` (Work with objects).

8. On the Work with Objects display, complete these tasks:

   a. Enter the library that contains the files to journal for example, `OSLOMF3`.

   b. Under object, set:

      - Name: `*ALL`
      - Type: `*FILE`
      - Attribute: PF-DTA

   Press Enter.

9. On the next display, type `SJ` next to each file that you want to journal. To journal *all* physical files, type `SJ` on the first line and press F13.

3.2.4 Stored procedures and SQL

You can easily run the stored procedures and other SQL commands by using Client Access. However, you can cut and paste the commands into an iSeries SQL session if required.

Stored procedures

Stored procedures link Java components with RPG programs. There are several of these procedures that must be created using the SQL scripts in the /OrderManagement/Stored Procedures folder. To create them, use the Client Access SQL tool, which is in iSeries Operations Navigator.

To run SQL scripts, follow these steps:

1. Open the **/Ordermanagement/Stored Procedures** folder in Windows Explorer and double-click the script you want to run.

2. Sign on as a valid user, and the script opens.

3. Select **Connection-> JDBC Setup** (ODBC setup pre-release V4R5).

4. The Client Access Express ODBC Setup (32-bit) window (Figure 3-4) opens. Click the **Format** tab.

5. Ensure that the system naming convention (*SYS) is selected.
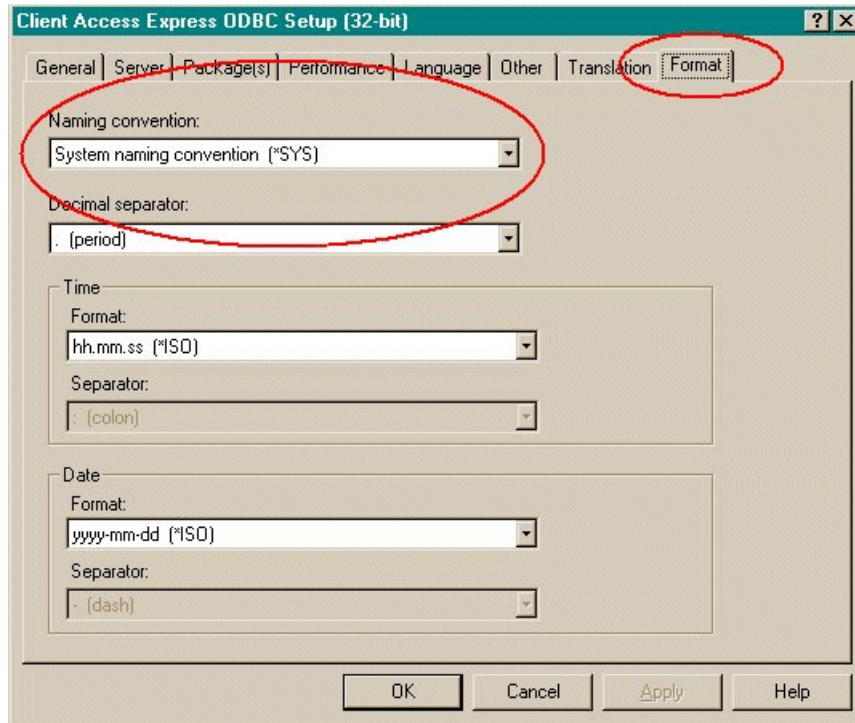
Figure 3-4   Client Access setup for the naming convention

6. Click the **Translation** tab.

7. Ensure that Translate CCSID 65535 is set.

8. Click the **Server** tab (Figure 3-5).

9. Ensure that the correct library list is set. Set the default libraries as:

```
,oslomf3, oslomd3, osld1f3, osld2f3, oslslf3, oslglf3
```

Note: Do not forget to include the initial comma (**,**) in the list.
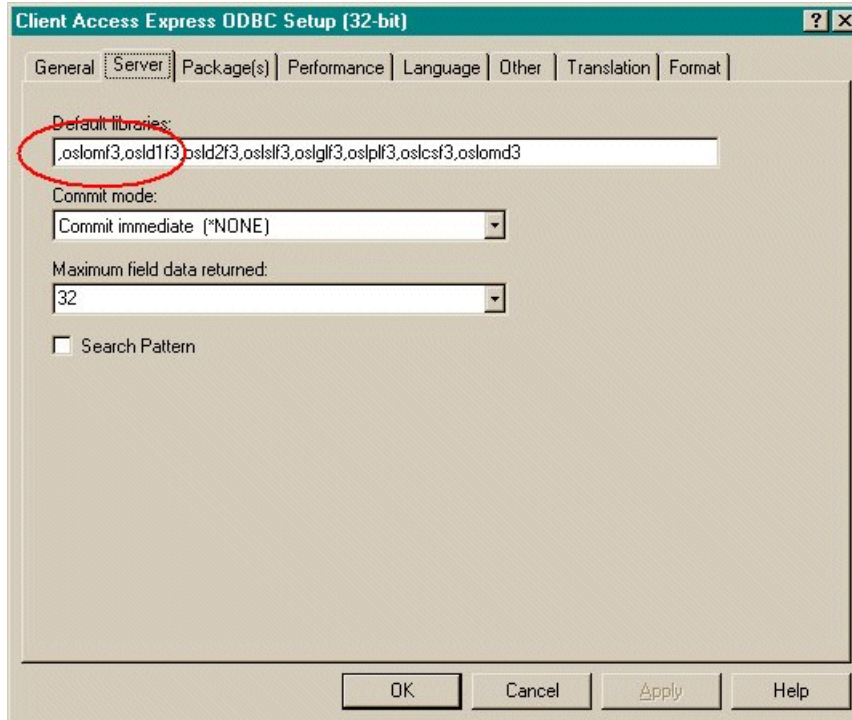
Figure 3-5   Client Access server setup

10.After you make all necessary changes, click **Apply** and then click **OK**.

As an alternative, you can perform these steps:

1. Open Operations Navigator.
2. Sign on as OMUSER.
3. Expand the **iSeries**.
4. Right-click the database and select **Run SQL Scripts**.
5. Open the **/OrderManagement/Stored Procedures** folder.
6. Double-click the script to select it.
7. Change the parameters as indicated in the previous set of instructions.

The scripts all start with a DROP PROCEDURE instruction. This removes any previously installed stored procedures. However, this does not run the first time the scripts start. In this case, position the cursor on the second line of the script. Then select **Run-> From Selected**.

Other SQL

Some static data must be set up. You achieve this by running the SQL scripts in the SQL Scripts folder. Use the Client Access SQL tool as explained in the previous section and run all the scripts in this folder.

3.2.5 Java Message Service

JMS is used to send messages from the server to the client. It is used when, for example, there are supply problems or the order is suspended. It is also used to allow complex time consuming steps to be completed asynchronously, for example when the order is written to System21.

Ensure that WebSphere is started.

The prerequisites include:

► WebSphere 3.5.5
► MQSeries 5.2
► MA88 MQSeries classes for Java and Java Message Service (requires V4R5 or later)

Setting up MQSeries

This section explains how to configure MQSeries on the iSeries to be used by call.connect. Then it explains how to map the MQSeries queues onto JMS queues.

Creating a queue manager

First you must create a queue manager as explained here:

1. If the subsystem QMQM is not started (check by using the Work with Subsystem (WRKSBS) command), enter the following command:

   STRSBS QMQM/QMQM

2. Enter the Work with Message Queue Manager (WRKMQM) command.

3. Press F6 (Create queue manager).

4. On the Create Message Queue Manager display (Figure 3-6), complete the following fields:

   – Message Queue Manager Name: CALLCONNECTQM
   – Text 'description': 'Call Connect Queue Manager'
   – Undelivered Message Queue: SYSTEM.DEAD.LETTER.QUEUE

   Press Enter to create the message queue manager.

```
                     Create Message Queue Manager (CRTMQM)

 Type choices, press Enter.

 Message Queue Manager name . . .    CALLCONNECTQM

 Text 'description' . . . . . . .    Call Connect Queue Manager

 Trigger interval . . . . . . . .    999999999    0-999999999
 Undelivered message queue . . .    SYSTEM.DEAD.LETTER.QUEUE

 Default transmission queue . . .    *NONE

 Maximum handle limit . . . . . .    256          0-999999999
 Maximum uncommitted messages . .    10000        1-10000
 Default Queue Manager . . . . .    *NO          *YES, *NO




                                                                      Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 3-6   Create Message Queue Manager display

5. Start the queue manager.

6. On the Start Queue Manager display, enter option 14 next to CALLCONNECTQM and press Enter.

7. Wait until the Status changes to *Active*. Then press F5 to refresh the display.

> **Note:** You need to start the queue manager before you can create any queues.

Creating a queue

Now create a queue as explained here:

1. On the Start Queue Manager display, type 18 next to CALLCONNECTQM and press Enter.

2. The Work with MQM Queues display appears. Press F6 to create the queue.

3. On the Create MQM Queue display (Figure 3-7), press F9 to display all parameters.

   a. Complete the following fields as shown here:

      - Queue Name: SALESORDER.QUEUE
      - Queue Type: *LCL
      - Text 'description': Sales Order Queue
      - Default Message Persistence: *YES

```
                    Create MQM Queue (CRTMQMQ)

 Type choices, press Enter.

 Queue name  . . . . . . . . . . .     SALESORDER.QUEUE

 Queue type  . . . . . . . . . . .     *LCL            *ALS, *LCL, *MDL, *RMT
 Message Queue Manager name  . . . >   CALLCONNECTQM

 Replace  . . . . . . . . . . . . .    *NO             *NO, *YES
 Text 'description'  . . . . . . . .   'Sales Order Queue'

 Put enabled  . . . . . . . . . . .    *YES            *SYSDFTQ, *NO, *YES
 Default message priority . . . .      0               0-9, *SYSDFTQ
 Default message persistence  . .      *YES            *SYSDFTQ, *NO, *YES
 Process name . . . . . . . . . . .    *NONE

 Triggering enabled . . . . . . .      *NO             *SYSDFTQ, *NO, *YES
 Get enabled  . . . . . . . . . . .    *YES            *SYSDFTQ, *NO, *YES
 Sharing enabled  . . . . . . . . .    *YES            *SYSDFTQ, *NO, *YES
                                                                      More...
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 3-7   Create MQM Queue display

   b. Press Enter to create the message queue.

Granting authority

Next, you must grant authority for both the queue manager and queue:

1. Enter the Grant MQM Object Authority (GRTMQMAUT) command for your queue manager.

2. On the Grant MQM Object Authority display (Figure 3-8), follow these steps:

   a. Complete the fields as shown here:

      - Object Name: CALLCONNECTQM
      - Object Type: *MQM
      - User Names: *PUBLIC (see note below)
      - Authority: *ALL
      - Message Queue Manager Name: CALLCONNECTQM

   b. Press Enter to grant authority.

```
                    Grant MQM Object Authority (GRTMQMAUT)

 Type choices, press Enter.

 Object name  . . . . . . . . . . .    CALLCONNECTQM

 Object type  . . . . . . . . . . .    *MQM          *ALL, *Q, *ALSQ, *LCLQ...
 User names . . . . . . . . . . . .    *PUBLIC       Name, *PUBLIC
              + for more values
 Authority  . . . . . . . . . . . .    *ALL          *ALTUSR, *BROWSE, *CONNECT...
              + for more values
 Message Queue Manager name . . .      CALLCONNECTQM




                                                                       Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 3-8   Create MQM Object Authority display for your queue manager

3. Enter the GRTMQMAUT command again to grant MQM object authority to your queue.

4. On the Grant MQM Object Authority display (Figure 3-9), complete the following fields as shown here:

   – Object Name: SALESORDER.QUEUE
   – Object Type: *Q
   – User Names: *PUBLIC (see the following note)
   – Authority: *ALL
   – Message Queue Manager Name: CALLCONNECTQM

   Note: You do not need to give object authority to all users (*PUBLIC). But, you can give it only to the appropriate accounts. QEJB needs full authority to these objects and the user under whom the SalesOrderRecevier.sh runs. This user is currently OMUSER.

   Press Enter to grant authority.

```
                    Grant MQM Object Authority (GRTMQMAUT)

 Type choices, press Enter.

 Object name  . . . . . . . . . . .    SALESORDER.QUEUE

 Object type  . . . . . . . . . . .    *Q            *ALL, *Q, *ALSQ, *LCLQ...
 User names . . . . . . . . . . . .    *PUBLIC       Name, *PUBLIC
              + for more values
 Authority  . . . . . . . . . . . .    *ALL          *ALTUSR, *BROWSE, *CONNECT...
              + for more values
 Message Queue Manager name . . .      CALLCONNECTQM




                                                                       Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Figure 3-9   Grant MQM Object Authority display for your queue

Additional files needed

You need the following files to configure and run your message receiver:

- ▶ **JMSAdmin.sh**: JMSAdmin tool executable from QSH
- ▶ **JMSAdmin.config**: Configuration file used by JMSAdmin
- ▶ **SendTestMessage.sh**: Enables test messages to be sent

You can find these programs in the /OrderManagement/mq directory on the iSeries server. If these files are missing, map to the iSeries root directory using Windows Explorer, navigate to the mq directory, and copy the missing files.

You may need to edit the JMSAdmin.config file. You can do this by using Windows Explorer. Alter the line that starts with `PROVIDER_URL……` so that it points to the correct port number on your iSeries, for example:

```
PROVIDER_URL=iiop://localhost:900/
```

Creating the JNDI/MQ objects

Follow these steps to create the JNDI/MQ objects:

1. Use Qshell to create the objects. Enter the `STRQSH` command.

2. Change the directory, using `cd directory name`, to where you installed the JMSAdmin files, for example:

   ```
   cd /OrderManagement/Build/mq
   ```

3. Type the following command:

   ```
   JMSAdmin.sh
   ```

   Press Enter. The JMSAdmin tool creates the objects within the namespace stated in your JMSAdmin.config file.

4. Type the following commands in the order shown in your qsh session. Press Enter after you enter each command:

   Note: You must use the exact case as shown here.

   ```
   def ctx(geac)
   change ctx(geac)
   def qcf(CallConnectQueueManager) qmgr(CALLCONNECTQM)
   def q(SalesOrderQueue) qmgr(CALLCONNECTQM) queue(SALESORDER.QUEUE)
   end
   ```

You have now created your JNDI/MQ objects. Press F3 to exit Qshell.

Standard properties

Add the following properties to your standard.properties file either on the iSeries server or client:

```
MQ.queue.manager=geac/CallConnectQueueManager
MQ.queue=geac/SalesOrderQueue
messaging=1
```

You can also add "`fakeunusual=1`" if you want to test unusual messages through JMS or MQSeries.

Additional WebSphere configuration

Add the following argument to the command line arguments within your application server. This enables the Message Listener to start and close automatically with the enterprise beans.

```
-Dcom.ibm.ejs.sm.server.ServiceInitializer=
jba.rose.salesorder.messaging.SalesOrderListener
```

Testing the messaging

After call.connect is up and running, you can verify whether the messaging is working correctly by testing it as explained here:

1. Run the call.connect product.

2. Open a call.

3. Create an order.

4. Enter an order line with the quantity of 10.

5. This should create a JMS message that should be delivered back to the client. If the JMS message was not delivered, consult the error logs.

> **Note:** You can also test whether messaging is working by using QSH (your enterprise beans in WebSphere must be running). Type the following text into your QSH session:
>
> `. ./SendTestMessage.sh "now type your message here"`
>
> Press Enter.

6. Your SalesOrderListener should now receive your message. Check your call.connect logs to see your received message.

3.2.6 User profiles

This section explains how to set up user profiles on the iSeries server, in the call.connect user directory and within System21.

Job description

The call.connect library list is set within a job description OSLOMD3/OMJOBD that is supplied as already configured in the OSLOMD3 library. The library list should contain: OSLOMF3, OSLD1F3, OSLD2F3, OSLSLF3, OSLGLF3, OSLCSF3, OSLOMD3, OSLOEP3, OSLINP3, IPGAMF4, IPGCFF4, IPGAMP4, OSLOIF4, OSLOID4, OSLWTP3, OSLGLD3, OSLSLD3, IPGCFP4, and OSLCSD3.

However, if you must use other .connect products or other products that use a workflow, change the job description to include OSLWFF3 in its library list.

Creating iSeries user profiles

You must create iSeries user profiles for *all* call.connect users. To do this, sign on as QSECOFR and create the profiles as required.

In addition, you must create a special user profile, normally OMUSER with password OMUSER, to be used by JDBC connections. Be sure to use the job description within this user profile. Set the profile with QUSER authority. Then set the CCSID accordingly as follows:

- ▶ United Kingdom: CCSID = 285
- ▶ United States: CCSID = 37
- ▶ Other countries (regions): Set as required

Sign on as `QSECOFR` and create the profile as shown here:

```
CRTUSRPRF USRPRF(OMUSER) TEXT('call.connect user profile') JOBD(OSLOMD3/OMJOBD) CCSID(285)
```

Setting up call.connect users in the XML user directory

You must set up all call.connect users on the server in the XML file /OrderManagement/log/UserDirectory.xml, with the same user IDs as those created on the iSeries. Open this XML file in a notepad and copy the following section:

```
<entry dn="cn=John Smith, ou=eCommerce, ou=Studley,o=Geac,c=UK">
     <objectclass>
       <oc-value>top</oc-value>
       <oc-value>person</oc-value>
       <oc-value>organizationalPerson</oc-value>
       <oc-value>inetOrgPerson</oc-value>
       <oc-value>ePerson</oc-value>
       <oc-value>s21User</oc-value>
     </objectclass>
     <attr name="cn"><value>John</value></attr>
     <attr name="sn"><value>Smith</value></attr>
     <attr name="uid"><value>Operator1</value></attr>
     <attr name="userpassword">Teacup</value></attr>
     <attr name="mail"><value>-none-</value></attr>
     <attr name="s21username"><value>-none-</value> </attr>
   </entry>
```

Note the following explanation:

cn   Common name
sn   Surname
uid   User ID for call.connect
userpassword   Password for call.connect

You can edit the entries under dn (distinguished name), but they should be the same for all entries:

ou   Organizational unit
o   Organization
c   Country (region)

Amend the names and profiles as required. Most of this information is for memo purposes. The only mandatory lines are the user ID and the password (*uid* and *userpassword*). The user ID should not have spaces in it because this also needs to be a valid iSeries sign on.

Setting up System21 user profiles

All the users defined as call.connect users in the XML user directory must have a System21 user profile. They must also be defined as operators within the Order Management application.

To set up a System21 user profile, follow these steps:

1. Sign on to the iSeries as a user with access to System21 common functions. When delivered, QPGMR has the required authority.

2. Enter the following command to access the main functions:

   `STRIPGCF`

3. Select option 1 from the menu and enter the user ID to maintain.

4. Press F17 and type option 1 next to each menu that begins with *OM*. Press Enter.

5. On each display, press F15 to authorize the user to all options on that menu.

6. After all displays are presented, press F12 to return to the main user profile maintenance display.

7. Select F19 and type 1 next to the OM 03 application. Press Enter.

8. Type 1 next to all companies that are valid for this user. Press Enter.

For more information, see Chapter 1 in the Geac System21 *Administration Functions Active Enterprise Framework* manual.

Setting up an operator

This section explains how to set up an operator. It assumes that the user is authorized to the OM menus as explained earlier and that the OM company profile is already configured.

1. Sign on to the iSeries server using the profile OMUSER.

2. Enter the command:

   AM3

3. On the Option line, enter:

   /OMM

   This displays the OMM menu.

4. On the OMM menu, select option 1.

5. Add or maintain the operator codes as required. Now all users should have an iSeries profile, an XML profile, and a System21 profile. They should also be set up as operators. Figure 3-10 shows an example display of setting up an operator profile.

```
OM005          Z1 - UK Demonstration Company       User: GBJBAJH0     21/12/01
                                                                       12:22:05
Operator Profiles - Maintain

Operator Code............... 0001   New Operator

Operator Name............... Operator 0001_____
Operator Extension Number.... 375_
Associated User Profile Id... OP001_____

Operator skill level......?.. 1___   1  Trainee

Minutes/call................. _5     Minutes Rest/Hour........... _5

                Mon      Tue      Wed      Thu      Fri      Sat      Sun
Working......    1        1        1        1        1        0        0
Start Time...  9:00     9:00     9:00     9:00     8:30     0:00     0:00
End Time ....  17:30    17:30    17:30    17:30    15:45    0:00     0:00


_____
F3=Exit  F4=Prompt  F5=Refresh   F8=Update            F12=Previous
F13=Absences  F21=Text
```

Figure 3-10   Operator setup display

For more information, see the *System21 Setup for call.connect* manual.

3.2.7 System21 data set up

System21 data setup is covered in the *Basic System21 Setup for call.connect* manual. You must complete this setup in order for call.connect to operate successfully. This section outlines the steps to configure the Order Management module (see Table 3-3). These steps assume that other parts of System21, such as customers, items, depots, etc., are already setup.

Table 3-3   System21 Order Management setup

Step	Action	Completed
1	Create a search family code	
2	Create an Order Management company profile	
3	Create buying lists	
4	Create buying list rules	
5	Create operator profiles	
6	Create teams	
7	Create OM customer details	
8	Create OM customer contacts	
9	Create outgoing call schedules	
10	Create item sales details	
11	Create supply rules and policies	
12	Create supply points	
13	Create customer and item attributes	
14	Generate call lists	

3.2.8 Java Web Start

This section provides instructions for a straightforward way to configure a new iSeries. Two HTTP Servers are available at different releases of the operating system:

► IBM HTTP Server for AS/400 (original): For V4R4 and earlier
► IBM HTTP Server for iSeries (powered by Apache): For V5R1 and later

V4R5 supports both original and powered by Apache.

Ensuring that the HTTP administrative server is ready

The following steps are the same for both servers:

1. Verify that the HTTP Server software is loaded. On the iSeries, type the following command:

   ```
   GO LICPGM
   ```

2. On the display that appears, select option 10 (Display). Check the suffixes. Look for **DG1**. If you do not find it, load it before you proceed to the next step.

3. Start the HTTP Server. On the iSeries, run the command:

   ```
   STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
   ```

4. Verify that the server is running. Enter the following command:

```
WRKACTJOB
```

5. Verify that ADMIN jobs are running in the QHTTPSVR subsystem.

Configuring the HTTP Server on the iSeries

To configure the HTTP Server on the iSeries server, follow these steps:

1. In a Web browser, type:

```
http://<iSeries>:2001
```

Enter your iSeries server name for `iSeries`. This is the port on which the administration of HTTP is listening.

2. Sign on as `QSECOFR`.

3. The AS/400 Tasks page appears as shown in Figure 3-11. In this example, the pages are for an iSeries called *Needjava*. Select **IBM HTTP Server for AS/400**.
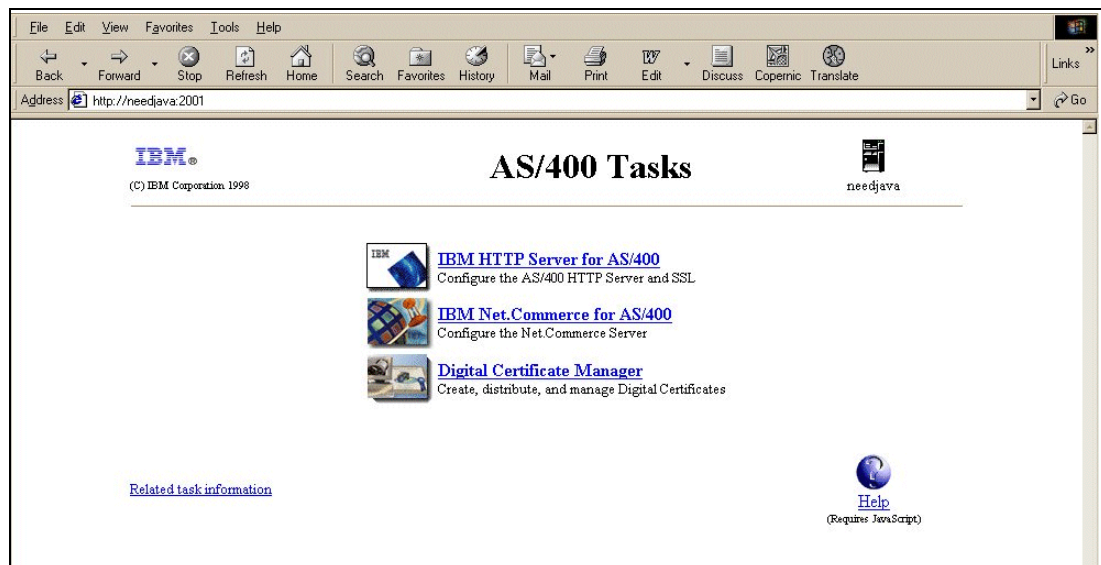
Figure 3-11   HTTP Server configuration

4. The IBM HTTP Server for AS/400 page appears as shown in Figure 3-12. Select the **Configuration and Administration** icon on the left.
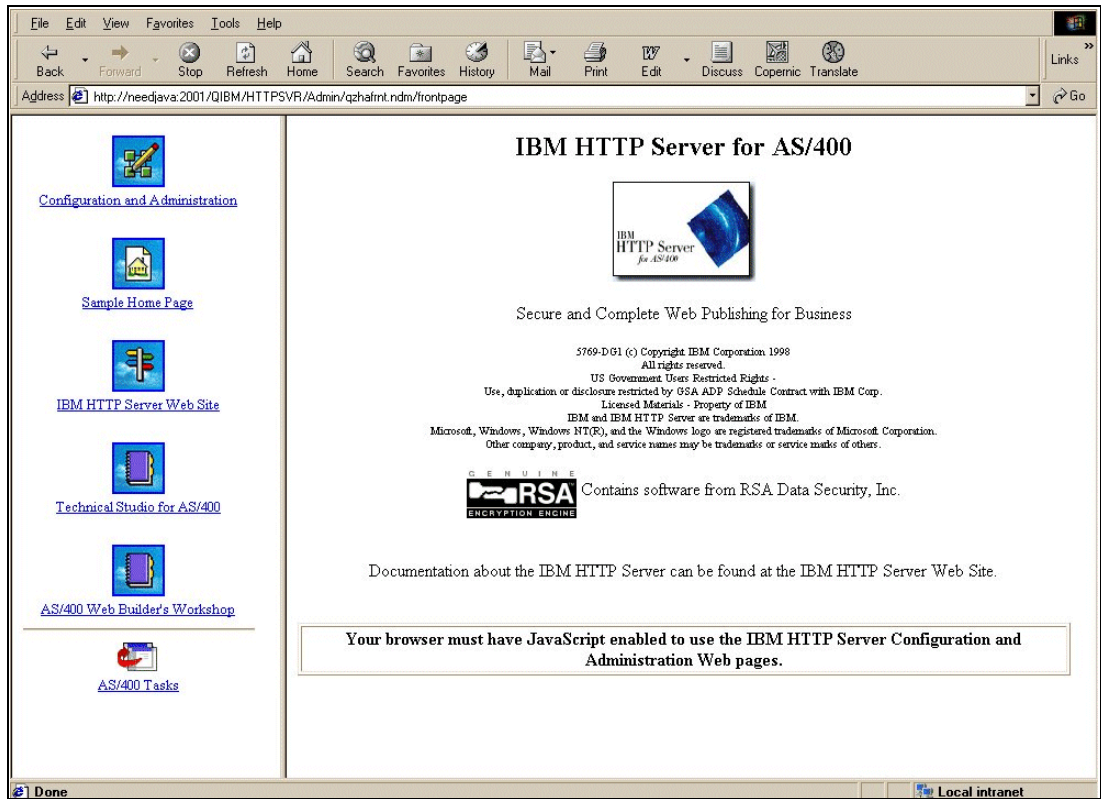
Figure 3-12   HTTP Server configuration

From this point forward, the instructions are different depending on whether you are using IBM HTTP Server (original) or IBM HTTP Server (powered by Apache).

Configuring an HTTP Server for iSeries (original)

To configure an IBM HTTP Server (original) at V4R5 or earlier, follow these steps. First, you should see the IBM HTTP Server Configuration and Administration page (Figure 3-13). Click the **Configurations** option on the left.
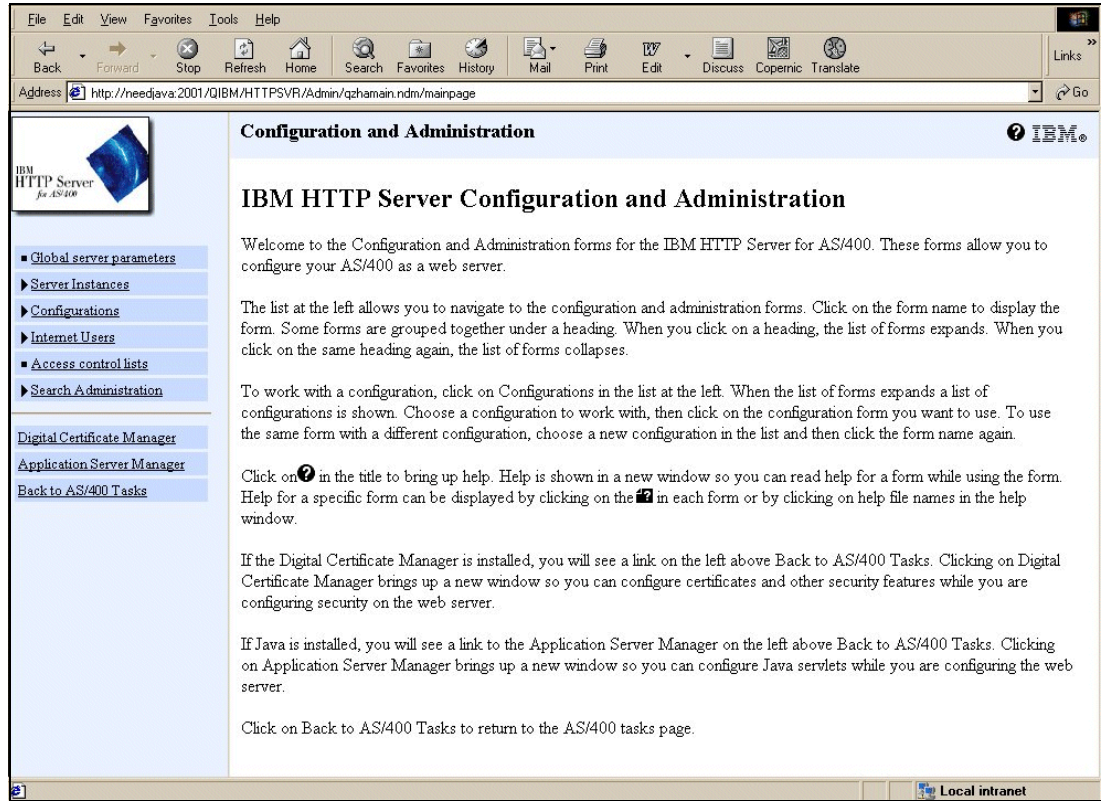
Figure 3-13   IBM HTTP Server Configuration and Administration page

Next you need to create a new configuration and a new instance, based on the IBM-supplied configurations and instances. This is explained in the following sections.

Creating a configuration

Follow these steps to create a configuration:

1. Under the Configurations option, select the **Create configuration** link.

2. The Create configuration page appears (Figure 3-14). Follow these steps:

   a. Insert the name of the configuration, such as the iSeries name.
   b. Select the **Create based on existing configuration** option and enter `CONFIG`.
   c. Click **Apply**.
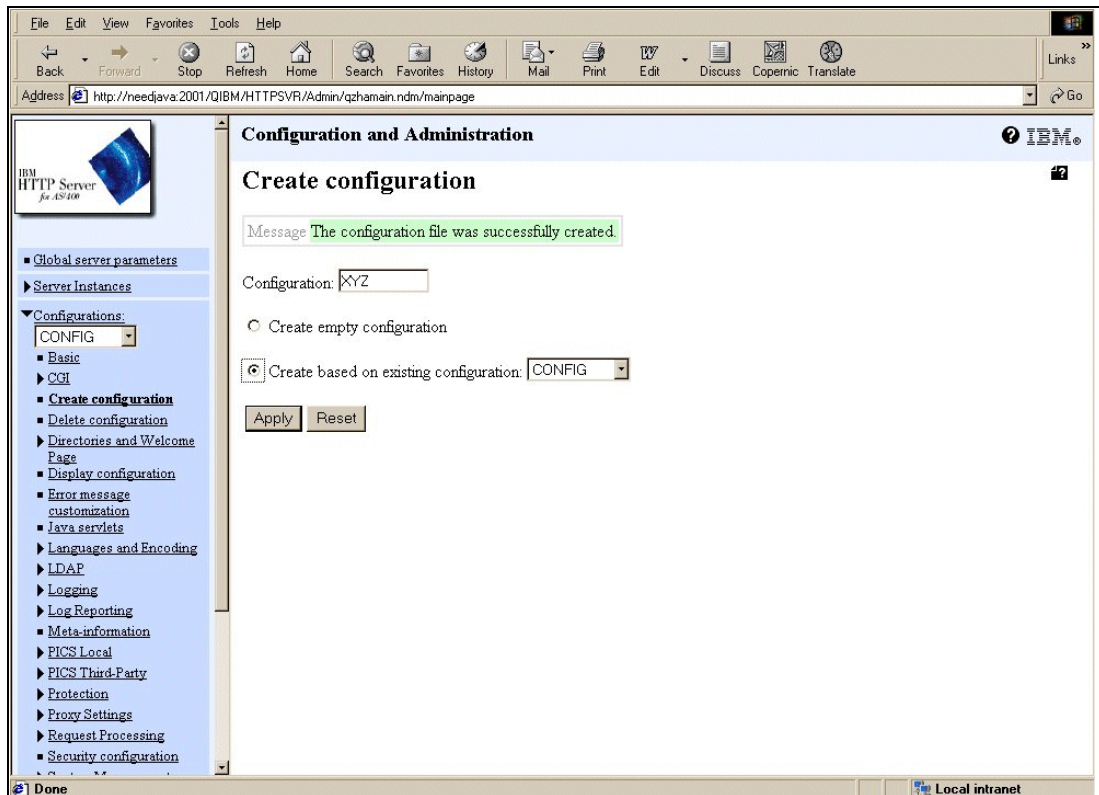
Figure 3-14   Create configuration page

3. Select the **Request Processing** option on the left and select the **Request routing** link.

4. The Request routing page (Figure 3-15) appears. Insert the following actions, templates, and replacement file paths. This allows the system to find actual files when the user logs on to call.connect. Be sure to click the **Insert after** radio button:

Action	URL template	Replacement file path
Pass	/callconnect/	/ordermanagement/client/index.html
Pass	/callconnect/*.gif	/ordermanagement/client/*.gif
Pass	/callconnect/callconnect.jnlp	/ordermanagement/client/callconnect.jnlp
Pass	/callconnect/*.jar	/ordermanagement/client/*.jar
Pass	/callconnect/setup.exe	/ordermanagement/client/setup.exe
Pass	/callconnect/help/	/ordermanagement/client/help/introduction.htm
Pass	/callconnect/help/*.gif	/ordermanagement/client/help/*.gif

Click **Apply**.
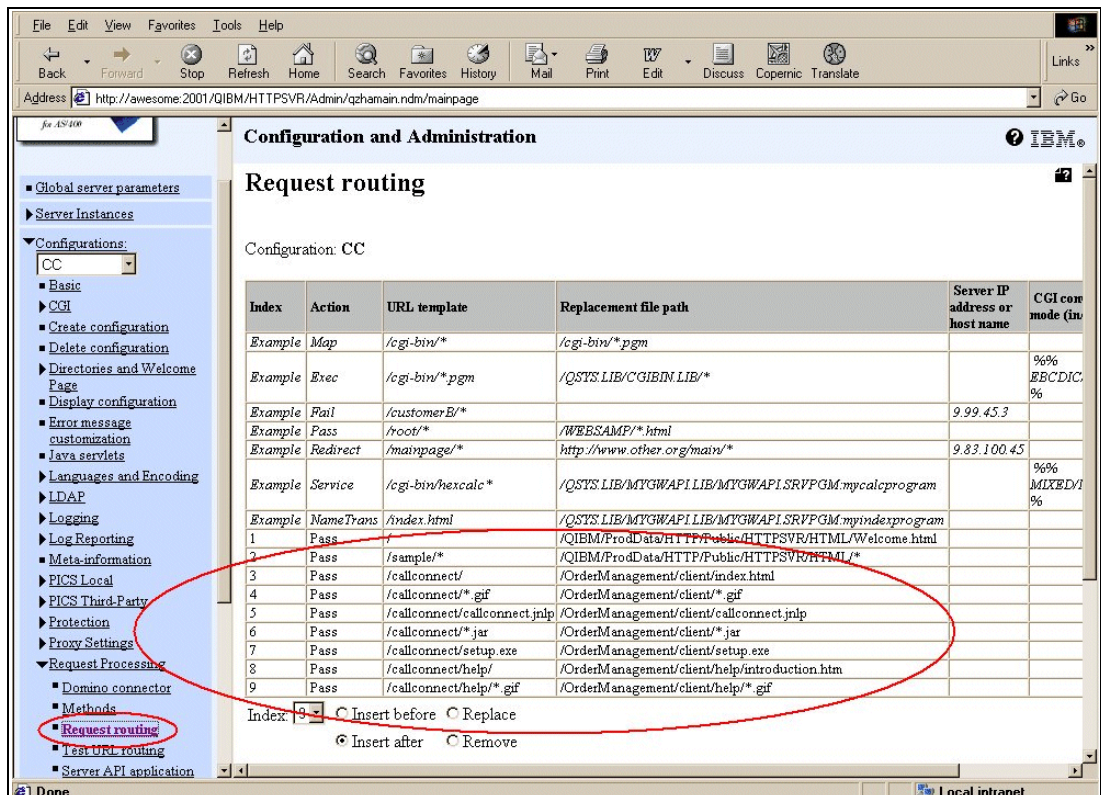
Figure 3-15   Request routing page

5. Select the **Languages and Encoding** link on the left. Then select **MIME types**.

6. The MIME types page (Figure 3-16) appears. Enter the following information for the system to recognize the .jnlp and .htm extensions. Be sure to click the **Insert after** radio button:

– **.jnlp extension**

- File extension: `.jnlp`
- Content type/subtype: `application/x-java-jnlp-file`
- Encoding: `8 bit`
- Quality: `1.0`

– **.htm extension**

- File extension: `.htm`
- Content type/subtype: `text/HTML`
- Encoding: `8 bit`
- Quality: `1.0`
- Character Set: `ISO-8859-1`

Click **Apply**.
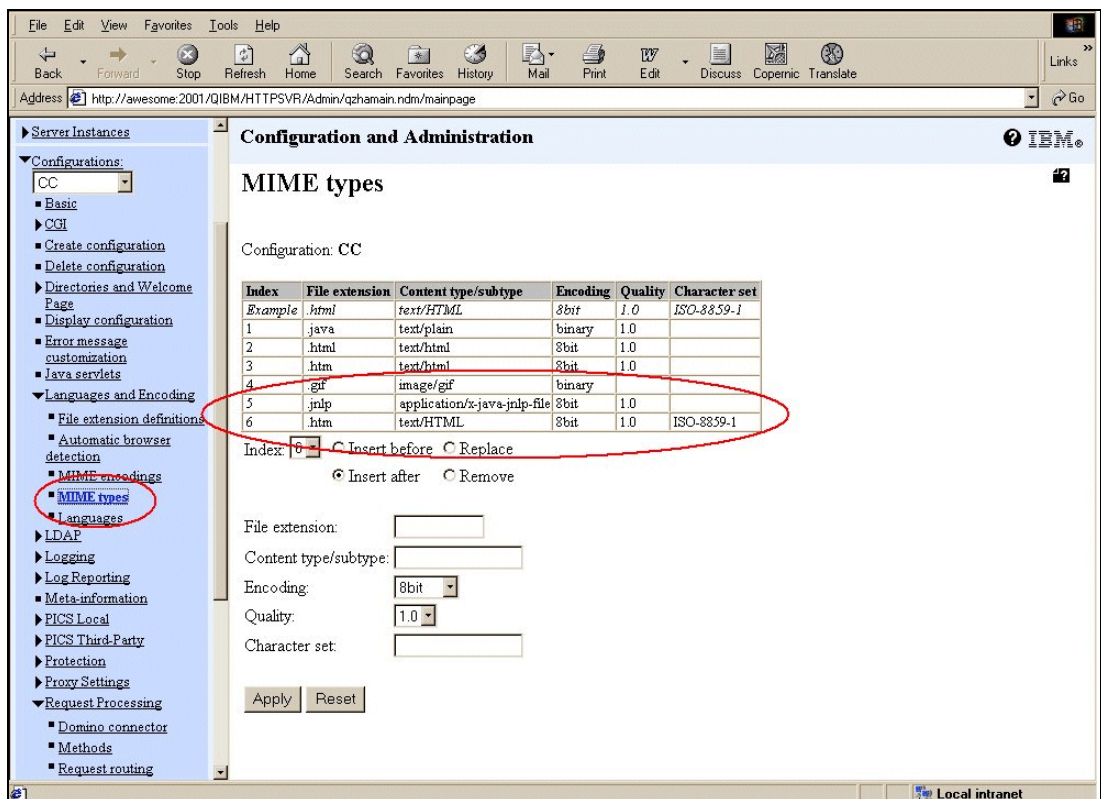
Figure 3-16   MIME types page

Creating an instance

Now you must create a server instance:

1. On the left-hand panel, select **Server Instances** and then the **Create server instance** link.

2. The Create server instance page (Figure 3-17) appears. Follow these steps:

   a. Enter the server instance name, such as the iSeries name.
   b. Enter the configuration that was just created.
   c. Click **Create**.
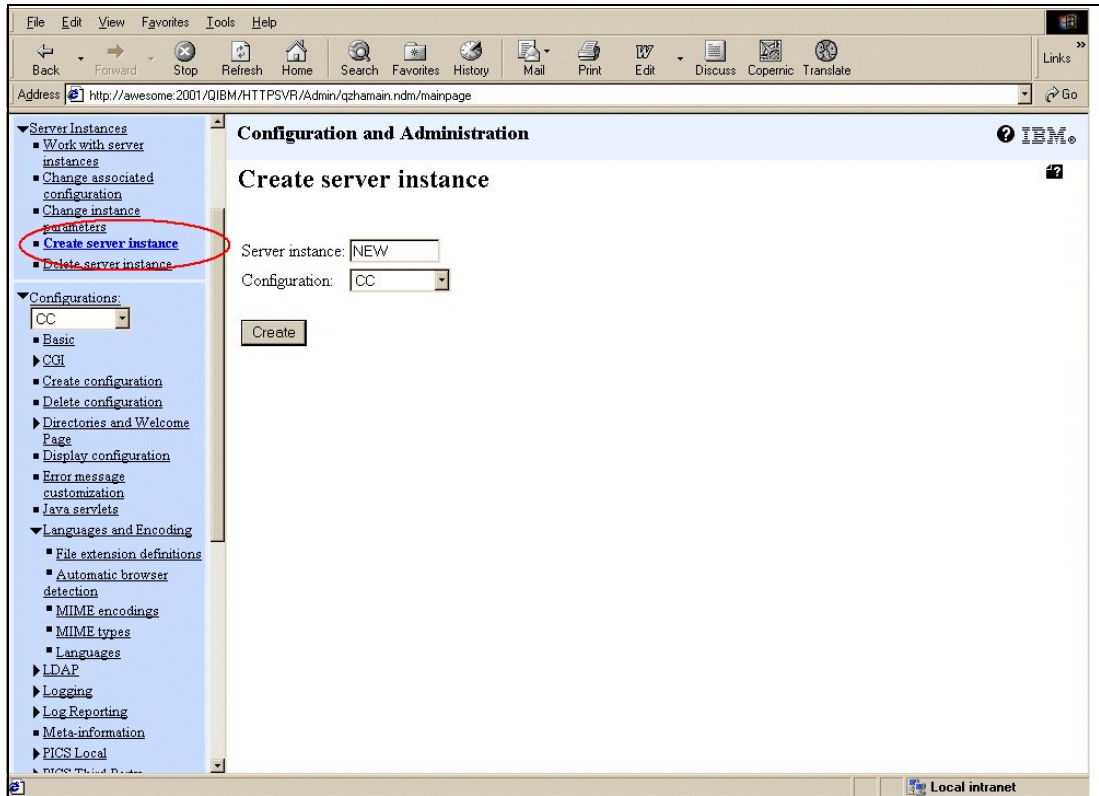
Figure 3-17   Create server instance page

You see the message "`Message The server instance was successfully created`".

In WRKACTJOB SBS(QHTTPSVR), you now see ADMIN jobs and jobs with the name of the instance that was just created. All should be in *wait* states.

Now follow the steps in "Configuring CallConnect.jnlp" on page 51.

Configuring an HTTP Server for iSeries (powered by Apache)

This section explains how to configure an HTTP Server for iSeries (powered by Apache) from the point where Configuration and Administration was selecteda:

1. To begin, the Manage HTTP Servers page (Figure 3-18) should be displayed. Select **Administration** (if it is not already selected) from the options across the top of the page.

2. Under HTTP Server Creation in the left-hand panel, select **Create HTTP Server**.
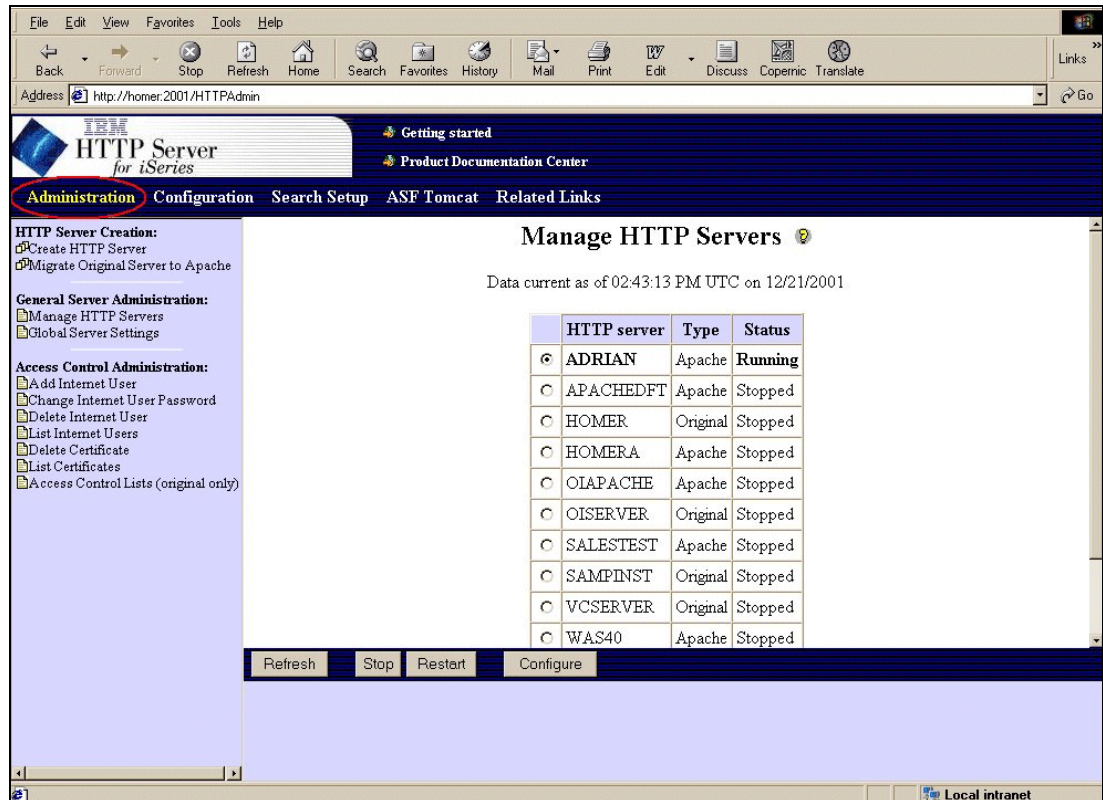
Figure 3-18   Manage HTTP Servers page

3. The Create HTTP Server page (Figure 3-19) appears. Fill in the following information when prompted on each page that appears next. Click **Next** on each page to advance.

   – Type of server: **HTTP Server (powered by Apache) –** *recommended*
   – Server name: Give a suitable name. Could be the iSeries name.
   – Configure base on existing: **No**
   – Server root: Select the default
   – Document root: Select the default
   – IP address and TCP/IP port: Select the default
   – Logging: **Combined**

   Click **Finish** on the last page.
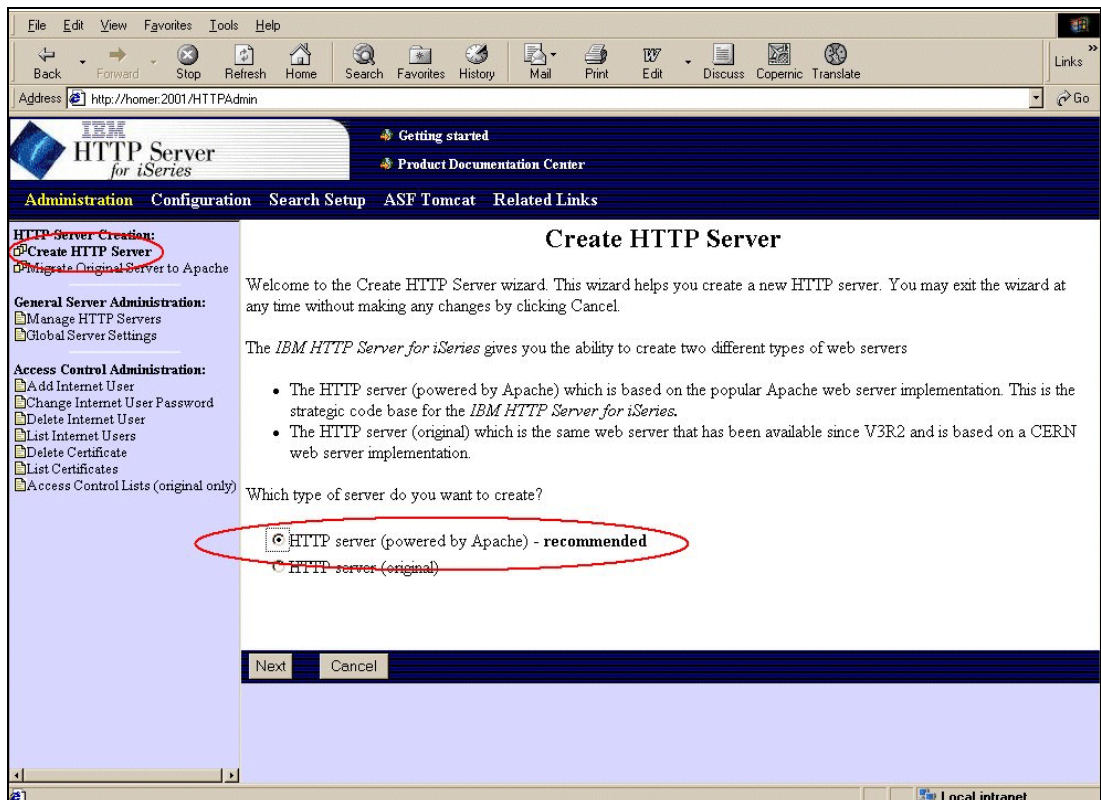
Figure 3-19   HTTP Server configuration 2 (Apache)

4. At the top of the page, select **Configuration**.

5. Under Configuration structure, verify that **HTTP Server Name global settings** appears in bold.

6. You should see the *HTTP Server Name* (HOMERA) global settings page (Figure 3-20). Under Web Site Definition, select **Serve New Directory wizard**. Click **Next**.
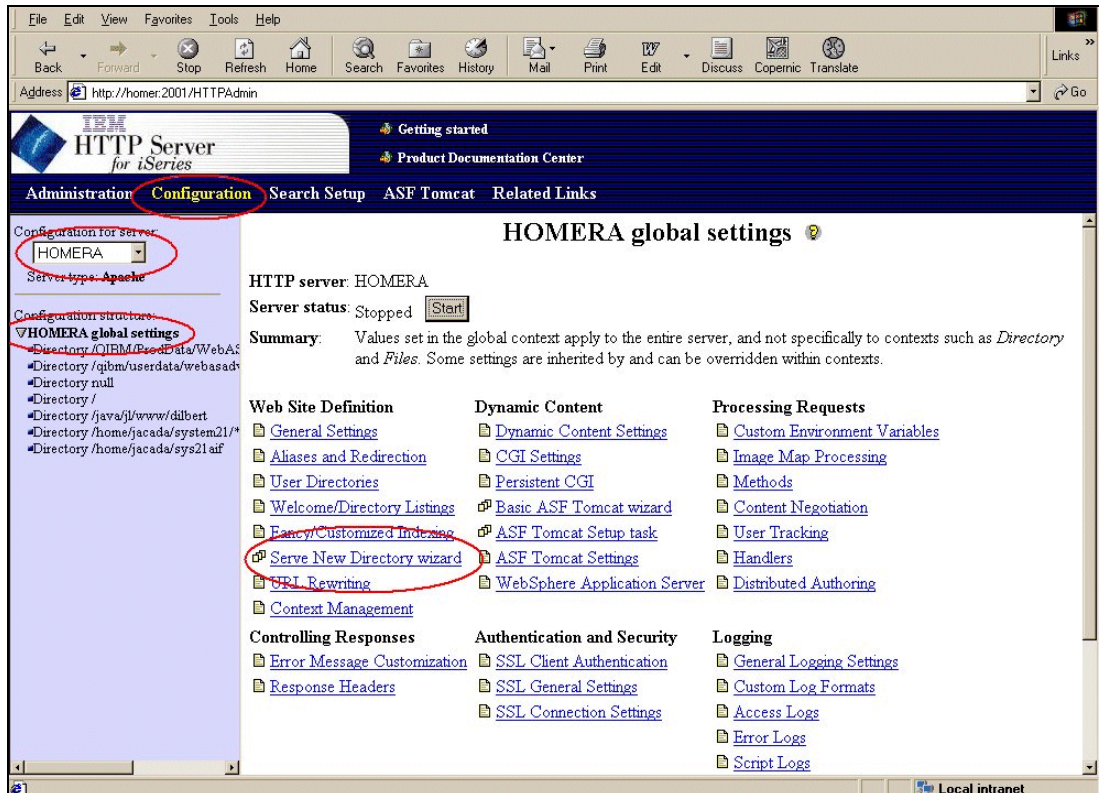
Figure 3-20   HTTP Server name (HOMERA) global settings page

7. On the Serve New Directory wizard, complete the following tasks:

   a. Select **Static Web Pages and Files**.

   b. For Directory to Serve from, enter the client directory, such as
      `<iSeries>/OrderManagement/client`.

   c. For Alias, enter the name required on the browser, such as `/callconnect/`.

   d. Click **Finish**.

8. Click the **OrderManagement/client** directory under Configuration structure to select it.

9. On the Director /OrderManagement/client page (Figure 3-21), under Processing Requests, click **Content Negotiation**.
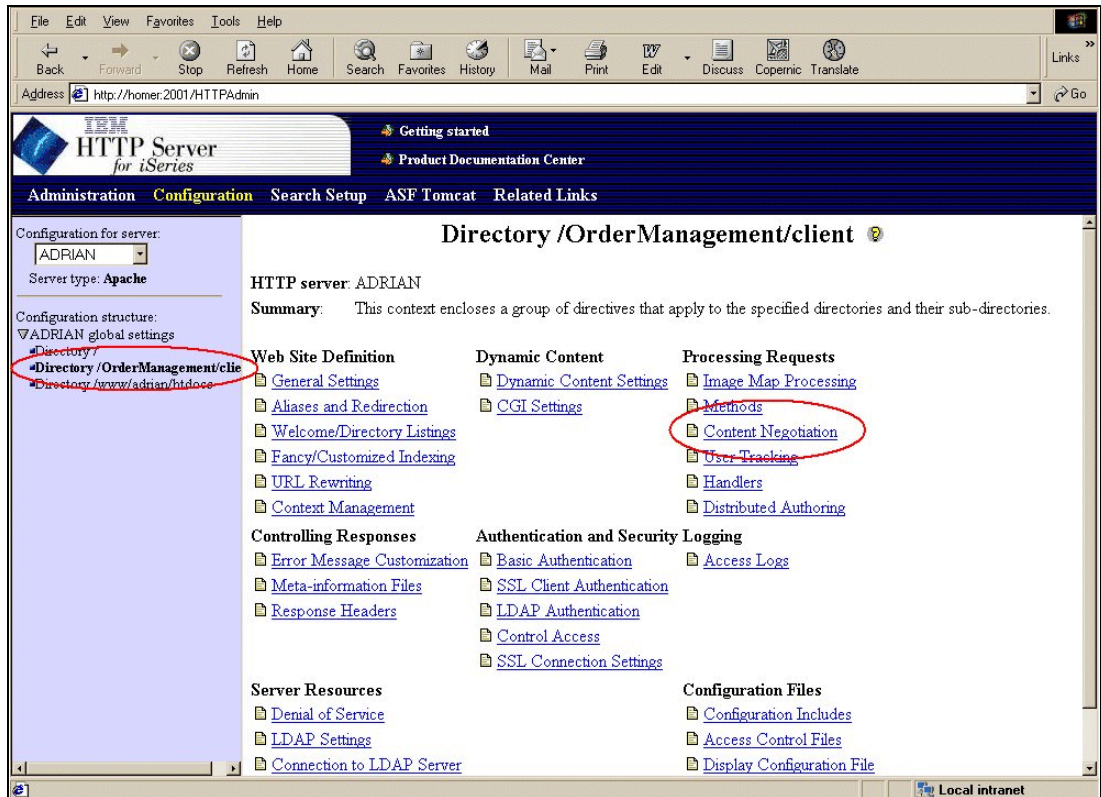
Figure 3-21   Directory /OrderManagement/client page

10.On the Content Negotiation page, scroll about half way down the page to **Additional Meta (MIME) information** (Figure 3-22). Then follow these steps:

a. Click **Add**.

b. Enter the following information:
   - File Extension: `.jnlp`
   - Type: `content-type`
   - Value: `application/x-java-jnlp-file`

c. Click **Add**.

d. Enter the following information:
   - File Extension: `.jar`
   - Type: `content-type`
   - Value: `application/x-zip-compressed`

e. Click **Add**.

f. Click **Apply**.
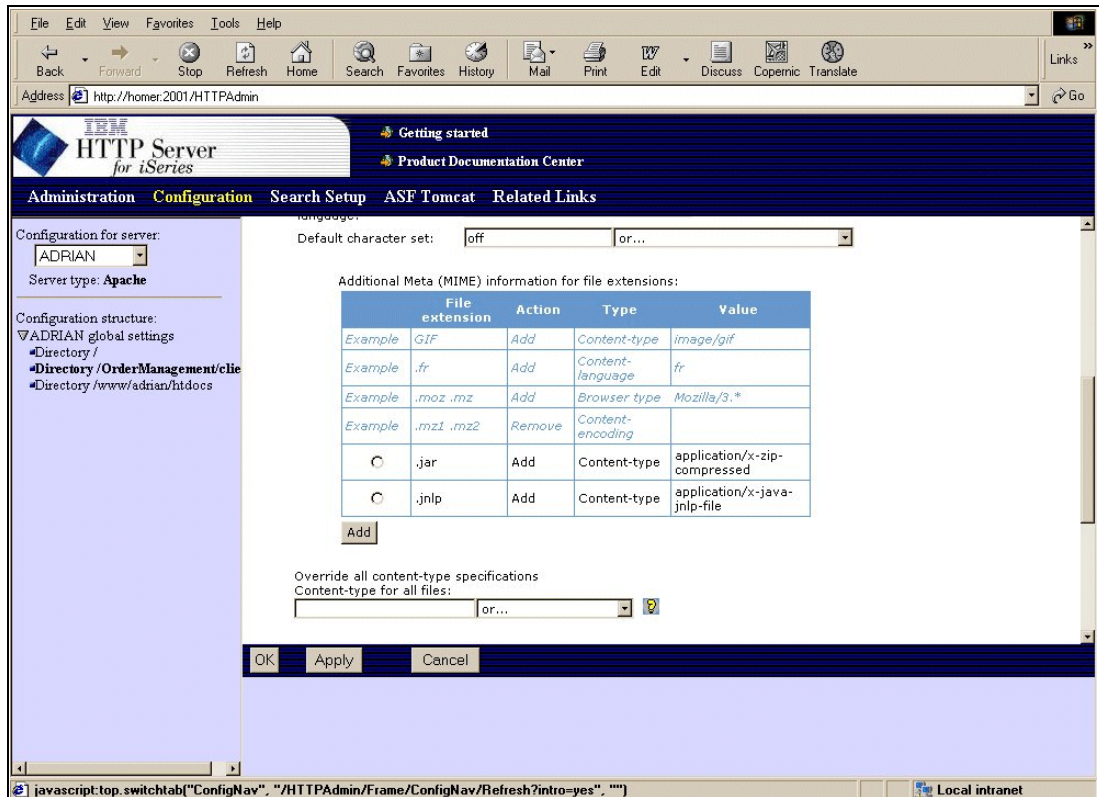
Figure 3-22   Additional Meta (MIME) information for file extensions section

11. At the top of the page, select **Administration**.

12. Click the **Manage HTTP Servers** link in the left-hand panel.

13. On the Manage HTTP Servers page (Figure 3-18 on page 47), stop the HTTP Server and then start it.

> **Attention:** For increased security, configure the aliases in a similar way to those suggested for the original server.

Configuring CallConnect.jnlp

In Windows Explorer, modify the /ordermanagement/client/callconnect.jnlp file as follows to reflect the alias that you previously set:

```
codebase="http://<iSeries>/callconnect/"
```

Change the server name. Also change the following line to point to the correct iSeries and port number:

```
<property name="java.naming.provider.url" value="iiop://<iSeries>:900"/>
```

The default port number for the default instance of WebSphere is 900.

Configuring JWS.bat

After you set up the HTTP Server, configure JWS.bat as explained here:

1. In Windows Explorer, enter your iSeries name in the following line in the /ordermanagement/client/JWS.bat file:

```
"C:\Program Files\Java Web Start\javaws" http://<iSeries>/callconnect/CallConnect.jnlp
```

2. Launch Java Web Start for the first time. In a Web browser, go to:

   `http://<iSeries>/callconnect/`

3. Click **Install Java Web Start**.

4. Select **Run this program from its current location**.

5. Select **Yes** to install and run setup.exe from the iSeries.

6. After this runs, select **Launch call.connect** to run the application. On the first time through, when you reach the prompt "Do you want to install and run: call.connect?", select **Start**.

7. Enter the following information:

   – Default User: User for call.connect
   – Default Password: Password for user
   – S21 Company: XX (the AR company to be used)

   This creates a desktop icon to run call.connect. Future launches of call.connect automatically access the latest updates to the client software. Furthermore, you only need to enter your user ID and password.

> **Note:** This user must be a valid user that is set up in the XML User Directory, defined to System21, and set up as an operator in Order Management.

3.2.9 Backup

After you have a satisfactory and working system, back up System21 and the integrated files system (IFS) objects. For System21 backup, see *Geac System21 Installation Guide.*

To back up the IFS objects, use the SAV command on the iSeries. The following example shows how to back up the /OrderManagement folder to tape:

```
SAV DEV('/qsys.lib/tap01.devd') OBJ(('/OrderManagement'))
```

3.3 call.connect housekeeping

call.connect requires little housekeeping after you set it up and start running it.

3.3.1 Daily backups

Back up the System21 file libraries as usual, including library OSLOMF3.

The buying lists displayed in call.connect display items that were ordered in the last six weeks. To keep them up to date, run Generate Buying Lists, option 11/OMP, every night. You can set this up as an auto day, end job in the System21 Machine manager.

Certain data is cached in WebSphere so users may want to stop and start WebSphere daily or weekly as explained in the following sections.

3.3.2 Stopping WebSphere

This section offers helpful information for situations where the iSeries is powered down or you want to stop and start WebSphere overnight.

Under normal circumstances, the application (for example, SalesOrder) continues to run in WebSphere when the admin job QEJBADMIN is stopped. This means that to stop the system overnight, the easiest way is to end QEJBSBS by using the following command:

```
ENDSBS SBS(QEJBSBS) OPTION(*IMMED)
```

This is a standard, tested iSeries command. It has the advantage of ending the JMS jobs and any extra WebSphere instances. Users who want to end the subsystem more gently, may want to use *CNTRLD end with a time limit. If you do not specify a time limit, the subsystem will never end, because the JMS jobs keep running.

As an alternative for stopping and starting jobs, use XMLConfig as in a WebSphere configuration. However, keep in mind that this method has not yet been fully tested by Geac.

3.3.3 Starting call.connect

To start call.connect, assuming the SalesOrder application was left running, you must follow the steps to start WebSphere using either a default instance or other instances as explained in the following sections.

Starting WebSphere (default instance)

With a default instance, you simply start QEJBSBS by using the following command:

```
STRSBS SBSD(QEJB/QEJBSBS)
```

This starts QEJBADMIN, QEJBMNTR, and the SALESORDER application. It requires some time, possibly several minutes to start.

Starting WebSphere (other instances)

If you need to start more than one instance of WebSphere, use the following command:

```
CALL PGM(OSLOMD3/STRWSSVR) PARM(TEST)
```

This starts the test instance. You may start other instances while the default instance is still starting.

3.3.4 Restoring IFS objects

If for any reason you need to restore the IFS objects for call.connect, use the following command. Keep in mind that the system must have been previously backed up as recommended in 3.3.1, "Daily backups" on page 52:

```
RST DEV('/qsys.lib/tap01.devd') OBJ(('/OrderManagement'))
```

3.4 Troubleshooting

If you followed the installation instructions exactly, the installation should proceed smoothly. The primary cause of problems, particularly for those not experienced in these areas, is any errors in installation steps. Therefore, if the system does not start correctly, you must first carefully recheck all the installation steps.

You may also find the points in the following sections to be helpful. They are based on experiences gained from installations carried out over several months on different iSeries servers.

3.4.1 WebSphere node name

The node name in WebSphere is case sensitive. This is relevant when you start the console with AdminClient and when you specify an IIOP address for a JNDI lookup (for example, within the batch files which start client programs). It is also relevant to tools such as the XML export or import utility.

Usually the name is entirely in lowercase. However, on some systems, it is entirely in uppercase and could, in theory, be a mixture of both.

If you find that the console fails to start with such messages as "Could not get attributes" (similar to when the service/subsystem is not started), but the service is started, then the case of the name may be the problem.

You can verify the required name by using the following SQL to look at a WebSphere table:

```
select NAME
from EJSADMIN/NODE_TABLE
```

This shows the node name exactly as WebSphere wants it. Do not be tempted to change the contents of this table. It is liable to make things worse rather than better. (You need to have QSECOFR authority to read this table.)

3.4.2 Errors on starting the client

Verify that there are no tabs or spaces after any of the parameters in standard.properties, particularly if using the CCClient method where these are set up manually.

With data errors, once you sign on to the call.connect GUI, verify that advanced pricing is switched on for the company in use.

Also check that the search family code is set up in the OM company profile.

3.4.3 Errors when running the client

If pricing, tax, or discounting does not work correctly, check that the library list for the job description used by OMUSER is correct.

If amended data does not appear, for example an OM item specification that is newly setup, delete the .bl object in the log folder (see the following section). If you still have problems, stop and start the beans in WebSphere to update the cached data.

3.4.4 Cached data and .bl and .cd files

To make call.connect as fast as possible, data from more stable files is cached in WebSphere. If these files change, the new data will not be available to call.connect until the application server is stopped and restarted. The following information is cached:

► User profiles as held in /OrderManagement/log/UserDirectory.xml
► System21 company profiles

In addition, buying list and customer data is held in files in the /OrderManagement/log folder. These files are of type .bl and .cd respectively. These files are normally updated by the System21 options on menu /OMP, Generate Buying Lists and Call List Generation. You may delete the files if necessary, for example, for testing purposes.

3.4.5 Log files and debugging

There are log files for both the client and the server, where you can find output produced while the applications are running. For normal running, it is more efficient to minimize the output to these files. However, if an error arises without an obvious solution, it is a good idea to check these log files, which may have information on the cause of the error.

In addition, varying the logging level in the configuration files can change the amount of output to give more clues as to the cause of the problem. JMS and Pre Generate Buying Lists also have log files, which you may view, although the level cannot be changed.

Setting up logging on the client

This section explains how to set up logging on the client.

Java Web Start

Output from the client, if using Java Web Start, is directed to the file c:\Winnt\Profiles*<signed on user>*\callconnect.log.

To change the logging level, go to the file C:\Winnt\Profiles\<s*igned on user*>\log.cfg. Change the following line:

```
log4j.rootCategory=ERROR, basic
```

Possible values instead of ERROR are WARN and DEBUG. DEBUG is the highest level. For example, it produces the most output.

Manually configured client

If you are using a manual client configuration, then the logged output is in a file as directed in log.cfg, which is in the CCClient folder (see 3.5.3, "Manual client installation" on page 61). In the log.cfg file, find a line, as shown in the following example, to see where the logged output will be:

```
log4j.appender.basic.File=C:\\callConnect\\CCClient\\callConnect.log
```

In addition, the following line shows the logging level:

```
log4j.rootCategory=ERROR, basic
```

You may change ERROR to WARN or DEBUG to produce more output.

Server

The server has three log files for output. The first two, called *standard output* (stdout) and *standard error* (stderr), are defined in the application server in WebSphere. The logging level cannot be changed. Remember, if they are in WebSphere with an exclamation mark in front of them, for example !stdout, they are cleared every time WebSphere is restarted and logged information is lost.

The third output file is defined in the /OrderManagement/cfg/log.cfg file in the following line:

```
log4j.appender.basic.File=/OrderManagement/log/OrderManagement.log
```

At the end of the log.cfg file, the following command defines the logging level:

```
log4j.rootCategory=ERROR, basic
```

As mentioned in the previous section, you may change ERROR to WARN or DEBUG to increase the level of information output.

Loadlog.bat

If the logging level on the server is changed, these changes do not normally take effect until you stop and restart WebSphere. However, you can avoid this if you manually installed the CCClient folder. The loadlog.bat file in the /OrderManagement/CCClient folder may be run. You can change the logging level on the server to whatever is currently specified in the /OrderManagement/cfg/log.cfg file. You must edit this file (right-click the file and select **Edit**) as follows before you run it:

1. Set the path to the Callconnect folder:

   ```
   set ROSEAO_PATH=C:\CallConnect
   ```

2. Set the server:

   ```
   set SERVER=<iSeries>
   ```

3. Set the port number to the port required:

   ```
   start javaw -Djava.naming.provider.url=iiop://%SERVER%:900…
   ```

PreGenerate Buying Lists

The PreGenerateBuyingLists.log file is out put when you run System21 option 11/OMP. It is in the /OrderManagement/log folder by default. In test systems, you may modify the OSLOMD3/OM311CLP program to direct the output somewhere else.

3.5 Manual configuration

The standard installation instructions assume that the user wants to install call.connect in the easiest way. Since many variations are possible, this section includes extra information on alternative setups.

3.5.1 Non-standard Order Management and call.connect installation

You may want to install to libraries and folders other than those used in the standard installation. This section explains how to achieve this.

System21 and Order Management

All System21 applications may be installed in non-standard libraries. For more information, see the *Geac System21 Setup and Installation Guide.*

If call.connect is installed in non-standard files on the server, the OSLOMD3/OM311CLP CL program may be amended if required. It runs Buying List Generate 11/OMP in System21.

Java components and configuration files

You may install these components and configuration files anywhere on your iSeries server. However, it is generally simpler to keep all the subfolders in one folder.

If you want to install them somewhere other than in /OrderManagement, then you must amend the configuration files listed in the following sections. For each file, we assume that call.connect was installed to /OrderManagement/test.

/OrderManagement/test/cfg/ejb_default

Open this file in a notepad, and change the following lines:

```
environment.dir=/OrderManagement/test/cfg
log.config=/OrderManagement/test/cfg/log.cfg
```

/OrderManagement/test/cfg/log.cfg

Open this file in a notepad, and change the following line:

```
log4j.appender.basic.File=/OrderManagement/log/OrderManagement.log
```

/OrderManagement/test/cfg/standard.properties

Open this file in a notepad, and ensure that the following parameters are set as shown here:

```
Datasource.stateless=as400Source
system21.currentcompany=<The company you want to use>
system21.user=OMUSER
system21.password=OMUSER
system21.glcompany=<The GL company you want to use>
```

/OrderManagement/test/client/PreGenerateBuyingLists.sh

Open this file in a notepad, and change the following line:

```
export ROSEAOHOME=/OrderManagement/test
```

You must also amend the following lines if a test instance of WebSphere is running (see 3.6, "Alternative configurations" on page 62). In this example, *port 902* is used to run an instance of WebSphere called *test*:

```
echo java -Djava.naming.provider.url=iiop://localhost:902
java -Djava.version=1.2 -Djava.naming.provider.url=iiop://localhost:900
```

3.5.2 WebSphere manual configuration

If you are not using the XML import method to configure WebSphere, then proceed as explained in this section. The instructions refer to the standard instance of WebSphere and standard file structure. If you are using a test instance, vary the file paths accordingly.

Starting QEJBSBS on the iSeries

Follow these steps:

1. Sign on to the iSeries as `QSECOFR`.

2. Start the QEJBSBS subsystem as for the standard configuration.

3. Verify that QEJBADMIN and QEJBMNTR are active.

4. Check the job log of QEJBADMIN for the message "`WebSphere administration servers QEJBADMIN ready`". This may take a while, but do not proceed until you see this message in the job log.

Starting the WebSphere console on a PC

Follow these steps:

1. In a command prompt, change the directory to:

   ```
   ..\WebSphere\appserver\bin
   ```

   This is the standard directory for WebSphere installations. If it is somewhere else, amend the path accordingly.

2. To start the console for the specified machine in the default environment, enter the following command:

   ```
   adminclient <iSeries>
   ```

   This may take some time.

3. Wait for the message "`Console Ready`". The WebSphere topology view appears by default as shown in Figure 3-23.
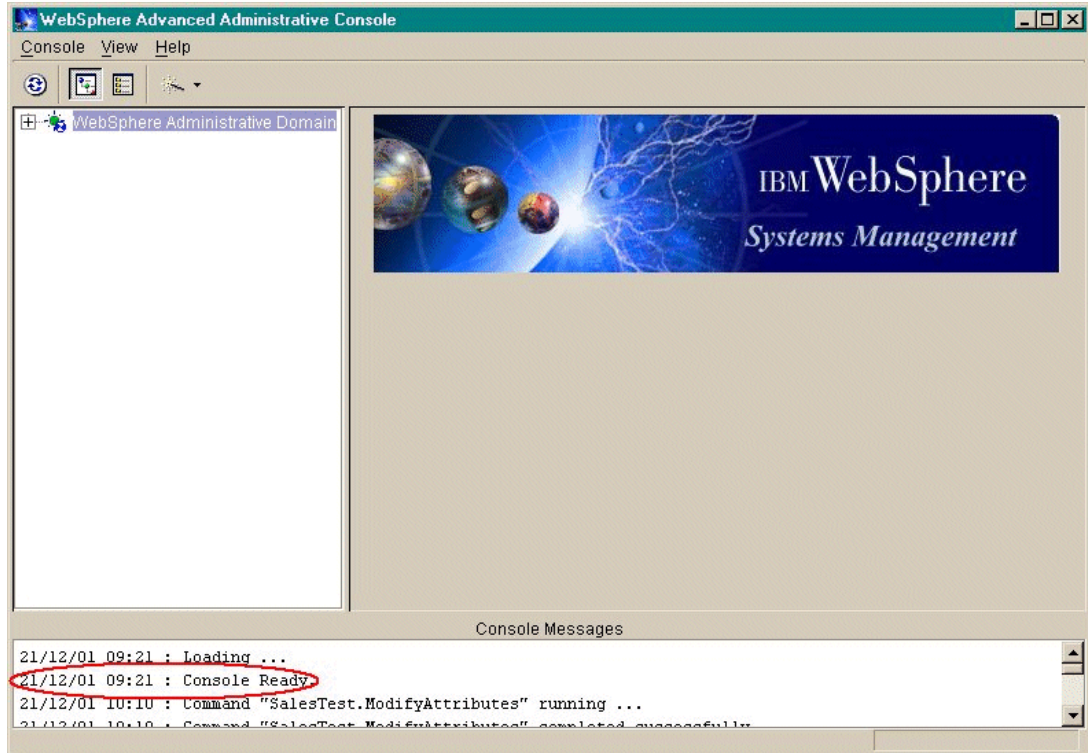
Figure 3-23   WebSphere console

Creating JDBC drivers

In the WebSphere console, create a Native JDBC driver as explained here:

1. Select the **Types** tab. On the Types page, select **JDBC Drivers**, right-click, and select **Create**.

   Alternatively, you can select the **Topology** tab. Select **WebSphere Administrative Domain**, right-click, and select **Create-> JDBC driver**.

2. In the JDBC Drivers Create window, you can enter anything for Name, but "Native" is conventional on an iSeries server. For Implementation class, select the one that ends DB2Driver.

3. Wait for the message, "Create completed successfully".

4. In the Topology page, right-click the **driver** and select **Install**. This is not necessary for a native driver.

5. Select **iSeries-> Browse for JAR file-> QIBM-> ProdData-> Java400-> ext-> db2_classes.jar**.

6. Increase the number of connections. See Chapter 5, "Performance tuning" on page 107.

Creating DataSources

The call.connect software is delivered with the ejb_default file on both the client and server set to use a data source called jt400. If you want to use another data source, then you need to modify these files to reflect the new name.

On the iSeries, in the /OrderManagement/cfg folder, find **ejb_default.cfg**. Check the standard.properties to ensure that the datasource.stateless name is the same as the jndi.datasource parameter in ejb_default.cfg.

After you decide a name, create the data source as explained here:

1. Click the **Types** tab. On the Types page, select **DataSource**, right-click, and select **Create**.

   Alternatively, you can click the **Topology** tab. Right-click **WebSphere Administrative Domain** and select **Create-> DataSource**.

2. Enter the following values in the Create a DataSource panel:

   ```
   Data Source Name = jt400 (or as specified in ejb_default.cfg)
   Database name = <iSeries>;naming=system;translate binary = true
   ```

   If additional options are required, for example, jdbc trace, then add `;trace = true` and so on here.

3. Create and wait for the completion message.

Setting the node parameters

The Deployed JAR directory should be set to the folder that will contain the JAR files for deployment. In a standard deployment, this is /OrderManagement/Deployed.

1. In WebSphere, select **Topology**. Click the iSeries node and in the Dependent Classpath.

2. Manually enter the names of all the "_logic" JAR files + jevServerSupport.jar + log4j.jar, fully qualified using the naming system of the server machine, for example:

   ```
   /OrderManagement/Deployed/application_logic.jar:/OrderManagement/Deployed/salesorder_log
   ic.jar:…/OrderManagement/Deployed/jevServerSupport.jar:/OrderManagement/Deployed/
   log4j.jar
   ```

   > **Note:** ONLY LOGIC JARS HERE + jevServerSupport.jar + log4j.jar

3. Apply and wait for the completion message.

Creating the application server

To create the application server, follow these steps:

1. Click the **Topology** tab and the **iSeries node**. Right-click the node and select **Create-> Application Server**.

2. On the window that appears, for Name, type `SalesOrder` or anything that makes sense. The name must be less than or equal to 10 characters in length because this name will appear in the WRKACTJOB SBS(QEJBSBS) command.

3. For command line arguments, manually key the entry. The easiest way to do this is to open a notepad session and create the following string. Then cut and paste it into WebSphere:

   ```
   -Xms512m -Djev.config=/OrderManagement/cfg/ejb_default.cfg -classpath
   /OrderManagement/deployed/fixes.jar:/OrderManagement/deployed/log4j.jar:/OrderManagement
   /deployed/xerces.jar:/OrderManagement/deployed/util.jar:/OrderManagement/deployed/
   secman.jar:/OrderManagement/deployed/jevServerSupport.jar:/OrderManagement/deployed/
   log4j-full.jar:/OrderManagement/deployed/enquiries_logic.jar:/OrderManagement/deployed/
   security_logic.jar:/OrderManagement/deployed/core_application.jar:/OrderManagement/
   deployed/core_technical.jar:/OrderManagement/deployed/roseao_logic.jar:/OrderManagement/
   jms/openjms-0.5.jar:/OrderManagement/jms/openjms-rmi-0.5.jar:/OrderManagement/jms/
   jms.jar:/OrderManagement/jms/exolabcore-0.1.jar:/OrderManagement/deployed/
   salesorder_msg.jar:/OrderManagement/deployed/messaging_logic.jar:/OrderManagement/client
   /roseao_client.jar
   ```

   There are no spaces or line feeds after -classpath in this statement.

4. Set the working directory as /OrderManagement/log.

5. Use the WRKLNK / command to ensure that *PUBLIC has *RWX authority to this folder.

   An exclamation mark (!) in front of the stdout and stderr files means that they will be cleared each time WebSphere is restarted, which is recommended.

6. Create the application server and wait for a message.

7. Start the application server to ensure that it is configured correctly. If it turns blue, click **OK** and then stop it. Otherwise investigate the problem and correct it before you proceed.

Creating an EJB container

Follow these steps:

1. On the application server that you just created, right-click and select **Create EJB Container**.

2. For Name, enter anything sensible, such as `SalesOrderContainer` or the same name as the application server.

3. Create the EJB container and wait.

4. Start the application server and EJB container to ensure that they are configured correctly. If they turn blue, click **OK** and then stop them. Otherwise investigate the problem and correct it before you proceed.

Creating enterprise beans

In the container that you just created, create the enterprise beans:

1. Right-click the **container** and select **Create-> Enterprise Bean**.

2. Browse for the JAR file. Be patient because this takes time. Do not double-click the file.

3. Drag the window with another IBM window behind to verify that it is working. If it is processing, multiple images of the front window appear.

4. If the Deployed JAR directory is set correctly on the node, then the contents of this directory appear by default, for example /OrderManagement/Deployed for a standard configuration. If you did not set this correctly, change the "Look in" directory at the top of the browse window.

5. Click the **_deployed** JAR file, for example **RoseAO_deployed.jar**, to select it.

6. A message appears about the number of beans being deployed. You may see a "`Not yet deployed`" message. Click **OK**.

7. Do not enable workload management. Select **No** when a message about work management appears.

8. Then one message appears for each bean. Click **OK** for each message them even though it may seem like nothing has happened.

9. Start the application server, container, and beans to ensure that they are configured correctly. If they turn blue, click **OK**, close the console, and leave them running. Otherwise investigate any problems and correct them.

The console should look like the example in Figure 3-24.
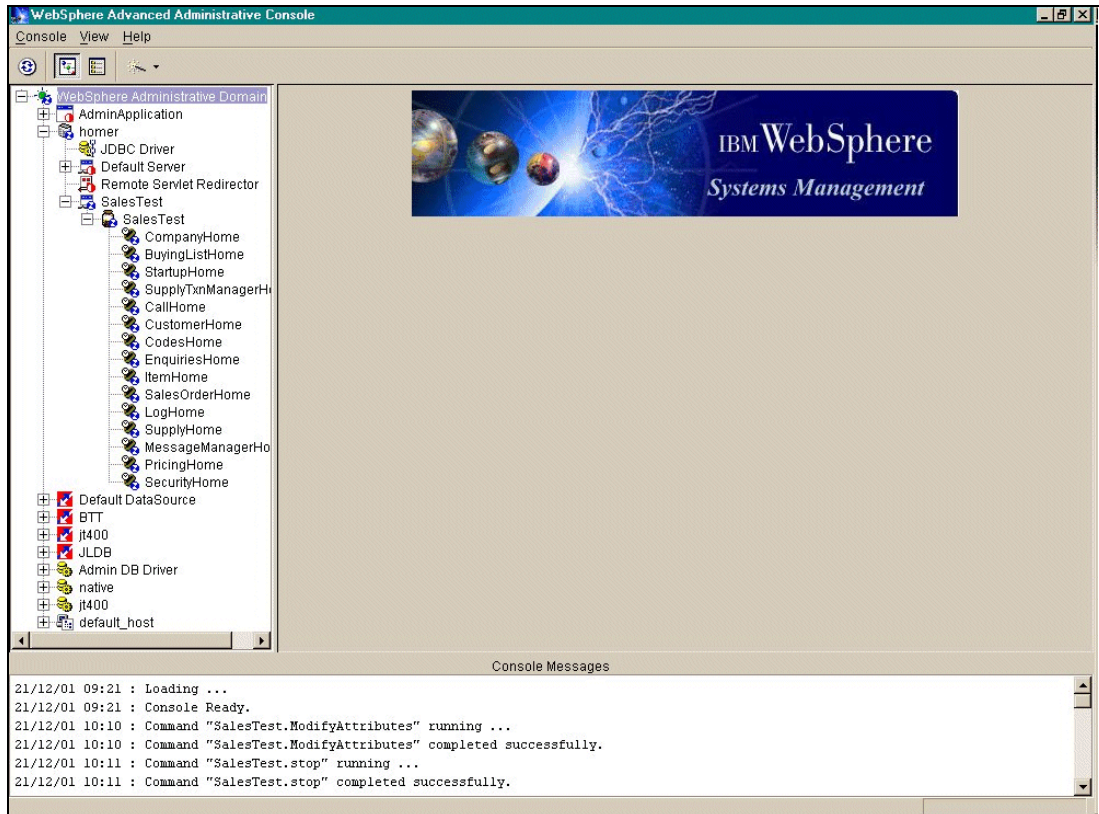
Figure 3-24   WebSphere console after deployment

3.5.3 Manual client installation

For a simple setup for one or two users, it is possible and easier to configure each client individually rather than using Java Web Start. However, configuring each client individually is not as well controlled as Java Web Start. Each user has their own set of client software, where with Java Web Start, only one set of software is installed on the server.

To install the client software manually, follow the instructions provided here. They assume that you want to install the software to a folder, called callConnect, on your C drive. If you want to install the software somewhere else, amend the path accordingly:

1. Copy the jre folder from /OrderManagement to the c:\callConnect folder on your client.

2. Copy the CCClient folder from CD to c:\callConnect\CCClient.

> **Attention:** You must maintain this structure. That is, the jre must be at the same level in your file structure as in the client folder.

3. Verify that the files within c:\callConnect\CCClient are correctly configured as listed in the following sections.

Ejb_default.cfg

The contents of this file are:

```
environment.dir=C:\\callConnect\\CCClient\\
jndi.as400Source=jdbc/jt400
log.config=C:\\callConnect\\CCClient\\log.cfg
```

Log.cfg

Check the following line:

```
log4j.appender.basic.File=C:\\callConnect\\CCClient\\callConnect.log
```

callConnect.bat

Verify that the ROSEAO_PATH is set as follows:

```
set ROSEAO_PATH=C:\callConnect
set SERVER=<iSeries>
```

Verify that the CLIENT_PATH is set as follows:

```
set CLIENT_PATH=%ROSEAO_PATH%\CCClient
```

If you are using a test instance, change the port number in the following line:

```
start javaw -Djava.naming.provider.url=iiop://%SERVER%:900 etc
```

Running the manually configured client

Create a desktop icon pointing to callConnect.bat. The first time you run this, it displays:

- ► Default User: User for call.connect
- ► Default Password: Password for user
- ► S21 Company: XX (the AR company to be used)
- ► S21 GL Company: XX (the GL company to be used)
- ► S21 User Name: OMUSER or user set up in 3.2.6, "User profiles" on page 36
- ► S21 Password: OMUSER
- ► Datasource: as400Source. This must be the JNDI name used in ejb_default. It is not the name of the datasource set up in WebSphere.

After the first time, the system requests only the user and password to sign on. Remember, users and passwords are case sensitive.

3.6 Alternative configurations

This section provides information about alternative configurations of WebSphere and call.connect. It is useful to anyone who wants to set up testing facilities for example.

3.6.1 Setting up a test instance of WebSphere

The instructions in 3.2, "Standard installation procedures" on page 22, explain how to install and configure call.connect using the default instance of WebSphere, which runs on port 900. The call.connect software on the CD comes pre-configured to run this instance in a folder named OrderManagement. However, it is often useful or necessary to set up other instances of WebSphere. For example, this would allow you to test fixes in the form of new EJBs before you move them to the live system. Users may want to use different System21 data for testing. However, this alone does not require a different configuration of WebSphere. It merely requires a different job description and a user profile to use that job description.

The steps outlined in Table 3-4 are required to configure a different instance of WebSphere.

Table 3-4 Steps required to configure a WebSphere instance

Step	Action	Completed
1	Set up test data in System21, including OM (optional)	
2	Set up journaling	
3	Set up a test job description	
4	Set up a test user specifically for JDBC connections	
5	Configure server	
6	Run stored procedures and SQL scripts (optional)	
7	Configure WebSphere	
8	Configure JMS	
9	Configure Java Web Start or install client software	

3.6.2 Setting up an iSeries server for a test system

This section explains how to set up a test system for the various parts of call.connect installed on the iSeries server.

System21 data

To use a set of test data, you may set up a separate System21 environment. Normally, this involves setting up alternative libraries with different data or a copy of live data. A frequently used method is to change the name of the standard library OSLD1F3, OSLSLF3, etc. to TSTD1F3, TSTSLF3, etc. Within System21, these test libraries are used in place of the live libraries.

In the rest of this section, TST… is used to refer to test libraries and other variables, but you can use other names. It is also possible to use different libraries for other object so that you may also create such libraries as TSTOMD3, etc.

Journaling

Journal the test files. The minimum set of files is the same as for the standard configuration, but in the TST libraries. You must set up new libraries TSTF3 and optionally TSTF3R for the journal. You may optionally set up the receiver, if an ASP is used.

Job description

The default configuration comes with a preconfigured job description OMJOBD, with its library list already set. This job description is used by a special user OMUSER to set the library lists for JDBC connections.

To use the test libraries, you must create a test job description, such as TSTJOBD. This can be held in the TSTF3 library. The job description library list should include TSTOMF3, TSTD1F3, TSTD2F3, TSTSLF3, TSTGLF3, TSTCSF3, TSTOMD3, TSTOEP3, TSTINP3, TSTGLD3, TSTSLD3, IPGAMF4, IPGCFF4, IPGAMP4, IPGCFP4. TSTOIF4, TSTOID4, TSTWTP3, and TSTCSD3.

If any of these libraries are not created, then the previous list should show the corresponding OSL version.

If other .connect products or other products that use a workflow are running, you must add TSTWFF3 to the job description library list.

Test user

You must set up a test user profile, such as TSTUSR, for the JDBC connections for the test instance. This user must use the job description created previously.

You do not need extra profiles for the test system in UserDirectory.XML or System21.

3.6.3 Server configuration

Copy the server objects into a new folder, such as /OrderManagement/test. This folder is used in the examples throughout the following sections. Next, change the configuration files listed in the following sections.

Server ejb_default.cfg

Change ejb_default.cfg to point to the correct place for environment.dir and log.config, for example:

```
environment.dir=/OrderManagement/test/cfg
log.config=/OrderManagement/test/cfg/log.cfg
```

Server log.cfg

Edit log.cfg so that log4j.appender.basic.File points to the log folder, for example:

```
log4j.appender.basic.File=/OrderManagement/test/log/OrderManagement.log
```

Server standard.properties

Edit standard.properties as shown here:

- ► Datasource.stateless: <*The datasource you want to use*>
- ► system21.currentcompany: <*The company you want to use*>
- ► system21.user: <*The test user defined previously*>
- ► system21.password: <*Password for this user*>
- ► system21.glcompany: <*The GL company you want to use*>

See 3.6.4, "WebSphere administration" on page 65, for more information about the datasource.

Client folder

Edit or view the file PreGenerateBuyingLists.sh in the client folder to ensure it points to the correct files as follows:

```
export ROSEAOHOME=/OrderManagement/test
```

Amend the port number on the `echo java` and `java` commands at the end of this script to point to the appropriate port depending on the instance of WebSphere used. The default is 900, so to use port 902 change:

```
localhost:902
```

Stored procedures

Create the stored procedures in the same way as for the default instance. However, be sure that the library list is set as tstomf3, tstomd3, tstd1f3, tstd2f3, tstslf3, and tstglf3.

In addition, it is possible that the actual RPG programs (not just the data files) may be in test libraries. In this case, create new copies of the scripts and amend them to point at the new libraries (for example, TSTOMD3).

3.6.4 WebSphere administration

This section covers the topics related to WebSphere.

Starting the administration server

QEJBSBS must be running. If the default instance is not required, QEJBADMIN and QEJBMNTR may be ended. Follow these steps:

1. Sign on as QSECOFR.

2. To start a WebSphere instance called *test*, call program STRWSSVR:

   ```
   CALL PGM(OSLOMD3/STRWSSVR) PARM(TEST)
   ```

3. Jobs TESTADMIN and TESTMNTR should be displayed in the QEJBSBS subsystem. Check the job log of TESTADMIN and wait for the message "WebSphere administration server TESTADMIN ready", similar to the default system.

   WebSphere cannot start until the administration server is ready.

Importing the configuration file

> **Attention:** To import the configuration file, the WebSphere instance must be running. However, the console must *not* be running.

Follow these steps:

1. The import tool needs to run within Qshell. Run the start Qshell command, and after each command, wait for $ signs to appear.

   ```
   STRQSH
   ```

2. Switch the current directory within Qshell:

   ```
   cd /QIBM/ProdData/WebASAdv/bin
   ```

3. Import the configuration file Config.xml from OrderManagement/Config.

4. To use the delivered file to configure the test instance running on port 902 on an iSeries, use the following command, substituting your server name for *iSeries*:

   ```
   XMLConfig -adminNodeName <iSeries> -import /OrderManagement/test/Config/Config.xml
   -instance test -nameServiceHost <iSeries> -nameServicePort 902
   ```

   > **Notes:**
   > ► Leave a space before each - sign. Leave a space after each keyword, for example `-adminNodeName`, `-import`, and `-substitute`.
   > ► There is only one pair of quotes around the entire substitution string.

The config.xml file contains the following variables, which are substituted on the import command and replaced with real names:

► **$NodeName$**: This is the WebSphere node name. It is usually the system name, but it is case sensitive.

► **dir**: This is the root directory for the installation. This is the directory that contains the subdirectories: cfg, deployed, and log. For an instance called *test*, this could be /OrderManagement/test. The EJBs to be deployed are in /OrderManagement/Test/Deployed.

▶ **$appname$**: This is the application name, such as SalesTest. It should be different from the default application name so that they can be distinguished if they are both running together.

> **Note:** The application name should be less than 11 characters in length. This way you can see it clearly on the iSeries using the WRKACTJOB SBS(QEJBSBS) command. Longer names are allowed, but they are truncated when viewed this way.

▶ **$container$**: This is the container name, which is typically the same as the application name, but it can be different.

Starting application server

Follow the steps to start the application server and its beans:

1. Start the application.

2. Start a WebSphere console.

3. In a command prompt, change the directory to:

   `..\WebSphere\appserver\bin`

   > **Note:** This is a standard directory after a default installation of the console, but you may install it someplace else.

4. To start the console, enter the command:

   `adminclient <iSeries> <Port Number>`

   For this example, to start a console for the instance running on port 902, you may enter:

   `adminclient <iSeries 902>`

5. Wait for message "`Console Ready`" to appear in the console. The topology view is displayed by default.

6. Open the node, application server, and container to display the beans.

7. Click the **application server**. Click **Start** and then wait. The beans turn blue as they start. This may take several minutes.

3.6.5 Manual client installation

For one or two test users, it may be easier to use a manually installed client rather than Java Web Start. To change from live to test versions of Java Web Start, you must change the configuration each time, although they are fairly simple. Details on both are included here.

For a simple setup for one or two users, it is possible to configure a client individually rather than using Java Web Start. This is simpler than using JWS, but is not as well controlled. Every user will have their own set of client software, whereas with JWS, only one set is installed on the server.

To install the client software manually, follow the instructions provided here. They assume that you want to install the software to a folder, called callConnect, on your C drive. If you want to install the software somewhere else, amend the path accordingly:

1. Copy the jre folder from /OrderManagement/test to a folder c:\callConnect\test on your client.

2. Copy the CCClient folder from /OrderManagement/test to c:\callConnect\test\CCClient.

> **Important:** You must maintain this structure. For example, the jre folder must be at the same level in your file structure as the client folder.

3. Verify that the files within c:\callConnect\CCClient are correctly configured as listed in the following sections.

Ejb_default.cfg
Verify the contents of this file as they are shown here:

```
environment.dir=C:\\callConnect\\test\\CCClient\\
jndi.as400Source=jdbc/jt400
log.config=C:\\callConnect\\test\\CCClient\\log.cfg
```

Log.cfg
Check the following line:

```
log4j.appender.basic.File=C:\\callConnect\\test\\CCClient\\callConnect.log
```

callConnect.bat
Verify that the ROSEAO_PATH is set as follows:

```
set ROSEAO_PATH=C:\callConnect\test
set SERVER=<iSeries>
```

Verify that the CLIENT_PATH is set as follows:

```
set CLIENT_PATH=%ROSEAO_PATH%\CCClient
```

If using a test instance, change the port number in the following line to 902, for example:

```
start javaw -Djava.naming.provider.url=iiop://%SERVER%:900 etc
```

Running the manually configured client
Create a desktop icon pointing to callConnect.bat. The first time you run this, it displays:

► Default User: User for call.connect
► Default Password: Password for user
► S21 Company: XX (the AR company to be used)
► S21 GL Company: XX (the GL company to be used)
► S21 User Name: OMUSER or user set up in 3.2.6, "User profiles" on page 36
► S21 Password: OMUSER
► Datasource: as400Source. This must be the JNDI name used in ejb_default. It is not the name of the datasource set up in WebSphere.

After the first time, the system requests only the user and password to sign on. Remember, users and passwords are case sensitive.

Testing Java Web Start
Follow the steps for a standard configuration until you reach the create configuration step. Then complete the steps in the following sections for your HTTP Server.

HTTP Server for iSeries (original)
Configure your IBM HTTP Server for iSeries (original) as explained in the following steps:

1. Under the Configurations option, select the **Create Configuration** link.

2. The Create Configuration page appears (see Figure 3-14 on page 43 for an example). Follow these steps:

a. Insert the name of the configuration, such as the iSeries name.

b. Select the **Create based on existing configuration** option and enter `CONFIG`.

c. Click **Apply**.

3. Select the **Request Processing** option on the left and select the **Request routing** link.

4. The Request routing page (see Figure 3-15 on page 44 for an example) appears. Insert the following actions, templates, and replacement file paths. Be sure to click the **Insert after** radio button:

Index	Action	URL Template Replacement File Path
Pass	/callconnect/test/	/ordermanagement/test/client/index.html
Pass	/callconnect/test/*.gif	/ordermanagement/test/client/*.gif
Pass	/callconnect/test/callconnect.jnlp	/ordermanagement/test/client/callconnect.jnlp
Pass	/callconnect/test/*.jar	/ordermanagement/test/client/*.jar
Pass	/callconnect/test/setup.exe	/ordermanagement/test/client/setup.exe
Pass	/callconnect/test/help/	/ordermanagement/test/client/help/introduction.htm
Pass	/callconnect/test/help/*.gif	/ordermanagement/test/client/help/*.gif

5. Select the **Languages and Encoding** link on the left. Then select **MIME types**.

6. The MIME types page (see Figure 3-16 on page 45 for an example) appears. Enter the following information for the system to recognize the .jnlp and .htm extensions. Be sure to click the **Insert after** radio button:

   – **.jnlp extension**

     • File extension: `.jnlp`
     • Content type/subtype: `application/x-java-jnlp-file`
     • Encoding: `8 bit`
     • Quality: `1.0`

   – **.htm extension**

     • File extension: `.htm`
     • Content type/subtype: `text/HTML`
     • Encoding: `8 bit`
     • Quality: `1.0`
     • Character Set: `ISO-8859-1`

   Click **Apply**.

Now you create the instance:

1. On the left-hand panel, select **Server Instances** and then the **Create server instance** link.

2. The Create server instance page (Figure 3-17) appears. Follow these steps:

   a. Enter the server instance name, such as the iSeries name.
   b. Enter the configuration that was just created.
   c. Click **Create**.

You see the message "`Message The server instance was successfully created`".

In WRKACTJOB SBS(QHTTPSVR), you now see ADMIN jobs and jobs with the name of the instance that was just created. All should be in *wait* states.

HTTP Server for iSeries (powered by Apache)

Configure your IBM HTTP Server for iSeries (powered by Apache) as explained in the following steps:

1. At the top of the page, select **Configuration**.

2. Under Configuration structure, verify that *HTTP Server Name* **global settings** appears in bold.

3. You should see the *HTTP Server Name* (HOMERA) global settings page (see Figure 3-20 on page 49 for an example). Under Web Site Definition, select **Serve New Directory wizard**. Click **Next**.

4. On the Serve New Directory wizard, complete the following tasks:

   a. Select **Static Web Pages and Files**.

   b. For Directory to Serve from, enter the client directory, such as
      `iSeries/OrderManagement/test/client`.

   c. For Alias, enter the name required on the browser, such as `/callconnect/test/`.

   d. Click **Finish**.

5. Click the **OrderManagement/client** directory under Configuration structure to select it.

6. On the Director /OrderManagement/client page (Figure 3-21), under Processing Requests, click **Content Negotiation**.

7. On the Content Negotiation page, scroll about half way down to **Additional Meta (MIME) information** (Figure 3-22). Then follow these steps:

   a. Click **Add**.

   b. Enter the following information:
      - File Extension: `.jnlp`
      - Type: `content-type`
      - Value: `application/x-java-jnlp-file`

   c. Click **Add**.

   d. Enter the following information:
      - File Extension: `.jar`
      - Type: `content-type`
      - Value: `application/x-zip-compressed`

   e. Click **Add**.

8. Click **Apply**.

9. At the top of the page, select **Administration**.

10. Click the **Manage HTTP Servers** link in the left-hand panel.

11. On the Manage HTTP Servers page (Figure 3-25), stop the HTTP Server and then start it.
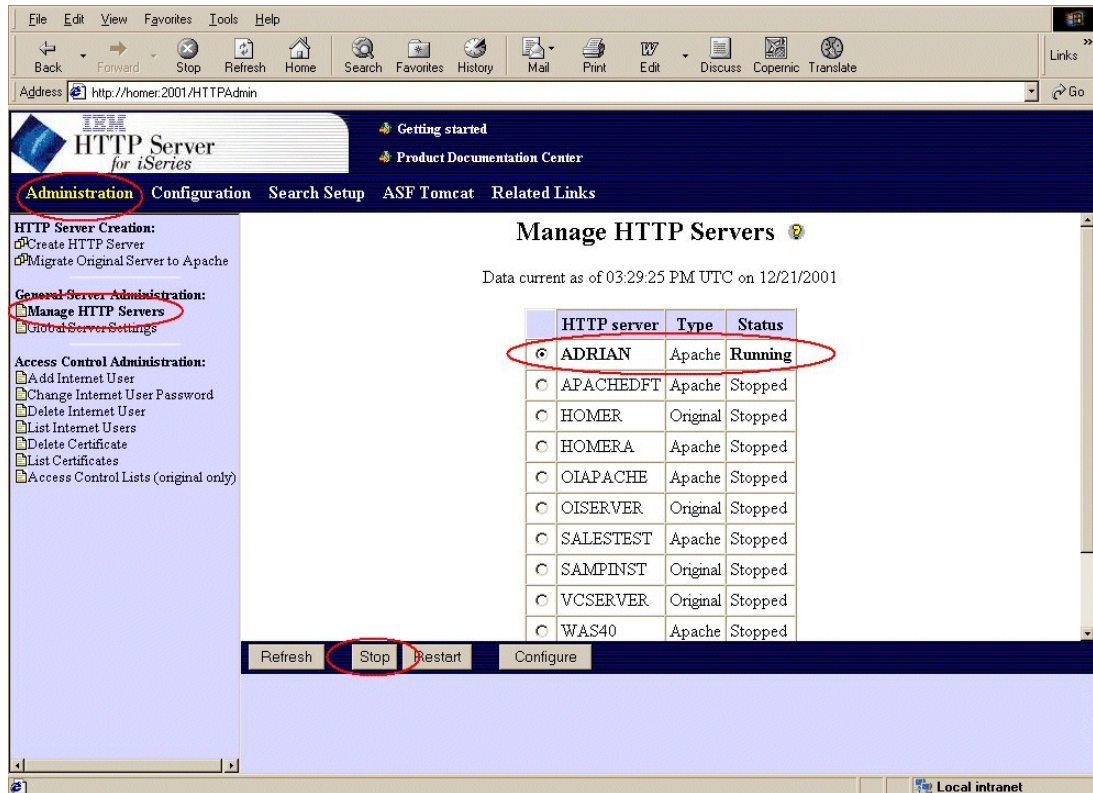
Figure 3-25   Manage HTTP Servers page: Stopping the server

Configuring CallConnect.jnlp

In Windows Explorer, modify the file /ordermanagement/test/client/callconnect.jnlp as follows to reflect the alias that you previously set up:

```
codebase="http://<iSeries>/callconnect/test/"
```

Change the server name. Also change the following line to point to the correct iSeries and port number:

```
<property name="java.naming.provider.url" value="iiop://<iSeries>:900"/>
```

The default port number for the default instance of WebSphere is 900. If you use the test instance, as recommended in this chapter, you should change the port number to 902.

Configuring JWS.bat

In Windows Explorer, enter your iSeries name in the following line in the /ordermanagement/test/client/JWS.bat file:

```
"C:\Program Files\Java Web Start\javaws" http://<iSeries>/callconnect/test/CallConnect.jnlp
```

Changing to another instance

If you want to change to another instance of WebSphere, such as test to default, then you have to change the client configuration files as created by the Java Web Start installation. Users may want to create two versions of these files and swap them as required.

These files are installed to C:\Winnt\Profiles*<signed on user>*.

The files to modify are:

► ejb_default.cfg
► log.cfg
► standard.properties

See 3.6.5, "Manual client installation" on page 66.

4

Installing and setting up vendor.connect

This chapter explains how to install and set up Geac's vendor.connect product.

Throughout the installation instructions, you will see variables in *italics*. For each of these variables, you need to provide a real name. For example, for the variable *<iSeries>*, you must change it to the name of the machine you are using.

Keep in mind that all the required components are on the vendor.connect CD. However, throughout this chapter, references are made to the Geac applications *commerce.platform*, *@ctive Modeler*, and *secure.connect*. vendor.connect users may install these applications, although they are not essential.

This chapter also refers to a folder named *Jacada*, which contains all the components of Jacada that required for vendor.connect. You may obtain a Jacada server from your Geac distributor, although one is not required for vendor.connect.

4.1 Preparing for the installation

This section discusses the activities and checks that are required before you begin installing vendor.connect.

4.1.1 Skills required

You must be familiar with the workstation, particularly with the operation of the keyboard and mouse. You should also be familiar with the following applications, which are used in the vendor.connect configuration. Although instructions are provided at each stage, a complete beginner may find the installation to be difficult and may need help from someone with more experience.

- ► Windows NT Explorer
- ► Notepad and Wordpad or other tool to edit files for example, .bat and .xml files
- ► FTP
- ► Client Access
- ► MS-DOS
- ► OS/400 commands
- ► A Web Browser for example, Internet Explorer
- ► System21
- ► Qshell

In addition, you should also be familiar with the following products or tools:

- ► WebSphere Application Server.
- ► CL programming on the iSeries
- ► Shell scripts
- ► SQL
- ► @ctive modeler

For straightforward configurations, you may be able to proceed regardless of your familiarity with these items.

4.2 Installing vendor.connect

This section explains how to install vendor.connect. It even explains a basic setup using the configuration that is supplied on the CD.

Basic steps checklist

Table 4-1 summarizes the steps that are required to install and configure vendor.connect. You can find additional information about each step in the sections that follow.

Table 4-1   vendor.connect installation checklist

Step	Action	Completed
1	Load System21, including distribution and finance applications, Advanced Receiving (AG), Vendor Scheduling (VS), and Work Management	
2	Load Java components and configuration files for vendor.connect	
3	Restore extra libraries for vendor.connect	
4	Install and configure WebSphere	
5	Install and configure an HTTP Server	

Step	Action	Completed
6	Set up on the iSeries	
7	Configure MQSeries	
8	Install and configure the Work Management trigger handler	
9	Active Architecture framework	
10	Set up the Jconnects server	
11	User IDs and security (LDAP or XML)	
12	Set up database synchronization	
13	Test the configuration	
14	Back up the configuration	

vendor.connect components

Several different types of objects are required for this installation such as iSeries library, JAR, and configuration files. You can learn more about these in the following sections.

4.2.1 System21 base

> **Note:** This redbook does not provide instructions for installing System21. You can find information on installing System 21 and current details regarding your installation scenario in the *Geac System21 Installation and Setup Guide*.

To run vendor.connect, you need to load at least the following System21 applications at V3.5.2b Service Pack 5:

- ► Application Manager
- ► Geac System21
- ► Work Management
- ► Inventory (IN)
- ► Order Entry (OE)
- ► Purchase Management (PM)
- ► Vendor Scheduling (VS)
- ► Advanced Receiving (AG)
- ► Cash Management (CS)
- ► General Ledger (GL)
- ► Sales Ledger (SL)
- ► Purchase Ledger (PL)

You can obtain these applications from your Geac distributor.

4.2.2 Java components and configuration files

All other components of vendor.connect are on the vendor.connect CD. To load them, follow the instructions in the readme file.

4.2.3 Restoring libraries

Three additional libraries are delivered in the AS400 folder. These are held as save files. To restore them, use File Transfer Protocol (FTP) to copy them from your workstation to the iSeries server. See the following steps for an example:
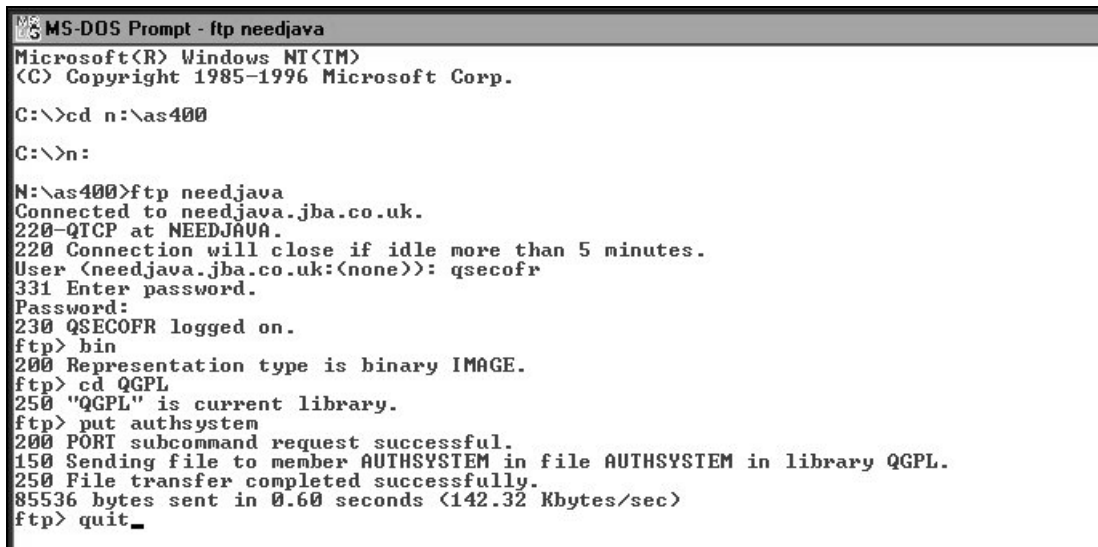
1. Open a DOS prompt and enter the following commands:

```
CD C:\AS400
FTP <iSeries>
```

2. Enter a user name and password for the iSeries.

3. Enter the following commands in the order shown:

```
Bin
Cd <the library where you want to put the Save File>
Put <Save File name>
Quit
```

Figure 4-1 shows an example of how this appears on the DOS prompt.

Figure 4-1   FTP commands to copy a Save File from N:\AS400 to QGPL on an iSeries called Needjava

4. The Save Files are called authsystem, oslvcf3, and oslvcd3. After they are on the iSeries, use the Restore Library command to restore the libraries (of the same name).

4.2.4 Installing and configuring WebSphere

The instructions in the following section are specific to the vendor.connect installation. For complete instructions on installing and configuring WebSphere Application Server, go to:

http://www-1.ibm.com/servers/eserver/iseries/software/WebSphere/wsappserver/docs/doc.htm

It is possible to use the default instance of WebSphere. However, you may want to create a new instance, which may easily have an HTTP instance attached to it. To use the default instance, start QEJBSBS as shown in the following example. Then stop the QEJBADMIN and QEJBMNTR jobs and restart them, using the `strwasinst` command with the HTTP parameter.

To create a new instance, start a Qshell session with the following command:

```
STRQSH
```

Wait for the $ sign to appear after each entry.

Creating a new instance

To create a new instance called VC on port 905, enter:

```
crtnewinst -instance VC -bootstrap 905 -lsd 9005
```

For full details, see the WebSphere Application Server documents on the Web at:

http://www-1.ibm.com/servers/eserver/iseries/software/WebSphere/wsappserver/docs/as400v35/docs/doccntr.pdf

Starting the administration server

To start the administration server, follow these steps:

1. On the iSeries, enter the command:

   ```
   STRSBS SBSD(QEJB/QEJBSBS)
   ```

   This starts the QEJB subsystem and two autostart jobs QEJBADMIN and QEJBMNTR.

2. WebSphere cannot start until the administration server is ready. To find out the status of the administration server, check the job log for the QEJBADMIN job. Enter the following command:

   ```
   WRKACTJOB SBS(QEJBSBS)
   ```

3. On the next display, type 5 next to QEJBADMIN to view the job.

4. Select option 10 to view the job log. Wait until the message "`WebSphere administration server QEJBADMIN ready.`" appears. This may take several minutes.

Starting the instance

Once QEJBSBS is running, the instance created above may start. If the default instance is not required for other purposes, QEJBADMIN and QEJBMNTR may be cancelled. To start the new instance, go to Qshell and enter the following command:

```
strwasinst -instance VC -http VC
```

This command starts the WebSphere instance VC that uses an HTTP Server instance, which is also called VC.

Importing the configuration file

Important: To import the configuration file, the WebSphere instance *must* be running, but the console *must not* be running.

The import tool needs to run within Qshell. Follow these steps:

1. Run the Start Qshell (STRQSH) command and after each command wait for $ signs to appear.

2. Change the current directory within Qshell. Enter the following command:

   ```
   cd /QIBM/ProdData/WebASAdv/bin
   ```

3. Import the config.xml configuration file from VendorConnect/Config.

4. To use the delivered file to configure the default instance on an iSeries server, use this command, substituting your server name for *iSeries*:

   ```
   XMLConfig -adminNodeName <iSeries> -import /VendorConnect/VC.xml
   -substitute "nodename=<iSeries>;host=<iSeries+domain>;port=<port number>;IP=<IP
   address>"
   ```

   It may run for some minutes.

Figure 4-2 shows an example of an imported configuration.

```
                          QSH Command Entry

   $
 > cd /QIBM/ProdData/WebASAdv/bin
   $







 ===> XMLConfig -adminNodeName homer -import /VendorConnect/vc.xml -substitute "
 nodname=homer;host=homer.jba.co.uk;port=85;IP=123.456.001.157"

 _



 F3=Exit    F6=Print F9=Retrieve F12=Disconnect
 F13=Clear F17=Top  F18=Bottom  F21=CL command entry
```

Figure 4-2   Importing the WebSphere configuration to an iSeries called homer

The import includes:

► The node
► The data source and JDBC driver
► The applications required for vendor.connect

The config.xml file contains the following variables, which are substituted on the import command and replaced with real names:

► **$NodeName$**: This is the WebSphere node name, which is case sensitive. This is relevant when you start the console with AdminClient and when you specify an Inter-ORB Protocol (IIOP) address for a Java Naming and Directory Interface (JNDI) lookup (for example, within the batch files which start client programs). It is also relevant to tools such as the XML export/import utility.

Usually the name is entirely in lowercase, However, on some systems, it is entirely in uppercase, and in theory, could be a mixture of both. If you find that the console fails to start with such messages as "`Could not get attributes`" (similar to when the service/subsystem is not started) but the service is started, then problem may be the case of the name.

You can verify the required name by using the following SQL to look at a WebSphere table:

```
select NAME
from EJSADMIN/NODE_TABLE
```

This shows the node name exactly as WebSphere wants it to appear. Do not be tempted to change the contents of this table. It is liable to make things worse rather than better.

> **Note:** You must have QSECOFR user authority to read this table.

- ▶ **$host$**: This is the iSeries name and domain, such as needjava.jab.co.uk.
- ▶ **$port$**: This is the port for the WebSphere instance running vendor.connect.
- ▶ **IP**: This is the IP address of the iSeries server.

Starting the application

To start the application, follow these steps:

1. Start a WebSphere console. In a command prompt, change the directory to:

   ```
   ..\WebSphere\appserver\bin
   ```

   > **Note:** This is a standard directory for a default installation of the console. However, you can install the console in a different directory if necessary.

2. To start the console for the specified machine in the default environment, enter the command:

   ```
   adminclient <iSeries> <Port Number>
   ```

   This may be quite slow. If an instance other than the default instance is required, enter the port number. Otherwise you may leave this blank.

3. Wait for the message "`Console Ready`" to appear in the console. The topology view appears by default.

4. Open the node, application server, and container to display the beans and servlets. You should see the application servers VennConn, VCSync, and Receiving. Click the **Application Server**. Click **Start** and wait. The beans and servlets turn blue as they start. This may take several minutes.

4.2.5 IBM HTTP Server for iSeries

vendor.connect is a Web-based application. For it to run, you must configure the HTTP Server as explained in the following sections. Two HTTP Servers are available:

- ▶ IBM HTTP Server for AS/400 (original): For V4R4 and earlier
- ▶ IBM HTTP Server for iSeries (powered by Apache): For V5R1 and later

Ensuring that the HTTP Server is ready

The following steps are the same for both servers:

1. Verify that the HTTP Server software is loaded. On the iSeries, type the following command:

   ```
   GO LICPGM
   ```

2. On the display that appears, select option 10 (Display). Check the suffixes. Look for **DG1**. If you do not find it, load it before you proceed to the next step.

3. Start the HTTP Server. On the iSeries, run the command:

   ```
   STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
   ```

4. Verify that the server is running. Enter the following command:

   ```
   WRKACTJOB
   ```

5. Verify that ADMIN jobs are running in the QHTTPSVR subsystem.

Configuring the HTTP Server on the iSeries

To configure the HTTP Server on the iSeries server, follow these steps:

1. In a Web browser, type:

   `http://<iSeries>:2001`

   Enter your iSeries server name for `iSeries`. This is the port on which the administration of HTTP is listening.

2. Sign on as `QSECOFR`.

3. The AS/400 Tasks page appears as shown in Figure 4-3. In this example, the pages are for an iSeries called *Needjava*. Select **IBM HTTP Server for AS/400**.
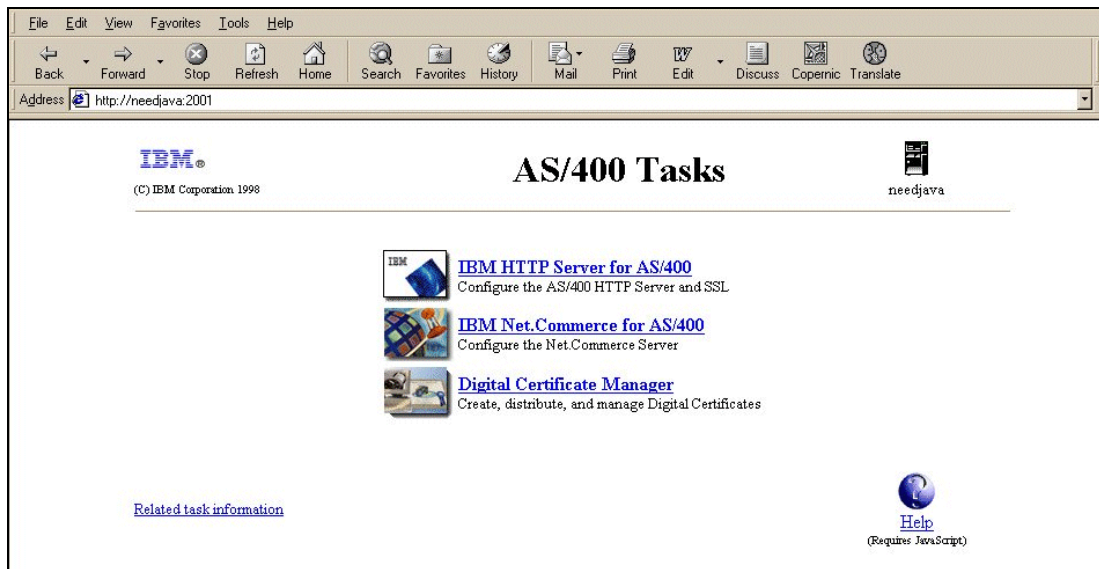
Figure 4-3   AS/400 Tasks page

4. The IBM HTTP Server for AS/400 page appears as shown in Figure 4-4. Select the **Configuration and Administration** icon on the left.
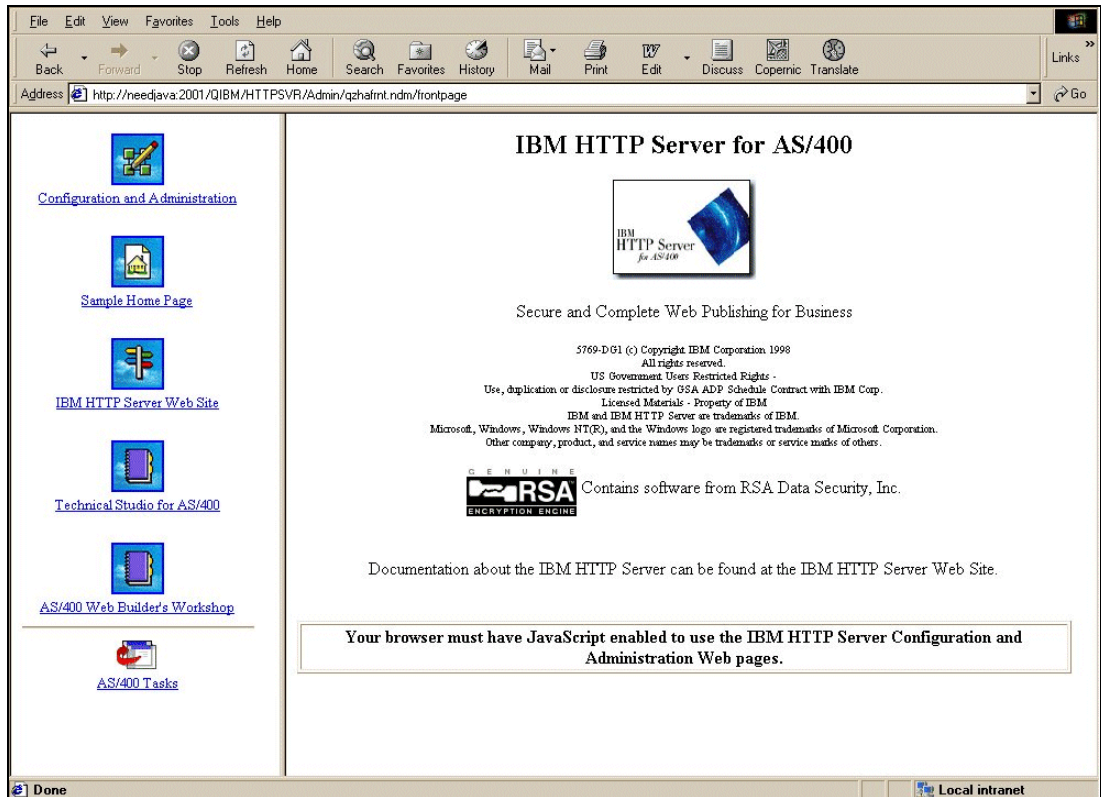
Figure 4-4   IBM HTTP Server for AS/400 page

Configuring IBM HTTP Server for iSeries (original)

This section explains how to install IBM HTTP Server for iSeries (original) on an iSeries server. The instructions to install IBM HTTP Server for iSeries (powered by Apache) are not currently available. The following steps apply to a configuration using the original server on V4R5 or earlier.

After you select the Configuration and Administration icon, as stated in the previous section, the IBM HTTP Server Configuration and Administration page appears as shown in Figure 4-5. Select the **Configurations** link from the left-hand panel on the page.
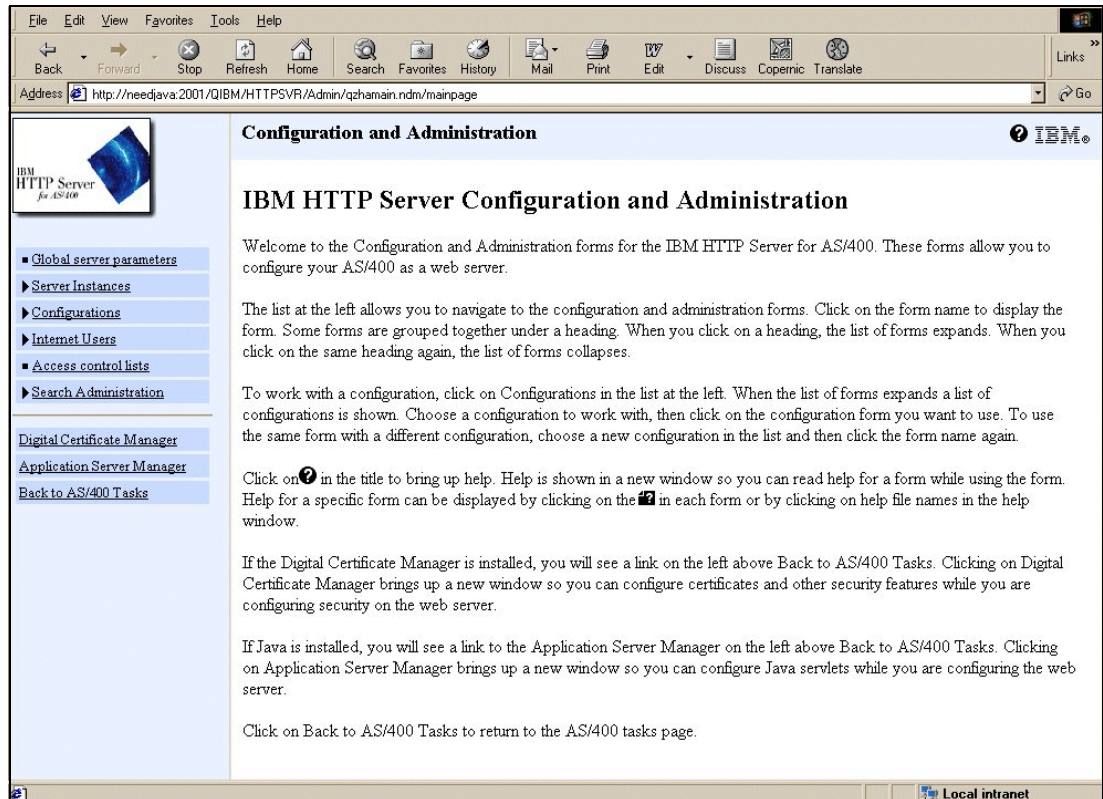
Figure 4-5   IBM HTTP Server Configuration and Administration page

Now you need to create a new configuration and a new instance, based on the IBM-supplied configurations and instances.

Creating a configuration
Follow these steps to create a configuration:

1. Under the Configurations option, select the **Create configuration** link.

2. The Create configuration page appears (Figure 4-6). Follow these steps:

   a. Insert the name of the configuration, such as the iSeries name.
   b. Select the **Create based on existing configuration** option and enter CONFIG.
   c. Click **Apply**.
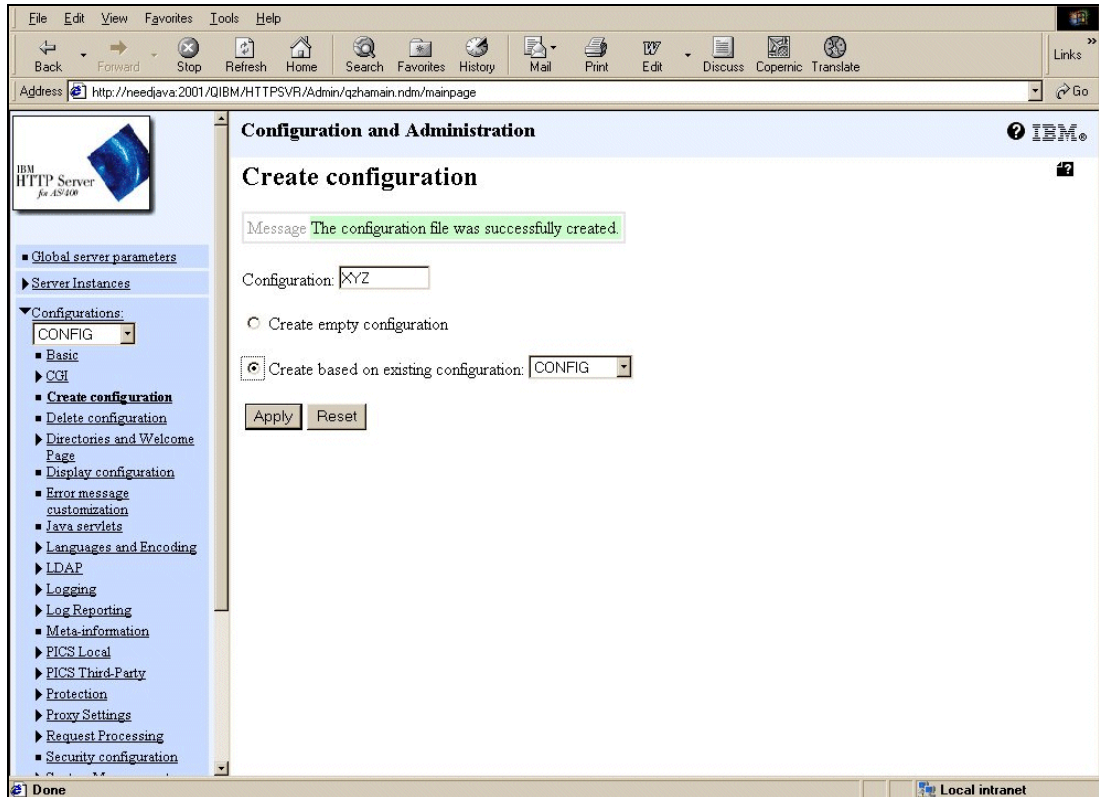
Figure 4-6   Create configuration page

Creating an instance

Now you must create a server instance:

1. On the left-hand panel, select **Server Instances** and then the **Create server instance** link.

2. The Create server instance page (Figure 4-7) appears. Follow these steps:

   a. Enter the server instance name, such as the NEW in our example.
   b. Enter the configuration that was just created, which is CC in this example.
   c. Click **Create**.
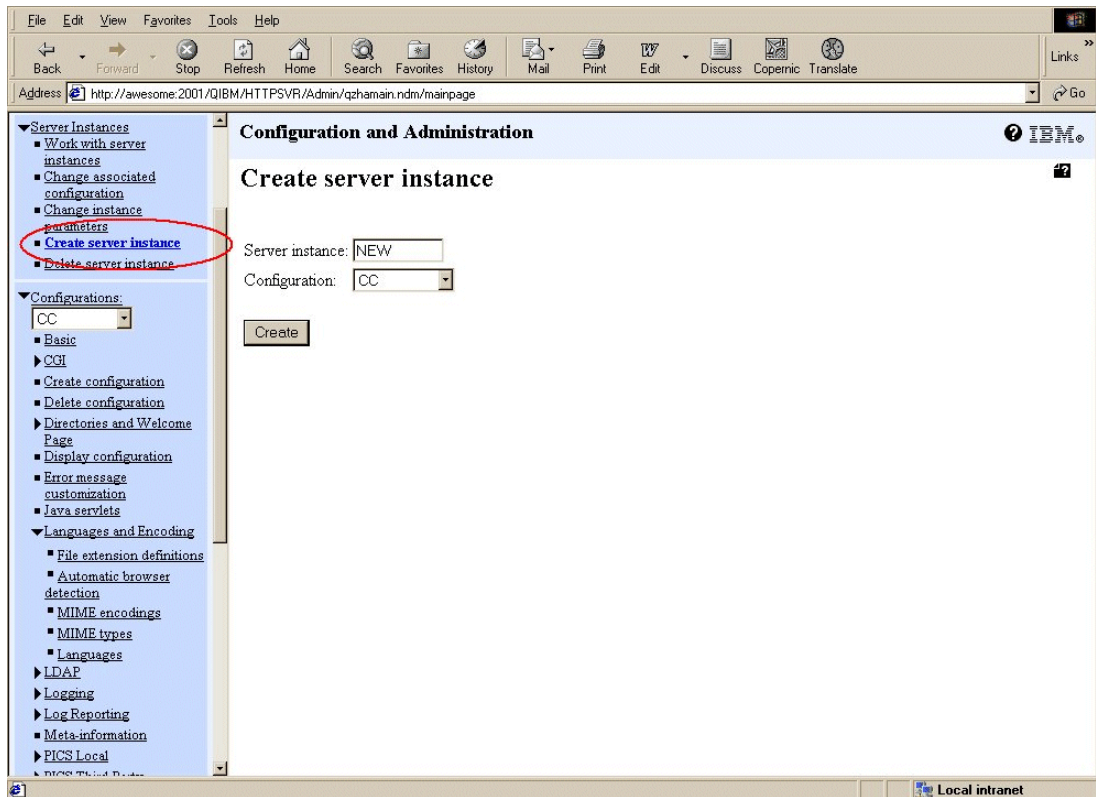
Figure 4-7   Create server instance page

You see the message "`Message The server instance was successfully created`".

In WRKACTJOB SBS(QHTTPSVR), you now see ADMIN jobs and jobs with the name of the instance that was just created. All should be in *wait* states. The first one should be in a *CNDW wait* state.

Setting up on the iSeries

This section explains how to set up journaling for certain files. It discusses special user profiles for vendor.connect that use a specifically configured job description to give vendor.connect its library list. This section also explains how to set up extra System21 data and database triggers for the required files.

Journaling

Because WebSphere applications run under commitment control, you must journal files that are used by vendor.connect. Since you must journal files from different libraries, you must create the journal receiver and the journal in a new library rather than in an existing System21 library. You must journal the files listed on the following page to operate vendor.connect. However, merely journaling these files does not give you full advantages of journaling such as extra security.

Performance improves if files are journaled to an auxiliary storage pool (ASP). The following steps provide commands for use with and without an ASP.

You must journal the following files at a minimum:

- All physical files in the OSLVCF3 library (except for files named XAPnn if they were there)
- The PMP02 and PMP09 files in the OSLD1F3 library
- The AGP00, AGP10, and AGP20 files in the OSLD2F3 library

To journal these files with *no ASP*, follow these steps:

> **Note:** The following commands are sample commands only. You may vary them as required.

1. Sign on as QSECOFR.
2. Create a new library for the journal receiver and journal such as OSLF3:

   ```
   CRTLIB LIB(OSLF3) TEXT('OSL journal library')
   ```
3. Create a journal receiver:

   ```
   CRTJRNRCV JRNRCV(OSLF3/OSL0001) THRESHOLD(50000) TEXT('System21 journal receiver.')
   ```
4. Create a journal:

   ```
   CRTJRN JRN(OSLF3/OSL) JRNRCV(OSLF3/OSL0001) MNGRCV(*SYSTEM) DLTRCV(*YES) TEXT('System21 journal')
   ```

To journal the files *using an ASP*, follow these steps:

1. Create a separate library for the journal receiver:

   ```
   CRTLIB LIB(OSLF3R) TEXT('OSL journal receiver library')
   ```
2. Create a journal receiver:

   ```
   CRTJRNRCV JRNRCV(OSLF3R/OSL0001) ASP(2) THRESHOLD(50000) TEXT('System21 journal receiver.')
   ```
3. Create the journal as shown in step 4 in the previous sequence of steps.

To journal a single file, such as ADDRESS, use the following command:

```
STRJRNPF FILE(OSLVCF3/ADDRESS) JRN(OSLF3/OSL) OMTJRNE(*OPNCLO)
```

To journal multiple files within a library, complete the following steps.

> **Note:** This is the recommended method.

1. Start the Programming Development Manager (PDM):

   ```
   STRPDM
   ```
2. Select option 9 (Work with user-defined options).

3. The Specify Option File to Work With display (Figure 4-8) appears. Press Enter to accept the defaults (File QAUOOPT, Library QGPL, and Member QAUOOPT).

```
                    Specify Option File to Work With

 Type choices, press Enter.

   File  . . . . . . . . . .   QAUOOPT      Name, F4 for list

     Library . . . . . . . .    QGPL        *LIBL, *CURLIB, name

   Member  . . . . . . . . .   QAUOOPT      Name








 F3=Exit     F5=Refresh     F12=Cancel

```
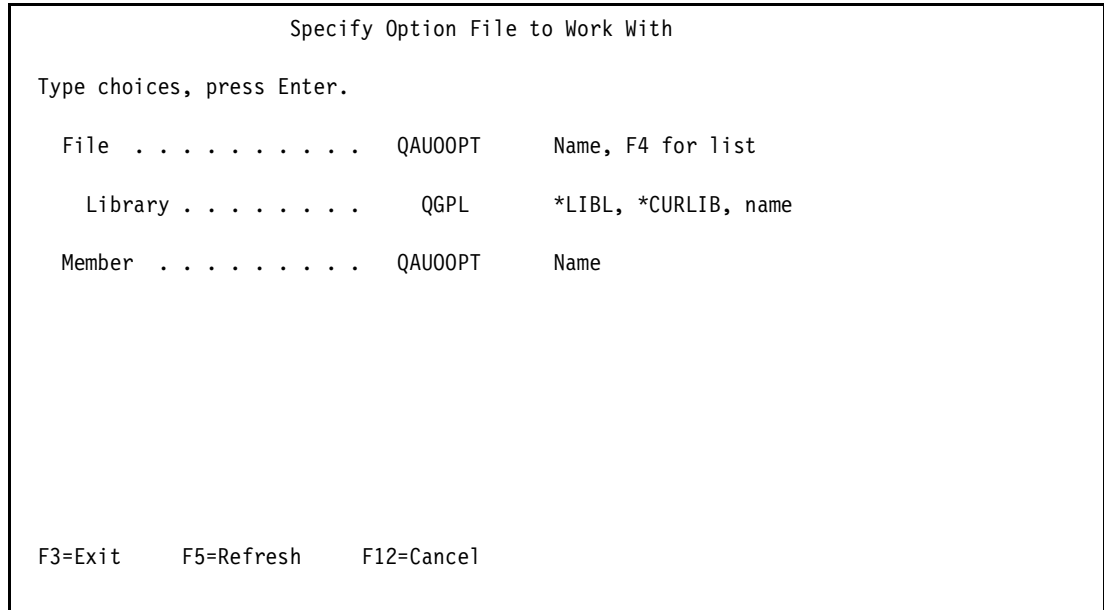
Figure 4-8   Working with the user-defined options

4. On the next panel, press F6 to create a new user-defined option.

5. On the Create User-Defined Option display (Figure 4-9), enter the following information:
   – Option: SJ
   – Command: STRJRNPF FILE(&L/&N) JRN(OSLF3/OSL) OMTJRNE(*OPNCLO)

   Press Enter.

6. You return to the list of user-defined options. Press F3.

```
                     Create User-Defined Option

 Type option and command, press Enter.

   Option  . . . . . . . . .   SJ   Option to create

   Command . . . . . . . . . STRJRNPF FILE(&L/&N) JRN(OMF3/OSL) OMTJRNE(*OPNCL
 O)




 F3=Exit       F4=Prompt       F12=Cancel

```
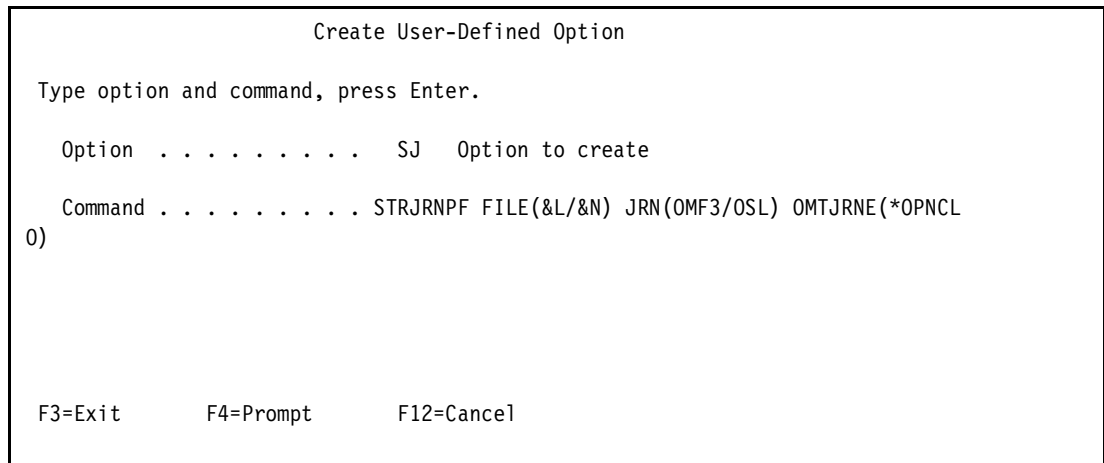
Figure 4-9   Create User-Defined Option display

7. You return to the PDM menu. Select option 2 (Work with objects).

8. On the next display, enter the library that contains the files to journal for example, OSLOMF3. Under object, set:

- Name: *ALL
- Type: *FILE
- Attribute: PF-DTA

9. On the following display, type SJ next to each file that you want to journal. To journal *all* physical files, type SJ on the first line and press F13.

User profiles

The vendor.connect library list is controlled by a job description attached to a special user profile as defined here.

► **Job description**: The vendor.connect library list is set within a job description OSLVCF3 in the OSLVCF3 library that is supplied as already configured. If you must use other .connect products, you must change this job description library list to include the extra libraries.

► **User profile**: You must create a user profile. As a standard, this should be called OSLVCF3. This user must have its job description set to OSLVCF3 as explained earlier. Set the current library to OSLVCF3. To create a user profile, enter the following command:

```
CRTUSRPRF USRPRF(OSLVCF3) CURLIB(OSLVCF3) TEXT(vendor.connect) JOBD(OSLVCF3/OSLVCF3)
```

System21 data setup

Follow these steps to set up the System21 data:

1. Using Data File Utility (DFU) or SQL, check the PM company profile record OSLD1F3/PMP10. Note the value of the RAVN10 field (Receiving Address Vendor Number).

2. Set the RECEIVINGADDRESSVENDOR field in OSLVCF3/COMPANYPROFILE to the value in RAVN10.

3. Set NEXTSHIPMENTNUMBER = 0 and update it.

4. Create a record in file OSLVCF3/UNIQUEIDENTIFIER for each company on PMP10 with file UNIQUEID = 0.

5. Check file OSLWFF3/WFP98 to ensure that it contains the following record:

```
HOST98    blank
PTAP98    MQ
PMRF98    TRIGGER.OUT
PMIF98    WM.TRIGGER.OUT
PVL198    0
PVL298    0
```

Database triggers

You must add database triggers to the following files:

► In library OSLD1F3: PMP02, PMP03, PMP06, and PMP09
► In library OSLPLF3: PLP05

To add database triggers, there must not be any locks on the file. For each file, you must add *INSERT, *UPDATE, and *DELETE triggers. Use the following command to add an *INSERT trigger to PMP02. Change the TRGEVENT parameter to create the *UPDATE and *DELETE triggers:

```
ADDPFTRG FILE(OSLD1F3/PMP02) TRGTIME(*AFTER) TRGEVENT(*INSERT) PGM(OSLWFF3/WF500A)
RPLTRG(*YES)ALWREPCHG(*YES)THDSAFE(*UNKNOWN)MLTTHDACN(*SYSVAL)TRGUPDCND(*ALWAYS)
```

The TRGUPDCND is only necessary for the *UPDATE trigger.

4.2.6 MQSeries

This section explains how to set up MQSeries for vendor.connect.

Creating a queue manager

You must first create a queue manager:

1. Enter the following command:

```
WRKMQM
```

If a default queue manager doesn't already exist, create one using the following command:

```
CRTMQM MQMNAME(VCQM) TEXT('vendor.connect queue manager') DFTQMGR(*YES)
```

Alternatively, press F6 on the WRKMQM display. You must create your queue manager as the default. You cannot change this at a later date. AIF database triggering only works with the default. We recommend that you call the queue manager the same name as the iSeries name, but it is not required.

2. Grant authorization to the MQSeries Manager by using the following command:

```
GRTMQMAUT OBJ(VCQM) OBJTYPE(*MQM) USER(*PUBLIC) AUT(*ALL)
```

3. On the WRKMQM display, select option 14 (Start the Work Queue Manager).

4. On the next display, select option 20 (Work with Channels).

5. On the Work with Channels display, create new channels with the following command:

```
CRTMQMCHL CHLNAME(WM.TRIGGER.CHL) CHLTYPE(*SVRCN) TRPTYPE(*TCP).
CRTMQMCHL CHLNAME(AIF.TRIGGER.CHL) CHLTYPE(*SVRCN) TRPTYPE(*TCP).
```

Alternatively, you can press F6 from the Work with Channels display.

6. Create new queues as follows:

```
CRTMQMQ QNAME(AIF.TRIGGER.OUT) QTYPE(*LCL) MQMNAME(*DFT)
CRTMQMQ QNAME(WM.TRIGGER.OUT) QTYPE(*LCL) MQMNAME(*DFT)
CRTMQMQ QNAME(WM.TRIGGER.OUT1) QTYPE(*LCL) MQMNAME(*DFT)
```

7. Grant authority to all the queues with the command:

```
GRTMQMAUT OBJ(queue name above) OBJTYPE(*Q) USER(*PUBLIC) AUT(*ALL) MQMNAME(*DFT)
```

Alternatively from WRKMQM, select option 18 (Work with queues) and press F6.

To ensure that MQSeries is automatically restarted if the machine is shut down for any reason, change the machine's IPL procedure to include the following commands:

```
STRSBS SBSD(QMQM/QMQM)
STRMQM
STRMQMCSVR
STRMQMLSR
```

Note that because the commerce.platform software currently only works with the MQSeries queue manager set as the default queue manager, no parameters are needed for the STRMQMxxx commands.

If all the subsystems are routinely taken down for backups, etc., then you must update the subsystems restart to also execute the previous four commands.

4.2.7 Work Management Trigger Handler for the iSeries

The OSLVCD3/TRGRHNDLR CL program is provided to start the Work Management Trigger Handler. The program runs a shell script, which is found in the vendorconnect folder on the iSeries (trigger.sh). To run the CL program, enter:

```
CALL OSLVCD3/TRGRHNDLR
```

After the program runs, the following three jobs should be running within QEJBSBS:

► QP0ZSPWP
► QZSHSH
► VCTRIGGERS

These files may be identified because they will run under the user profile of whoever started them. You should submit this program to run automatically as part of any day start routines or after an IPL.

4.2.8 Active Architecture framework

The Active Architecture framework handles the processing of the triggers in two steps. The publisher and the controller between them convert the triggers so that they are sent to the correct Java processor.

Publisher

The publisher process picks up the raw database trigger messages placed on MQSeries. TRIGGER.OUT queue by the OSLWFF3 and WF500A programs. Then it converts them to XML messages and puts them on the AIF.TRIGGER.OUT queue.

Verifying whether folders have write authority

You should load the AEF, com, and com.geac.erp.system21.aef folders on the iSeries so that they are directly off the root directory in the file structure. Use the WRKLNK command to ensure that these folders are all writable:

```
WRKLNK '/<directory>'
```

Select option 9 to view authority. Assign *PUBLIC *RWX authority.

Checking the properties file

Check AEF/TriggerHandler.properties file to ensure that the MQSeries names correspond to those that are set up in the MQSeries setup in 4.2.6, "MQSeries" on page 88.

Updating the event documents

There are events (XML documents under the com.geac.erp.system21.aef/xml folder) that filter out any unwanted messages. To reduce the amount of data initially held in the vendor.connect database, these events filter based on the suppliers that currently use vendor.connect. This enables a vendor.connect user to gradually roll out the system to more suppliers, until the majority are using it, at which time the Supplier Filter Rule (but not the other rules, which are an integral part of the operation of the system) can be removed completely.

These events are machine specific. Therefore, you must change the folder names to match the same name as your iSeries server.

Changing the folder names

There are two folder names that you must change. One is under com.geac.erp.system21.aef/xml/rules, and one is under

com.Geac.erp.system21.aef/xml/maps. Inside these folders, there are additional folders that represent the iSeries libraries where the System21 PMPxx and PLPxx tables reside. You must also change these folder names to match your libraries (see Figure 4-10).
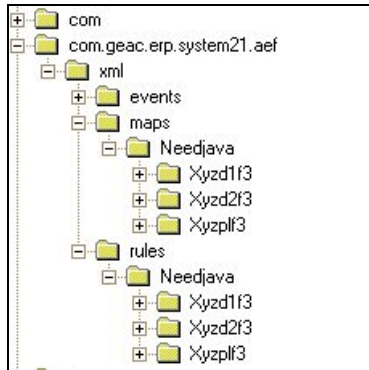
Figure 4-10   File structure for libraries named Xyz…

Updating the events rules

There are two ways to update these event rules:

► Edit the XML documents provided (preferably using Notepad). *Do not* use an editor that reformats the XML, for example, one that adds carriage returns and formatting such as XMLSpy.

► If you have @ctive Modeler, import the vendorconnect Business Process (found in the /bpr directory) into @M. Click the **Events** bar in the Palette. If this is not visible, right-click the **Palette** bar and select the **Show WM Controls** option.

Edit the relevant event (by double-clicking the event, double-clicking the rule, double-clicking the VNDRxx test, and changing the list of suppliers and saving it), and re-activate the changed event.

If you are editing the XML documents directly (option 1), proceed as explained here:

1. Change all the rules for PMP02, PMP03, and PMP09 (in /com.geac.erp.system21.aef/xml/rules). There are nine events or XML files to edit or reactivate in total.

2. In each event, edit the VNDR comparison test to specifically include the list of suppliers that are in use.

3. Change the list of dummy suppliers abc, def, and ghi (see the following example) into the real list of supplier IDs.

4. Change the VDNRxx reference to match the 02, 03, or 09 file to which the rule applies:

```
<tests>
  <test>
      <format>1</format>
      <format_string>COMPARISON</format_string>
      <field_id>DTLC02</field_id>
      <field_scope>1</field_scope>
      <field_scope_string>AFTER</field_scope_string>
      <diff_field_id />
      <diff_field_scope>1</diff_field_scope>

          <diff_field_scope_string>AFTER</diff_field_scope_string>
      <condition>1</condition>
      <condition_string>EQ</condition_string>
      <value>9999999</value>
```

```
            <value_format>1</value_format>
            <value_format_string>STATIC</value_format_string>
            <value_scope>1</value_scope>
            <value_scope_string>AFTER</value_scope_string>
        </test>
        <test>
            <format>1</format>
            <format_string>COMPARISON</format_string>
            <field_id>VNDRO2</field_id>
            <field_scope>1</field_scope>
            <field_scope_string>AFTER</field_scope_string>
            <diff_field_id />
            <diff_field_scope>1</diff_field_scope>

            <diff_field_scope_string>AFTER</diff_field_scope_string>
            <condition>9</condition>
            <condition_string>IN</condition_string>
            <value>abc,def,ghi</value>
            <value_format>1</value_format>
            <value_format_string>STATIC</value_format_string>
            <value_scope>1</value_scope>
            <value_scope_string>AFTER</value_scope_string>
        </test>
    </tests>
```

Controller

This process reads the XML triggers from the AIF.TRIGGER.OUT queue and sends them to the relevant Java processor, based on the header information contained in each XML document.

1. Edit the /AEF/controller.properties file to reflect the iSeries name and the MQSeries Manager that was setup earlier on the iSeries.

2. A shell script is provided to start the AIF controller (/vendorConnect/aifcntrlr.sh). On the last line, insert your iSeries name and the port for the instance of WebSphere that you are using:

```
java -Djava.naming.provider.url=iiop://<iSeries>:<port number>
-Djev.config=/VendorConnect/deployed/ejb_default.cfg

com.geac.erp.system21.aif.toolkit.Controller -p Controller.properties -v
```

3. Call the OSLVCD3/AIFCNTRLR CL program. The program runs the script. To run the CL program, enter:

```
CALL OSLVCD3/AIFCNTRLR
```

When this has run, the following three jobs should be running within QEJBSBS:

► QP0ZSPWP
► QZSHSH
► AIFCNTRLR

They may be identified because the will be running under the user profile of whomever started them.

You may want to schedule this job to stop and start daily or after power downs. However, note that there should be a delay prior to starting this process to allow WebSphere to start the various application servers first. You may achieve this by creating a CL program to start WebSphere and then call AIFCNTRLR with a DLYJOB command in between. The delay may need to be several minutes. You may have to experiment to see what is necessary.

4.2.9 JConnects server

This links vendor.connect back in to System21. Copy the Jacada directory to the root of your iSeries server. Make sure the files are writable by running the WRKLNK command as explained earlier.

ConnectorManager.xml

Edit the /VendorConnect/deployed/ConnectorManager.xml file. Change the following entries as appropriate to your iSeries setup:

```
<company>Z1</company>
<environment>XXX</environment>
.
.
.

<host>your iSeries</host>
.
.
.

<user>userid</user>
<password>password</password>
```

The user here *must* have access to the 2/AG option within System21 for the company also specified here. This option must be defined with a date format of D in Application Manager.

AIF .ini files

1. Open the **/Jacada/SYS21AIF/classes/appls/SYS21AIF/server/resources/sys21aif.ini** file.

2. At the start of the file, ensure that the RtRootDir is correct:

   ```
   RtRootDir=/JACADA/SYS21AIF/
   ```

3. At the end of the file, change the Host statement to have the correct iSeries name:

   ```
   After [GUISys TN5250]
   .
   Host=<youriSeries>
   ```

4. Open the **/Jacada/SYS21AIF/classes/jacadasv.ini** file.

5. Ensure that the RtRootDir value is the same as above.

4.2.10 Setting up new vendor.connect user IDs and supplier IDs

There are two modes in which secure.connect can be run. The first holds the user details as an XML file called /vendorconnect/deployed/UserDirectory.xml. The second holds the user details in IBM's version of Lightweight Directory Access Protocol (LDAP) called *SecureWay*. Obviously, the latter method is the more secure, but it requires an additional process to run, either on the workstation or on the iSeries.

SecurityManager.xml

You can configure the system to use either method by changing the /vendorconnect/deployed/SecurityManager.xml file.

vendor.connect uses the authentication context named "DEFAULT" in the SecurityManager.xml file. This default context, as shipped, is configured to use an LDAP directory. However, you should verify the following settings:

```
<authentication_context id="DEFAULT">
    <directory id="SecureWay" type="LDAP">
```

```
        <property id="ldap_server" value="ldap://server.domain.co.uk/" />
        <property id="ldap_base" value="ou=ges,o=geac,c=uk" />
        <property id="ldap_bind_dn" value="cn=root" />
        <property id="ldap_bind_password" value="password" />
        <property id="authorization_group_type" value="businesscategory" />
    </directory>
</authentication_context>
```

Change the ldap.server parameter to the name and domain of the iSeries that is running the LDAP service. You may also change other LDAP properties if required, for example, LDAP base.

XML security

If an LDAP server is not present, then you can configure the DEFAULT context to use /vendorconnect/deployed/UserDirectory.xml as the user directory. Simply edit the following settings in SecurityManager.xml:

```
<authentication_context id="XML">
    <directory id="file" type="XML">
        <property id="file" value="UserDirectory.xml" />
    </directory>
</authentication_context>


<authentication_context id="DEFAULT">
    <directory id="SecureWay" type="LDAP">
        <property id="ldap_server" value="ldap://server.domain.co.uk/" />
    etc.
```

Change the first authentication context id="XML" to context id="DEFAULT". Since the context ID must be unique within the file, change the second authentication context ID immediately to anything other than DEFAULT as shown in the following example:

```
<authentication_context id="DEFAULT">
    <directory id="file" type="XML">
        <property id="file" value="UserDirectory.xml" />
    </directory>
  </authentication_context>


<authentication_context id="XYZ">
    <directory id="SecureWay" type="LDAP">
        <property id="ldap_server" value="ldap://server.domain.co.uk/" />
    etc.
```

Create the vcadmin user in UserDirectory.xml. Add the following section to this file, entering the user password as required:

```
<entry dn="cn=vcadmin, ou=cad, ou=ges,o=geac,c=uk">
    <objectclass>
        <oc-value>top</oc-value>
        <oc-value>person</oc-value>
        <oc-value>organizationalPerson</oc-value>
        <oc-value>inetOrgPerson</oc-value>
        <oc-value>ePerson</oc-value>
        <oc-value>s21User</oc-value>
    </objectclass>
    <attr name="cn"><value>vcadmin </value></attr>
    <attr name="sn"><value>Administrator </value></attr>
    <attr name="uid"><value>vcadmin</value></attr>
    <attr name="userpassword"><value>anything</value></attr>
  </entry>
```

Using LDAP

If you are using LDAP, you must install an IBM SecureWay client to create the administrator user account.

If you are using Geac's commerce.platform, the following settings may already be configured, but you should still check them. Edit the /vendorconnect/deployed/SecurityManager.xml file. Check the authorization_context section:

```
<authorization_context id="DEFAULT">
   <database id="author">
      <property id="db_driver" value="com.ibm.as400.access.AS400JDBCDriver" />
      <property id="db_url" value="jdbc:as400://<iSeries>/authsystem;trace=false" />
      <property id="db_user" value="oslvcf3" />
      <property id="db_password" value="oslvcf3" />
   </database>
  </authorization_context>
```

Set the iSeries name and verify property id="db_user" and "db_password" is OSLVCF3.

To access the user maintenance display, you must create a user with a user ID of "vcadmin" in the security directory. You must do this using the IBM SecureWay client if you are using LDAP, or you can manually add it to the UserDirectory.xml file.

Configuring the LDAP server

You must have installed Client Access Express V4R5 or later on your workstation. To configure the LDPA server, follow these steps:

1. In Operations Navigator, select the iSeries that is required and expand that node.

2. Sign on as security officer.

3. Select **Network-> Servers** and open the **TCP/IP** folder (Figure 4-11).
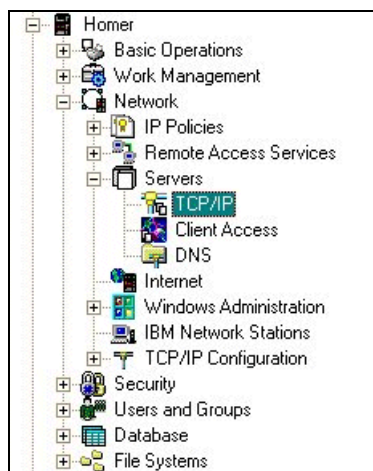
Figure 4-11   Operations Navigator view

4. Within TCP/IP, under Server Name, select **Directory** (these are not in alphabetical order), right-click, and select **Properties**.

5. The Directory Properties window (Figure 4-12) opens. Complete these tasks:

   a. Select the **Start server when TCP is started** box.
   b. Set Administrator name as `CN=root`.
   c. Click the **Password** button. Enter and confirm your password.
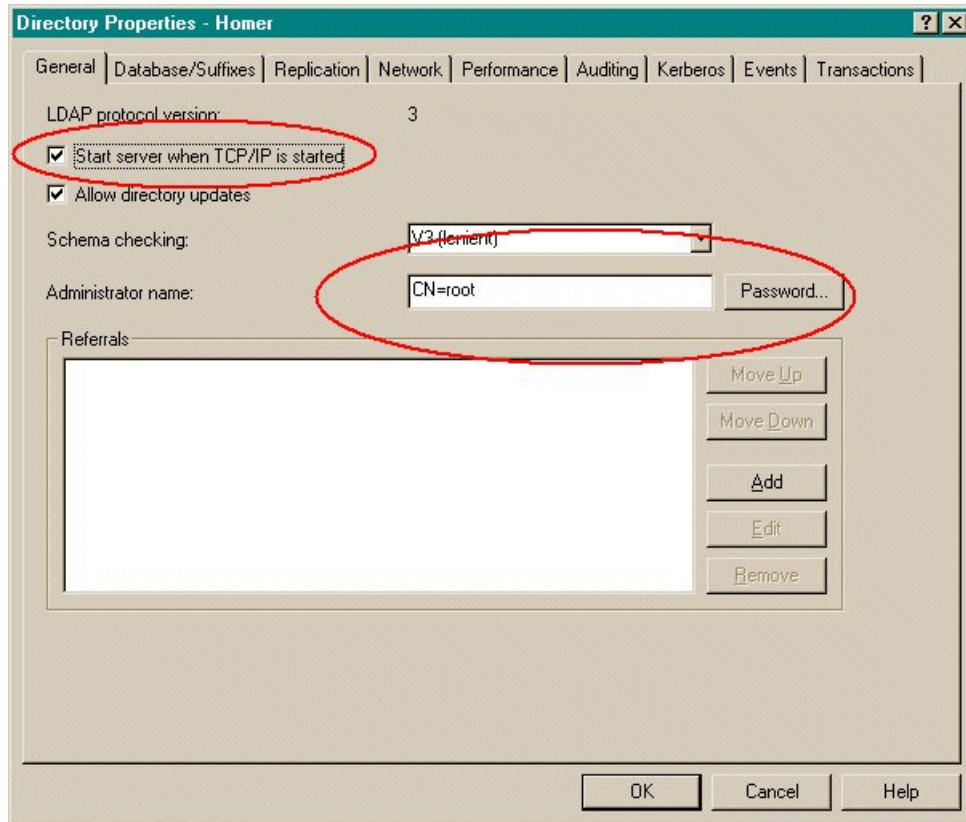
Figure 4-12   Directory Properties: General page

6. Click the **Databases/Suffixes** tab (Figure 4-13).

   a. Under Suffixes, enter a new suffix of:

      `ou=xxx,o=yyy,c=zzz,`

      Note the following explanation:

      ou Organization unit
      o   Organization
      c   Country (region)

   b. Click **Add**.
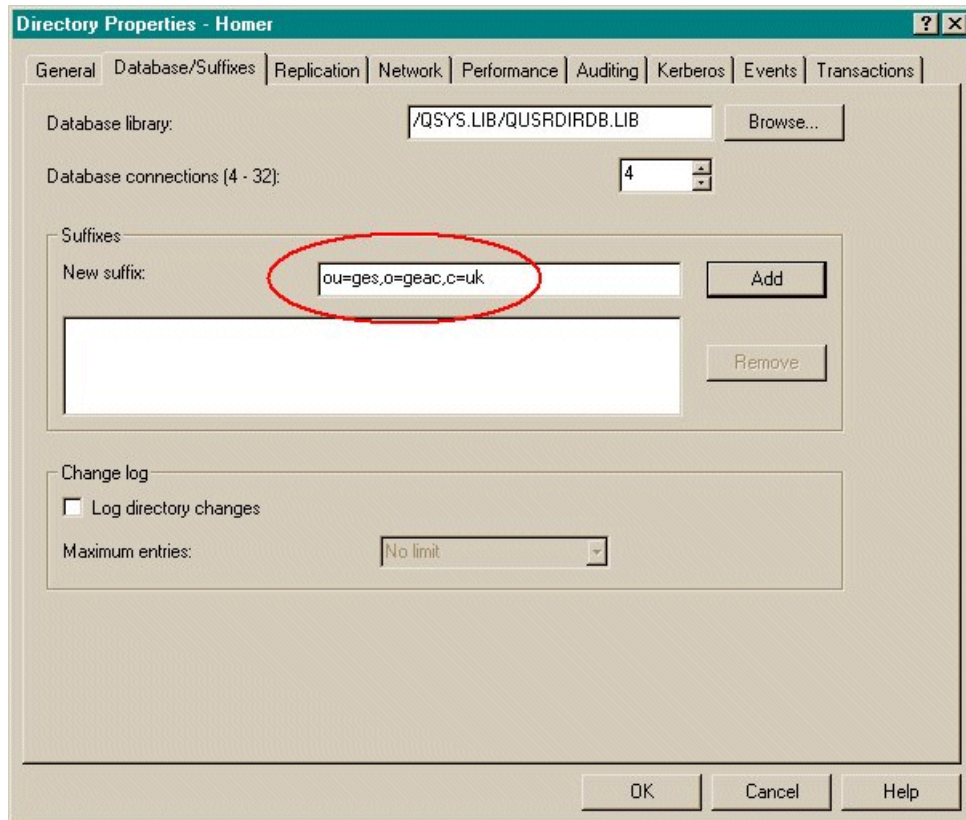
Figure 4-13   Directory Properties: Database/Suffixes page

Note: Password protection is set to SHA by default. This means that it cannot be decrypted. If a password is forgotten, it cannot be recovered. However, the security officer can create a new one. You can change this to "crypt" by selecting the **Network** tab and then selecting **Advanced**. In this case, you can view passwords. We recommend that you *do not* set this to *None*.

Using IBM SecureWay

Follow these steps:

1. Open C:\Program Files\IBM\LDAP\etc\dmt.conf (assuming a standard install) using Notepad.

2. Amend the following line to show your iSeries:

   `server1.url=ldap://<iSeries>:389`

If you don't do this and leave it set to localhost, you will receive a warning about attempting to contact LDAP on local machine. You may ignore this message.

3. Select **Start-> Programs-> IBM SecureWay Directory-> Directory Management Tool**. If this is the first time this has been used, you receive a warning message about the suffix not containing any data. You may ignore this message.

4. Click the **Add server** button at the bottom of the left-hand panel. On the Add directory server panel (Figure 4-14), complete these tasks:

   a. Enter the server name (needjava in this example).
   b. Leave the port as 389.
   c. Enter a user DN and user password.
   d. Click **OK**.
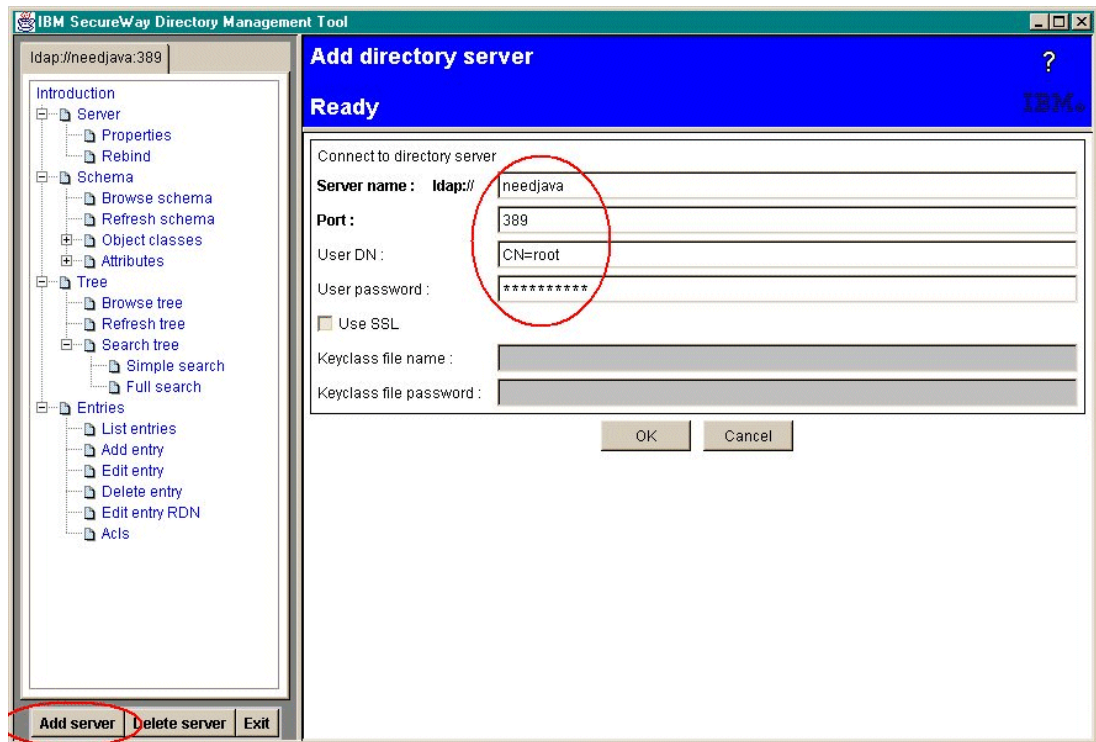
Figure 4-14   Add directory server panel

5. In the left-hand panel, select **Server-> Rebind**. On the Rebind to server panel (Figure 4-15) on the right-hand side, complete these tasks:

   a. Select the **Authenticated** option.
   b. Enter `CN=root` for User DN.
   c. Enter the password you entered on the Add directory server display.
   d. Click **OK**.
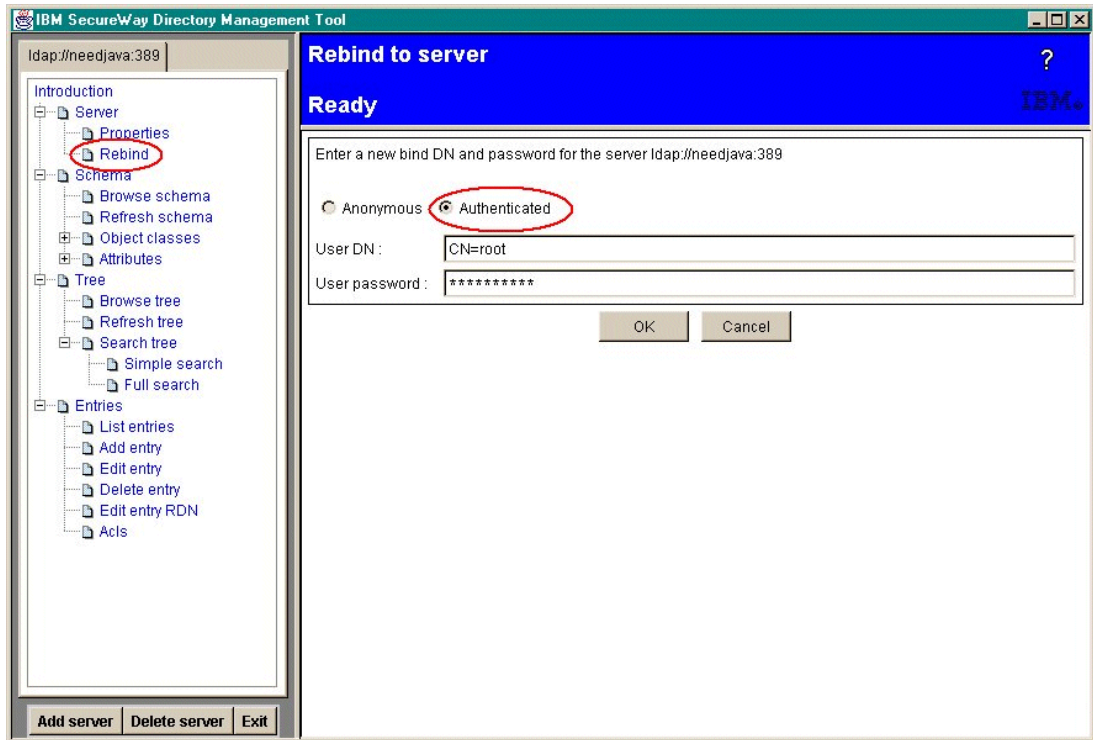
Figure 4-15   Rebind to server panel

6. Select **Entries-> Add Entry** in the left-hand panel. On the Add directory entry panel (Figure 4-16) on the right-hand side, complete these tasks:

    a. Leave the Parent DN field blank.
    b. In Entry RDN, enter the full name of your suffix (for example, `ou=ges,o=geac,c=uk`).
    c. Select the **Organizational unit** option.
    d. Click **Next**.
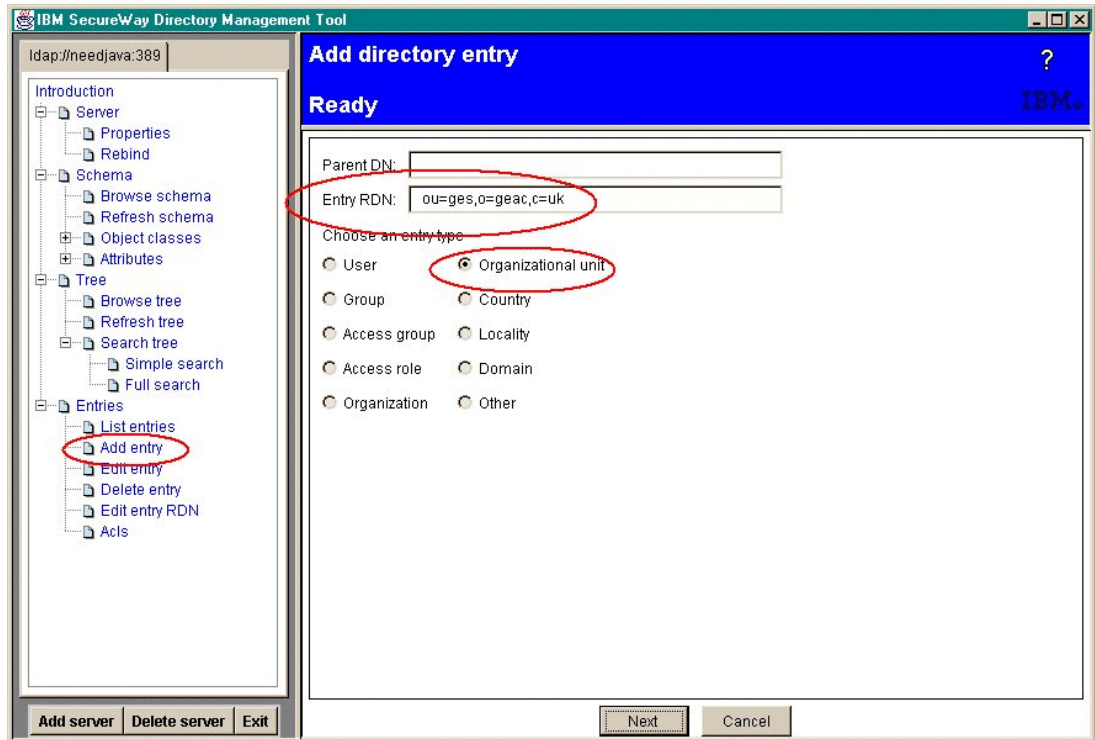
Figure 4-16   Adding an entry

7. For ou, enter (again) the organizational unit (ges). Click **Create**.

Creating the vcadmin user in IBM SecureWay

To create the vcadmin user, follow these steps:

1. Select **Tree-> Browse tree** from the left-hand panel. In the Browse directory tree panel (Figure 4-17), complete these tasks:

   a. Select the directory base (for example, "ou=ges,o=geac,c=uk") so that it is highlighted.
   b. Click the **Add** button at the top of the panel.
   c. In the Entry RDN field, enter `cn=vcadmin`.
   d. Click **Next**.
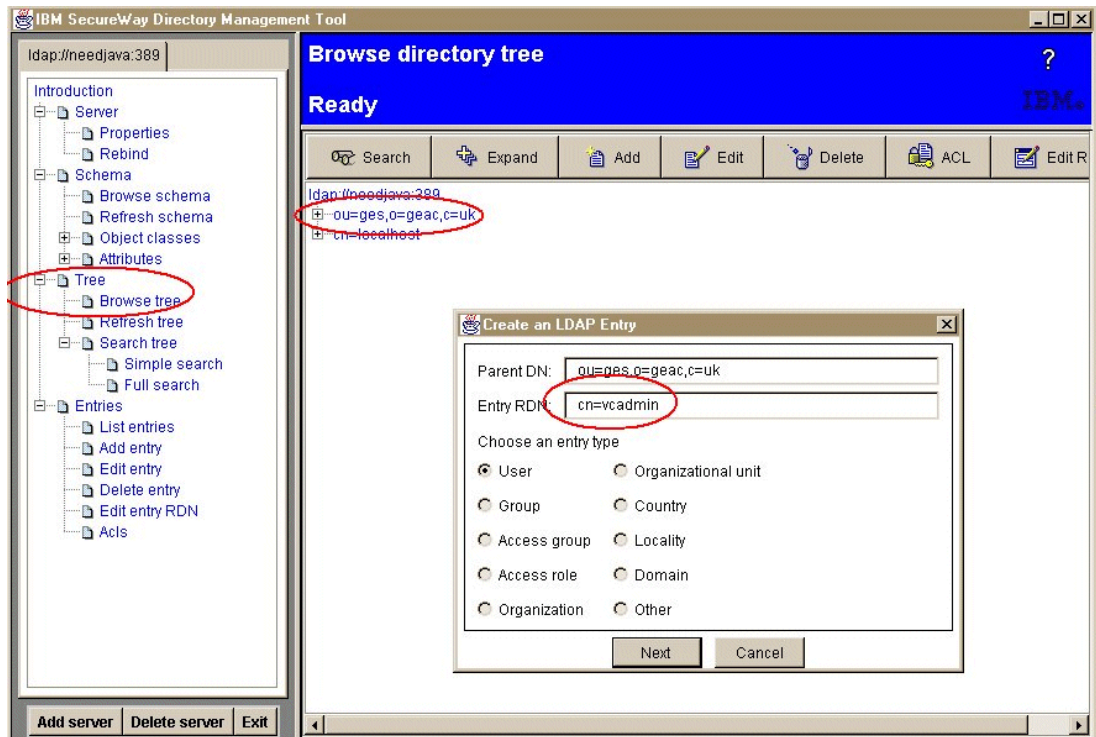
Figure 4-17   Creating the vcadmin user

2. In the create an LDAP User window (Figure 4-18), complete these tasks:

   a. For Common name, enter `vcadmin`.
   b. For Last name, enter `Administrator`.
   c. Click the **Business** tab. Scroll down to the bottom of the page, and enter a password in the userPassword field.
   d. Click the **Other** tab. Scroll down and locate the uid field, and enter `vcadmin`.
   e. Click **Create**.

Figure 4-18   Creating vcadmin detail showing the Other tab

Creating the authorization rule

The authorization rule is used by vendor.connect to match users with their authorized supplier code. If Geac's secure.connect is installed on the iSeries in the files that contain the rule, the rule will already be located in the AUTHSYSTEM library. If secure.connect is not installed, then you need to restore the AUTHSYSTEM library on the iSeries server. Refer to 4.2.3, "Restoring libraries" on page 76, for instructions on restoring this library from its save file.

Use your preferred SQL console (for example, Client Access or on the iSeries) to execute the statements in the VendorCreateRule.SQL file in the SQL folder. Make sure that the library AUTHSYSTEM appears in the library list when running this.

1. Log on to the vendor.connect Web site:

   `http://<iSeries>/VendorConnect/logon.jsp`

2. For all further user maintenance, log on to the Web site as user `vcadmin` as shown in Figure 4-19.

Figure 4-19   Logging on as vcadmin

3. Logging into the vendor.connect system as the vcadmin user takes you to the User Maintenance functions, rather than the standard supplier's Web site.

   The initial user maintenance page shows any existing users. Clicking an existing entry allows you to edit the user profile. Click **Add** to add a new user.

4. Enter a valid supplier code and address code, as well as the user ID, contact name, surname, and password to add a user as shown in Figure 4-20. Then click **Save**.

Figure 4-20   Setting up a new user

4.2.11 Database synchronization from System21 to the vendor.connect database

To set up the correct library lists needed for the database triggers to fire successfully, you must set the library list. You can do this manually, for example, using the Edit Library List (EDTLIBL) command. Or you can do this by selecting a System21 option that has the workflow (OSLWFF3) library in its list (for example, 41/PMP). Leave the option, but stay in the System21 menus and then use the STRSQL command.

If you are using Client Access SQL, follow these steps to set the library list:

1. Set the library list by connecting to the iSeries.

2. Select **Connection-> JDBC Setup** as shown in Figure 4-21.
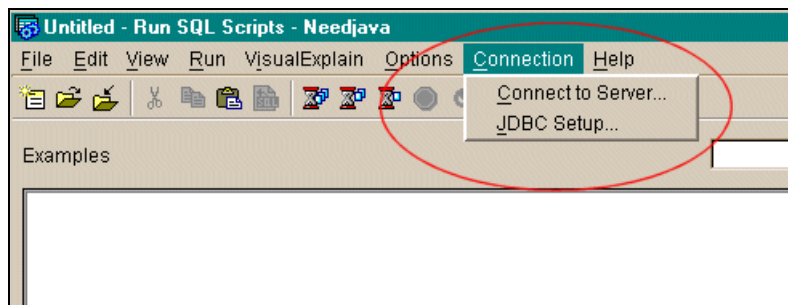
Figure 4-21   Selecting JDBC Setup in Client Access

3. To set up a library list within the Client Access SQL tool, see the example in Figure 4-22.
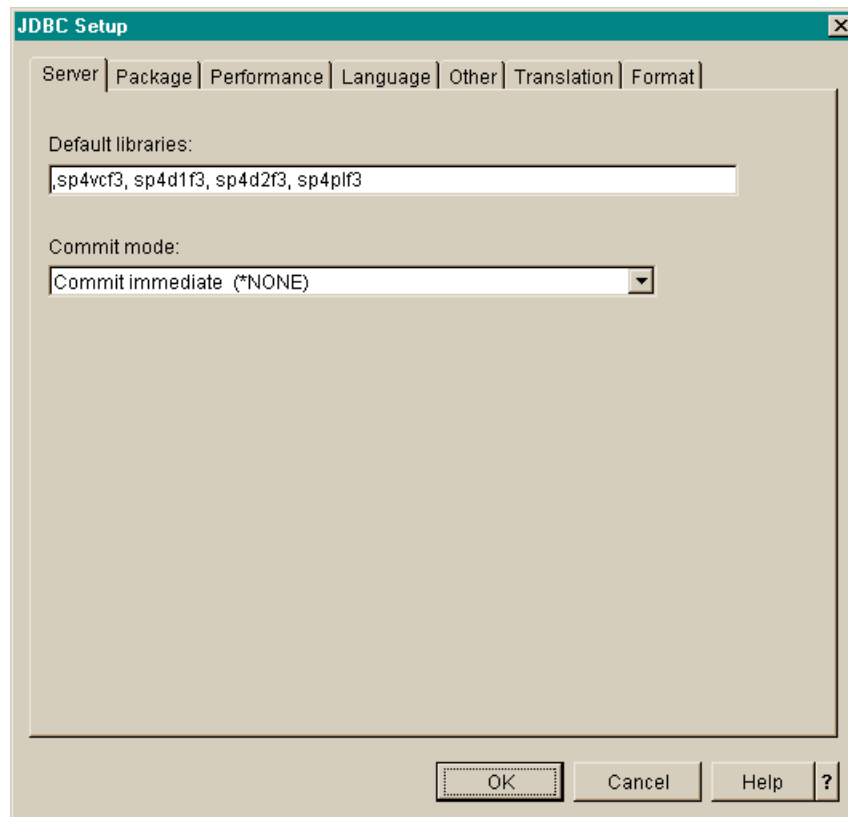
Figure 4-22   Setting the library list in Client Access

4. Run the set of SQL commands that are detailed in the following SQL example. They cause triggers to be fired so that the System21 data will be synchronized into the VendorConnect database. You can find the SQL in the SQL to sync dbs on initial load.txt file in the SQL folder.

5. If the amount of data to be synchronized is large, you may need to edit the SQL to sub-select the data. After messages in the MQSeries queues subside, you can run the next sub-select.

Here is an example of SQL to be run for company Z1 and suppliers GS02 and FS03:

```
Planners:
update pmp06 set cono06 = 'Z1'
where cono06 = 'Z1'


Suppliers and Receiving Addresses:
update plp05 set cono05 = 'Z1'
where cono05 = 'Z1' and
supn05 in ('GS02','FS03', 'PMSUP')
```

> **Note:** Remember to include the "dummy" receiving address supplier ID in the SUPN05 list, usually 'PMSUP', because this file holds both supplier and receiving address details (see "System21 data setup" on page 87).

```
Purchase Order Headers:
update pmp02 set cono02 = 'Z1'
where cono02 = 'Z1' and
dtlc02 = 9999999 and
ordn02 in
(select distinct ordn03
from pmp03 where cono03 = 'Z1' and
vndr03 in ('GS02','FS03') and
qtos03 > 0 group by ordn03)


Purchase Order Lines:
update pmp03 set cono03 = 'Z1'
where cono03 = 'Z1' and
dtlc03 = 9999999 and
bosn03 <> 99999 and
ordn03 in
(select distinct ordn03
from pmp03 where cono03 = 'Z1' and
vndr03 in ('GS02','FS03') and
qtos03 > 0 group by ordn03)


Purchase Order Transactions-Expected Receipts, Actual Receipts and Returns:
update pmp09 set cono09 = 'Z1'
where cono09 = 'Z1' and
ordn09 in
(select distinct ordn03
from pmp03 where cono03 = 'Z1' and
vndr03 in ('GS02','FS03') and
qtos03 > 0 group by ordn03)
```

4.2.12 Testing the vendor.connect Web site

You are now in a position to test the Web application. Launch Internet Explorer and enter the following URL:

```
http://<iSeries>/VendorConnect/logon.jsp
```

4.2.13 Backing up the configuration components

If all is well, you should back up the configured components, so that they can be restored in case of any emergency. Several folders must be saved (see the list of components to install in 4.2, "Installing vendor.connect" on page 74).

The following command shows how to back up the AEF and VendorConnect folders. You need to extend the command to include all folders:

```
SAV DEV('/qsys.lib/tap01.devd') OBJ(('/AEF') ('/VendorConnect'))
```

4.3 Changing the iSeries on which the application is running

If you want to change the iSeries on which the vendor.connect application is running, change the following files. You also need to change these files if you want to change the iSeries username or password:

- ► /AEF/controller.properties
- ► /AEF/standard.properties
- ► /AEF/TriggerHandler.properties
- ► /VendorConnect/deployed/ConnectorManager.xml
- ► /VendorConnect/deployed/ejb_default.cfg
- ► /VendorConnect/deployed/SecurityManager.xml
- ► /VendorConnect/deployed/standard.properties

You must also change the database trigger rules folder name and maps folder name in the /com.geac.erp.system21.aef/xml/rules and /com.geac.erp.system21.aef/xml/maps folders.

Also, go to the WebSphere Administrator's Console, click the **jt400BTT** data source name, and change the database name to the new iSeries ID.

You should also change the first two files above if you change your MQSeries setup (for example, change the default queue manager name, or change the queue/channel names).

4.4 Housekeeping

Once vendor.connect is running, it requires little housekeeping. However, you may want to consider the items that are explained in the following sections.

4.4.1 Daily

Back up the System21 files libraries as usual, including library OSLVCF3.

4.4.2 Stopping

This section offers helpful information for situations where the iSeries is powered down or you want to stop and start WebSphere overnight.

Under normal circumstances, the application (for example, VennConn) continues to run in WebSphere when the admin job QEJBADMIN is stopped. This means that to stop the system overnight, the easiest way is to end QEJBSBS by using the following command:

```
ENDSBS SBS(QEJBSBS) OPTION(*IMMED)
```

This is a standard, tested iSeries command. This has the advantage of ending the Trigger Handler, AIF controller jobs, and any extra WebSphere instances. Users who want to end the subsystem more gently, may want to use *CNTRLD end with a time limit. If you do not specify a time limit, the subsystem will never end, because the Trigger Handler and AIF controller jobs keep running.

As an alternative for stopping and starting jobs, use XMLConfig as in a WebSphere configuration. However, keep in mind that this method has not yet been fully tested by Geac.

4.4.3 Starting

To start vendor.connect assuming the VennConn, VSSync, and Receiving applications are still running, you must follow the steps to start WebSphere using either a default instance or other instances as explained in the following sections.

Starting WebSphere (default instance)

With a default instance, you simply start QEJBSBS by using the following command:

```
STRSBS SBSD(QEJB/QEJBSBS)
```

This starts QEJBADMIN, QEJBMNTR, and the VennConn, VCSync, and Receiving applications. It requires some time, possibly several minutes, to start.

Starting WebSphere (other instances)

If you need to start more than one instance of WebSphere, you may use the STRWSSVR command. The following example starts an instance called VC:

```
CALL PGM(OSLOMD3/STRWSSVR) PARM(VC)
```

Note: You may start other instances while the default instance is still starting.

Starting the trigger handler and AIF controller

You may start the trigger handler and AIF controller with the following commands:

```
CALL PGM(OSLVCD3/TRGRHNDLR)
CALL PGM(OSLVCD3/AIFCNTRLR)
```

It is possible to run these jobs in Auto Day Start and Auto Day End under Machine Manager in System21 or in some other automated scheduled way on the iSeries. If you do this, you must allow sufficient time between starting WebSphere and starting the AIF controller. We recommend that you wait at least 10 minutes. You can do this by submitting the commands through a CL program and inserting the Delay Job (DLYJOB) command in between them.

4.4.4 Restoring the vendor.connect IFS objects

If for any reason you need to restore the IFS objects for vendor.connect, use the following commands. In this example, only the AEF and VendorConnect folders are restored. You may extend the command to restore other folders.

```
RST DEV('/qsys.lib/tap01.devd') OBJ(('/AEF')('/VendorConnect'))
```

This assumes that the system is backed up in a similar way.

5

Performance tuning

This chapter covers performance tuning of Geac System21 products.

5.1 Hardware

When you tune your configuration and system for performance, always start with the hardware requirements for the configuration. For applications that use Enterprise JavaBeans (EJBs), as both call.connect and vendor.connect do, you must be running one of the following models with the corresponding feature code:

- ► Model 170 feature 2385
- ► Model 720 feature 2062
- ► Model 270 feature 2252
- ► Model 820 feature 2396

For all models, we recommend that you meet the minimum memory requirement of 1 GB.

To check the processor model and feature and the memory of your iSeries, use the following command:

```
WRKHDWRSC *PRC
```

Then the Work with Processor Resources display (Figure 5-1) appears. In this example, the model is shown against the resource CEC01. The feature is against MP01. You may need to add multiple memory cards. If necessary, page down to see the complete list. The system shown is a Model 170, with feature 2385 and 2 GB of memory. This satisfies the processor requirements and more than satisfies the memory requirements.

```
                        Work with Processor Resources
                                                    System:    HOMER
Type options, press Enter.
  7=Display resource detail

Opt    Resource        Type-model   Status           Text
  ▋    CEC01            9406-170    Operational      Main Card Enclosure
  _    PN01             2468-001    Operational      System Control Panel
  _    MP01             2385-000    Operational      System Processor Card
  _    SP01             0797-003    Operational      Service Processor Card
  _    BCC01                        Operational      Bus Extender
  _      MS01           3002-000    Operational      128MB Main Storage Card
  _      MS02           3002-000    Operational      128MB Main Storage Card
  _      MS03           3002-000    Operational      128MB Main Storage Card
  _      MS04           3002-000    Operational      128MB Main Storage Card
  _      MS05           3003-000    Operational      256MB Main Storage Card
  _      MS06           3003-000    Operational      256MB Main Storage Card
  _      MS07           3003-000    Operational      256MB Main Storage Card
  _      MS08           3003-000    Operational      256MB Main Storage Card
  _      MS09           3003-000    Operational      256MB Main Storage Card
  _      MS10           3003-000    Operational      256MB Main Storage Card
                                                                   Bottom
F3=Exit    F5=Refresh    F6=Print    F12=Cancel
```

Figure 5-1   Work with Processor Resources display

You can also check the model by using the Display System Value (DSPSYSVAL QMODEL) command and check the processor feature by using the DSPSYSVAL QPRCFEAT command. To check the memory, use the Work with System Status (WRKSYSSTS) command, but be sure to add the memory pools.

These requirements are simple, although they give no indication about how older models may perform. In addition, they do not provide any information about how the level of usage affects the requirements.

iSeries performance is usually measured by an IBM benchmark called *Commercial Processing Workload* (CPW). It is interesting to look at the CPW of the systems mentioned earlier in this section:

- ► Model 170 feature 2385 Processor CPW 460 Interactive CPW 50
- ► Model 720 feature 2062 Processor CPW 420 Interactive CPW 35

- ► Model 270 feature 2252 Processor CPW 950 Interactive CPW 0
- ► Model 820 feature 2396 Processor CPW 950 Interactive CPW 35

Notice the considerable variation in the processor CPW and the low interactive CPW. The variation in processor CPW may be because, although CPW is a good indicator of RPG and Cobol performance, it is not a good indicator of Java and WebSphere performance. Java performance is significantly affected by processor features, such as L2 cache, which do not affect the CPW rating as much.

There is a low interactive CPW is because WebSphere does not need interactive CPW. Little dumb terminal work is required to manage a WebSphere application. The small amount that is needed, such as starting the QEJBSBS subsystem, may be performed from the console. Because of the single task exemption, this is possible even on a system with an interactive CPW of 0. Of course, a call.connect or vendor.connect user may be running System21 on the same system. This requires a certain amount of interactive CPW, but the WebSphere applications do not add to this requirement.

In V5R1, IBM introduced a new benchmark called *Compute Intensive Workload* (CIW). This is calculated differently than CPW. It depends more on such factors as the processor speed and depends less on system services. It is hoped that this value will be a better indicator of the likely performance of Java- and WebSphere-based applications. Unfortunately, IBM has only published CIW figures for the new V5R1 models, which does not include any of the previously mentioned systems. In the future, this new benchmark should be useful. If you are considering an upgrade, then try to find the CIW of your current system and the ones to which you are considering upgrading.

For systems below these recommended minimum specifications, IBM states that you may use them in environments that support a limited number of users and where longer server initialization times can be tolerated.

Geac's experience is that a small installation may be successful if you are beneath the recommended processor levels. However, Geac recommends that you do not go below the recommended memory requirements. When there is insufficient memory, WebSphere performance is severely impaired. If you are considering an upgrade to improve WebSphere performance, then monitor your memory usage carefully. It may be more effective to upgrade the memory rather than the processor.

Older AS/400 models, before the 170 and 720 ranges, are liable to give poor WebSphere performance. Consider installing WebSphere on older AS/400 systems only if they are particularly large. Even so, the performance may be disappointing. Geac ran call.connect on a Model S30 with a CPW of 999, yet the performance was not much better than a 170-2385 with a CPW of only 460.

5.2 Operating System/400 (OS/400)

Several aspects of OS/400 configuration are important to WebSphere performance. These aspects are explained in the following sections.

5.2.1 SQL server job configuration

WebSphere applications make substantial use of Java Database Connectivity (JDBC). On the iSeries, there are two commonly used JDBC drivers:

- ► **Toolbox driver**: This is primarily used by Java applications that are not running on the iSeries itself.

► **Native driver**: This driver may only be used by a Java application that is running on the iSeries.

In most cases, if you are running WebSphere on the iSeries, the native driver offers better performance. If you used the standard configuration, as described in Chapter 3, "Installing and setting up call.connect" on page 19, then you will use the native driver.

Connections made by the native driver are serviced by jobs of the name QSQSRVR in the QSYSWRK subsystem. These are *prestart jobs*. Prestart jobs are used by IBM to speed connection times for many server jobs. Depending on the subsystem configuration, several server jobs may start together with the subsystem. The jobs then wait for a connection request. When a request is received, the jobs process the request. When a disconnect request is received, the jobs usually return to the wait state and wait for the next connection.

The default configuration of the QSQSRVR job is unlikely to give good performance for WebSphere applications. To check the configuration, follow these steps:

1. Sign on as QSECOFR.

2. Enter the following command:

   DSPSBSD QSYSWRK

3. Figure 5-2 shows the Display Subsystem Description menu. Select option 10.

```
                    Display Subsystem Description
                                                    System:    HOMER
Subsystem description:    QSYSWRK          Library:    QSYS
Status:    ACTIVE

Select one of the following:

     1. Operational attributes
     2. Pool definitions
     3. Autostart job entries
     4. Work station name entries
     5. Work station type entries
     6. Job queue entries
     7. Routing entries
     8. Communications entries
     9. Remote location name entries
    10. Prestart job entries

                                                          More...

Selection or command
===> █
_____

F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel
```

Figure 5-2   Display Subsystem Description menu

4. The Display Prestart Job Entries display (Figure 5-3) appears. Tab down to the program name QSQSRVR, type 5 next to it, and press Enter.

```
                        Display Prestart Job Entries
                                                    System:     HOMER
Subsystem description:    QSYSWRK         Status:    ACTIVE

Type options, press Enter.
  5=Display details

Opt       Program        Library        User Profile
          QANEAGNT       QSYS           QUSER
  _       QIWVPPJT       QIWS           QUSER
  _       QRWTSRVR       QSYS           QUSER
  5       QSQSRVR        QSYS           QUSER
  _       QSRRATBL       QSYS           QUSER
  _       QTMMSRVR       QTCP           QTCP
  _       QTMSCLCP       QTCP           QTCP
  _       QTMSSRCP       QTCP           QTCP
  _       Q5BWHSRV       QSYS           QUSER




                                                               Bottom
 F3=Exit    F9=Display all detailed descriptions    F12=Cancel
```

Figure 5-3   Display Prestart Job Entries list

Figure 5-4 shows the Display Prestart Job Entry Detail for the program QSQSRVR, highlighting the number of jobs. Note the following parameters for this example display:

► **Start jobs parameter**: The default is *NO. This means that the jobs do not start with the subsystem. They only start when the first connection request is received. This causes WebSphere to start slowly. This setting may be suitable if you rarely use WebSphere or any other application that uses the native JDBC driver. Otherwise, we do not recommend this setting.

► **Initial number of jobs parameter**: The default is 3. If the previous parameter is set to *NO, then this is the number of jobs that start when the first request is received. If the previous parameter is set to *YES, then this is the number of jobs that start when the subsystem is started. The default 3 is insufficient since even the WebSphere administration job QEJBADMIN uses five native JDBC connections.

► **Threshold parameter**: When the available jobs (those waiting) drop to this level, more jobs will be started.

► **Additional number of jobs parameter**: This is the number of additional jobs that are started when the threshold is reached.

► **Maximum number of jobs parameter**: IBM recommends that you set this parameter to *NOMAX. You may notice that the pool identifier on your system is 1, rather than 2 as in Figure 5-4. This is because the modification described in 5.2.3, "Subsystems and memory pools" on page 115, was already applied to this system.

```
                      Display Prestart Job Entry Detail
                                                    System:     HOMER
Subsystem description:    QSYSWRK         Status:    ACTIVE

Program  . . . . . . . . . . . . . . . . . . . . . :   QSQSRVR
  Library  . . . . . . . . . . . . . . . . . . . . :     QSYS
User profile . . . . . . . . . . . . . . . . . . . :   QUSER
Job  . . . . . . . . . . . . . . . . . . . . . . . :   QSQSRVR
Job description  . . . . . . . . . . . . . . . . . :   *USRPRF
  Library  . . . . . . . . . . . . . . . . . . . . :
Start jobs . . . . . . . . . . . . . . . . . . . . :   *YES
Initial number of jobs . . . . . . . . . . . . . . :   100
Threshold  . . . . . . . . . . . . . . . . . . . . :   20
Additional number of jobs  . . . . . . . . . . . . :   10
Maximum number of jobs . . . . . . . . . . . . . . :   *NOMAX
Maximum number of uses . . . . . . . . . . . . . . :   200
Wait for job . . . . . . . . . . . . . . . . . . . :   *YES
Pool identifier  . . . . . . . . . . . . . . . . . :   2

                                                               More...
 Press Enter to continue.

 F3=Exit    F12=Cancel    F14=Display previous entry
```

Figure 5-4   Display Prestart Job Entry Detail for QSQSRVR

You must coordinate the tuning of these server jobs with the tuning of the WebSphere datasources. For the moment, we assume a simple default WebSphere setup:

- ► Only one instance of WebSphere is running.
- ► Only one application (for example, call.connect) is running.
- ► The datasource used by that application has not been modified.
- ► There is only a small number of users.

As mentioned earlier, the WebSphere administration job uses five connections. The default datasource has a minimum size of 1 and a maximum size of 10.

One call.connect client commonly use two connections and occasionally use three connections. This way even a few users may reach this maximum.

One vendor.connect Web session uses one or occasionally two connections. However, the pooling performed by WebSphere means that unless the clients are extremely busy, the number of connections will be significantly less than the number of clients. If the maximum is likely to be reached, we recommend that your number of prestarted jobs is at least as large. For this discussion, we assume that there are 15 connections. The Display Prestart Job Entry Detail for this program would be:

- ► Start jobs: *YES
- ► Initial number of jobs: 20
- ► Threshold: 2
- ► Addition number of jobs: 2
- ► Maximum number of jobs: *NOMAX

If other Java applications are running on the iSeries that use the native JDBC driver, you must allow for them as well. To make this change, follow these steps:

1. Sign on as QSECOFR.

2. Ensure that WebSphere is not running.

3. Ensure that no other Java applications are using native JDBC connections.

4. End the prestart jobs by using the following command:

   ```
   ENDPJ SBS(QSYSWRK) PGM(QSQSRVR)
   ```

   If no server jobs are running, then you receive the error message "End Prestart Jobs command is not currently allowed". You can ignore this message.

5. Change the server settings with this command:

   ```
   CHGPJE SBSD(QSYSWRK) PGM(QSQSRVR) STRJOBS(*YES) INLJOBS(20) THRESHOLD(2) ADLJOBS(2)
   MAXJOBS(*NOMAX)
   ```

6. Restart the prestart jobs:

   ```
   STRPJ SBS(QSYSWRK) PGM(QSQSRVR)
   ```

7. Restart WebSphere and your application.

It is possible to view the server jobs in several ways. Look at the QSYSWRK subsystem:

```
WRKACTJOB SBS(QSYSWRK)
```

The results also show many other server jobs. Instead you can select a specific job name by using the following command:

```
WRKACTJOB JOB(QSQSRVR)
```

In either case, you may find fewer jobs than you expect. If you check when WebSphere is not active, then you may find none. This is because the Work with Active Jobs (WRKACTJOB)

command, by default, does not show prestart jobs that are waiting for a connection. However, if you press F14, they are shown.

Figure 5-5 shows the QSQSRVR jobs in the Work with Active Jobs display. Look at the Status column. PSRW indicates that the job is waiting for a connection. These are the jobs that were not shown until you pressed F14. If you press F14 again, you hide these jobs. In this example, F14 was pressed so jobs with status PSRW are shown as well as those with other status indicators such as CNDW.

Using F14 is a convenient way to check whether any of the jobs are in use. If the jobs appear in one view but not in the other, then they are started but not in use. If they do not appear in either view, it is most likely that the Start jobs parameter is still set to *NO and the jobs have not been used. Be sure to carefully check the spelling of the job name. If it is incorrect, you do not receive any error message.

```
                           Work with Active Jobs                      HOMER
                                                     20/12/01   12:10:17
CPU %:     1.2      Elapsed time:    00:33:56     Active jobs:    323

Type options, press Enter.
  2=Change   3=Hold   4=End    5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...

Opt   Subsystem/Job   User        Type   CPU %   Function       Status
■_      QSQSRVR       QUSER        PJ      .0                    PSRW
__      QSQSRVR       QUSER        PJ      .0                    PSRW
__      QSQSRVR       QUSER        PJ      .0                    PSRW
__      QSQSRVR       QUSER        PJ      .0                    PSRW
__      QSQSRVR       QUSER        PJ      .0                    PSRW
__      QSQSRVR       QUSER        PJ      .0                    CNDW
__      QSQSRVR       QUSER        PJ      .0                    PSRW
__      QSQSRVR       QUSER        PJ      .0                    CNDW
__      QSQSRVR       QUSER        PJ      .0                    PSRW
                                                                   More...
Parameters or command
===>
F3=Exit   F5=Refresh       F7=Find      F10=Restart statistics
F11=Display elapsed data   F12=Cancel   F23=More options   F24=More keys
```

Figure 5-5   The WRKACTJOB command showing the QSQSRVR jobs

5.2.2 Toolbox JDBC driver

There are two versions of the Toolbox driver:

► The licensed program version
► The Open Source edition

Both version use the same server jobs. Therefore, you choice in version does not affect the iSeries configuration.

The most likely reason for using the Toolbox driver is that you are running WebSphere and System21 on different systems. If WebSphere *is not* on another iSeries server, then you use the Toolbox driver. If WebSphere *is* on another iSeries server, then you use either driver. However the Toolbox driver may be easier to use and perform better.

The configuration is slightly different depending on whether you are using V5R1 or V4R5. We recommend that you do not use V4R4. However, if you were using it, then it would be the same as V4R5.

Note that the number of potential users of the Toolbox driver is greater. The clients can be on other systems, even ones that are not iSeries servers. The servers are shared by ODBC so non-Java applications can be clients to these server jobs.

It is necessary to check carefully that the server jobs are not in use. You can check whether the jobs are in use in a way that is similar to checking the QSQSRVR jobs. For example, you use this command:

```
WRKACTJOB JOB(QZDASOINIT)
```

If some jobs appear immediately, then some clients are connected. Look at the job log (by selecting option 5 and then option 10), and you should see the user name and the IP address of the client. If no jobs appear, then press F14, and some in the status PSRW should appear. These are idle jobs, which you may safely end. If no jobs appear in either view, then it may be that none are started. Since this is unlikely, be sure to verify the spelling of the job name.

For V5R1, use the procedure in 5.2.1, "SQL server job configuration" on page 109, but substitute the following commands:

Note: The commands are the same, but the parameters may be different.

```
DSPSBSD QUSRWRK
ENDPJ SBS(QUSRWRK) PGM(QZDASOINIT)
CHGPJE SBSD(QUSRWRK) PGM(QZDASOINIT) STRJOBS(*YES) INLJOBS(20) THRESHOLD(2) ADLJOBS(2)
STRPJ SBS(QUSRWRK) PGM(QZDASOINIT)
WRKACTJOB SBS(QUSRWRK)
WRKACTJOB JOB(QZDASOINIT)
```

Figure 5-6 shows the Display Prestart Job Entry Detail for the program QZDASOINIT, running on a V5R1 system. Notice the numbers that are highlighted for the job.


```
                       Display Prestart Job Entry Detail
                                                        System:      HOMER
Subsystem description:    QUSRWRK          Status:      ACTIVE

Program  . . . . . . . . . . . . . . . . . . . :    QZDASOINIT
  Library  . . . . . . . . . . . . . . . . . :      QSYS
User profile . . . . . . . . . . . . . . . . :    QUSER
Job  . . . . . . . . . . . . . . . . . . . . :    QZDASOINIT
Job description  . . . . . . . . . . . . . . :    *USRPRF
  Library  . . . . . . . . . . . . . . . . . :
Start jobs . . . . . . . . . . . . . . . . . :    *YES
Initial number of jobs . . . . . . . . . . . :    50
Threshold  . . . . . . . . . . . . . . . . . :    10
Additional number of jobs  . . . . . . . . . :    10
Maximum number of jobs . . . . . . . . . . . :    *NOMAX
Maximum number of uses . . . . . . . . . . . :    200
Wait for job . . . . . . . . . . . . . . . . :    *YES
Pool identifier  . . . . . . . . . . . . . . :    1

                                                                More...

Press Enter to continue.

F3=Exit    F12=Cancel    F14=Display previous entry
```

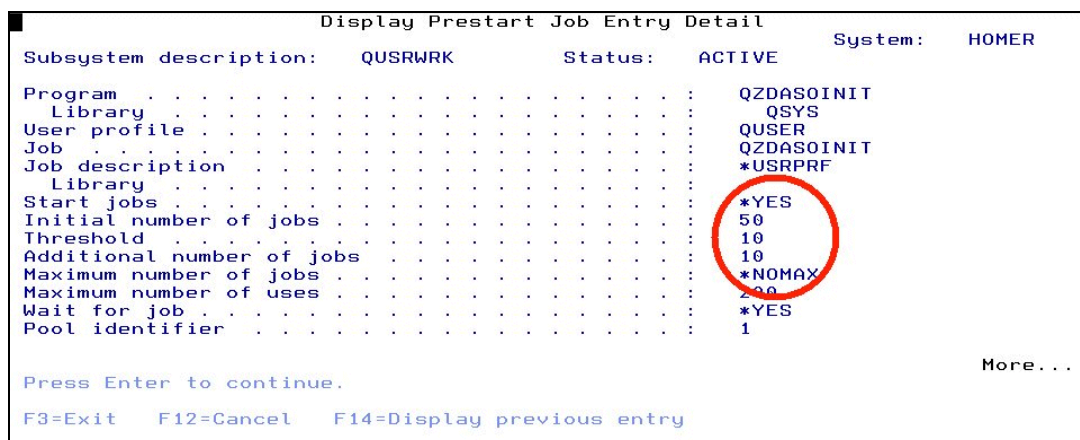*Figure 5-6   Display Prestart Job Entry Detail for QZDASOINIT*

V4R5 uses slightly different commands as listed here:

Note: The library name is required in the PGM parameters.

```
DSPSBSD QSERVER
ENDPJ SBS(QSERVER) PGM(QIWS/QZDASOINIT)
CHGPJE SBSD(QSERVER) PGM(QIWS/QZDASOINIT) STRJOBS(*YES) INLJOBS(20) THRESHOLD(2) ADLJOBS(2)
STRPJ SBS(QSERVER) PGM(QIWS/QZDASOINIT)
WRKACTJOB SBS(QSERVER)
WRKACTJOB JOB(QZDASOINIT)
```

5.2.3 Subsystems and memory pools

By default, WebSphere jobs run in the QEJBSBS subsystem in the base memory pool. In addition, WebSphere applications, such as call.connect and vendor.connect, substantially use the native JDBC server jobs that run in QSYSWRK. This is also in the base memory pool, by default.

If your system runs no other significant applications, then this simple setup may be sufficient. The resource usage of WebSphere applications is quite different from traditional interactive and batch jobs. Therefore, if you also run these types of jobs, it may be preferable to use a separate memory pool. A separate memory pool allows you control the resources allocated to WebSphere applications. You can enhance WebSphere performance, possibly at the expense of others jobs, by increasing its memory pool. Or you can reduce the impact on other applications by reducing it.

You can set up separate pools for the QEJBSBS subsystem and the QSQSRVR jobs, but in most cases, this is unnecessarily complicated to manage. If the two are combined, then you must have a shared pool.

Ideally, you should know the configuration of your system and know which shared pools are available. You can see if a shared pool is currently in use by using the WRKSYSSTS command.

On the Work with System Status display (Figure 5-7), press F11 until you see the Pool column. In this column, you will see such names as *MACHINE, *BASE, *INTERACT, and maybe *SHRPOOL1, *SHRPOOL2, etc. You can also use the Work with Shared Pool (WRKSHRPOOL) command. If the defined size and maximum active values are both zero, then the pool may never have been used.

This example assumes shared pool 1, but any other can be substituted. In Figure 5-7, a shared pool is already setup for WebSphere. Shared pool 2 was used.

Figure 5-7   WRKSYSSTS command showing the memory pool names

You need to specify the memory size and activity level. If possible, specify at least 500 MB and better still, specify 1 GB. If your client will be busy, then set the activity level to approximately the number of users. You can adjust both figures once the system is configured and used.

To set up a shared pool for WebSphere and the SQL server jobs, follow these steps:

1. End any WebSphere applications, WebSphere itself, and any other Java applications that are using the native JDBC driver.

2. End the QEJBSBS subsystem by using the following command:

```
ENDSBS QEJBSBS
```

3. View the shared pools with the WRKSHRPOOL command.

4. On the Work with Shared Pools display (Figure 5-8), choose an available pool, for example, *SHRPOOL1, and enter your chosen size and maximum activity level.

```
                        Work with Shared Pools
                                                    System:    HOMER
Main storage size (M)  . :          2048.00

Type changes (if allowed), press Enter.

                Defined      Max     Allocated    Pool  -Paging Option--
Pool            Size (M)   Active    Size (M)      ID   Defined  Current
*MACHINE         174.89    +++++       174.89       1   *FIXED   *FIXED
*BASE            323.10      176       323.10       2   *CALC    *CALC
*INTERACT         50.00       35        50.00       5   *CALC    *CALC
*SPOOL             .25         1                        *FIXED
*SHRPOOL1                                               *FIXED
*SHRPOOL2       1500.00      120      1500.00       3   *CALC    *CALC
*SHRPOOL3                                               *FIXED
*SHRPOOL4           .00        0                        *FIXED
*SHRPOOL5           .00        0                        *FIXED
*SHRPOOL6           .00        0                        *FIXED
                                                                  More...
Command
===>
F3=Exit    F4=Prompt    F5=Refresh    F9=Retrieve   F11=Display tuning data
F12=Cancel
```

Figure 5-8   WRKSHRPOOL command showing the pool defined for WebSphere

In this example, shared pool 2 is setup for use by WebSphere. In fact, this was done before the previous WRKSYSSTS command was run. You can see that shared pool 1 is also set, but no subsystems or jobs are using it since it did not appear on the WRKSYSSTS display.

Alternatively, you can do this by using the CHGSHRPOOL command:

```
CHGSHRPOOL POOL(*SHRPOOL1) SIZE(500000) ACTLVL(20)
```

Note that the sizes shown by WRKSHRPOOL are in Mb, but the sizes used by CHGSHRPOOL are in Kb. In addition, the CHRSHRPOOL command does not warn you that the pool is already in use.

5. Add the storage pool to the QEJBSBS subsystem.

6. Check whether the QEJBSBS subsystem was already modified by using the following command:

```
DSPSBSD SBSD(QEJB/QEJBSBS)
```

7. On the next display, select option 2 to see the pools.

8. On the Display Pool Definitions for QEJBSBS display (Figure 5-9), if there is a single entry with a pool ID of 1 and a size of *BASE, then the subsystem was not yet modified in this way. If it was modified, then determine why and how it was modified before you continue to the next step.

```
                       Display Pool Definitions
                                               System:     HOMER
Subsystem description:     QEJBSBS         Status:     ACTIVE

Pool        Storage     Activity
 ID         Size (K)     Level
  1          *BASE
  2        *SHRPOOL2




                                                        Bottom
Press Enter to continue.

F3=Exit    F12=Cancel
```

Figure 5-9   The Display Pool Definitions display for QEJBSBS

In this example, the modification was already made using shared pool 2, so subsystem pool 2 appears. Note that it is a coincidence that pool 2 of this subsystem happens to be shared pool 2. If you enter the following command, the subsystem pool 2 will be shared pool 1:

```
CHGSBSD SBSD(QEJB/QEJBSBS) POOLS((1 *BASE) (2 *SHRPOOL1))
```

For a more simplified method, you can enter:

```
CHGSBSD SBSD(QEJB/QEJBSBS) POOLS((1 *SHRPOOL1))
```

If you use this simpler method, you do not need to perform step 11. The advantage of the previous way is that the subsystem monitor job still runs in the base pool and can be segregated from the other jobs of the subsystem.

9. You may want to check the subsystem again using the following command:

```
DSPSBSD SBSD(QEJB/QEJBSBS)
```

10. On the next display, select option 2. The result should be similar to the example in Figure 5-9, except that the shared pool will be *SHRPOOL1.

11. Change the routing entries so that the new pool is used. Enter the following command:

```
CHGRTGE SBSD(QEJB/QEJBSBS) SEQNBR(9999) POOLID(2)
```

You should omit this step if you used the simplified method from step 8. However, if you used the original command, then it is essential that you perform this step. Otherwise, the allocated memory will be wasted. You may want to check this step.

a. Enter the following command:

```
DSPSBSD SBSD(QEJB/QEJBSBS)
```

b. On the next display, select option 7 for routing entries.

c. You should see a single entry with sequence number 9999. Type 5 next to this entry and press Enter.

The result should be similar to the Display Routing Entry Detail display example in Figure 5-10.

```
                       Display Routing Entry Detail
                                                        System:      HOMER
Subsystem description:      QEJBSBS            Status:      ACTIVE

Routing entry sequence number . . . . . . . . :        9999
Program . . . . . . . . . . . . . . . . . . . :        QCMD
  Library . . . . . . . . . . . . . . . . . . :          QSYS
Class . . . . . . . . . . . . . . . . . . . . :        QEJBCLS
  Library . . . . . . . . . . . . . . . . . . :          QEJB
Maximum active routing steps  . . . . . . . . :        *NOMAX
Pool identifier . . . . . . . . . . . . . . . :        2
Compare value . . . . . . . . . . . . . . . . :        *ANY

Compare start position  . . . . . . . . . . . :




Press Enter to continue.

F3=Exit    F12=Cancel     F14=Display previous entry
```

Figure 5-10   Display Routing Entry Detail display

12. End the SQL server prestart jobs by entering the following command:

    ENDPJ SBS(QSYSWRK) PGM(QSQSRVR)

13. Add the storage pool to the QSYSWRK subsystem.

14. Verify whether the QSYSWRK subsystem was already modified by entering the following command:

    DSPSBSD SBSD(QSYSWRK)

15. On the next display, select option 2 to see the pool.

16. On the Display Pool Definitions display (Figure 5-11), if there is a single entry with a pool ID of 1 and a size of *BASE, then the subsystem was not yet modified in this way. If it was modified, then determine why and how it was modified before you continue to the next step.

    If you want to continue with this modification, then change the pool ID (2 in this example) to a number higher than the highest number that is currently in use. Note that this number is the pool number relative to this subsystem and the system as a whole as shown by WRKSYSSTS.

```
                       Display Pool Definitions
                                                        System:      HOMER
Subsystem description:      QSYSWRK            Status:      ACTIVE

Pool          Storage       Activity
 ID           Size (K)       Level
   1           *BASE
   2          *SHRPOOL2




                                                              Bottom
Press Enter to continue.

F3=Exit    F12=Cancel
```

Figure 5-11   Display Pool Definitions display for QSYSWRK

Again, the modification was already made using shared pool 2. Therefore, subsystem pool 2 appears. In this case, the subsystem pool number is the same as for QEJBSBS, but this is not necessary. When started a system pool number, as shown by WRKSYSSTS, will be

assigned but this could vary. However because both this entry and the QEJBSBS entry above specify *SHRPOOL2, the two sets of jobs will share the same memory pool.

17. Change the SQL server prestart job to use the new pool:

`CHGPJE SBSD(QSYSWRK) PGM(QSQSRVR) POOLID(2)`

You may want to verify this step:

a. Use the following command:

`DSPSBSD SBSD(QSYSWRK)`

b. On the next display, select option 10 for prestart job entries.

c. On the next display, type 5 next the QSQSRVR entry and press Enter.

The result should be similar to the Display Prestart Job Entry Detail display (Figure 5-12) for the QSQSRVR program, with the pool identifier highlighted.
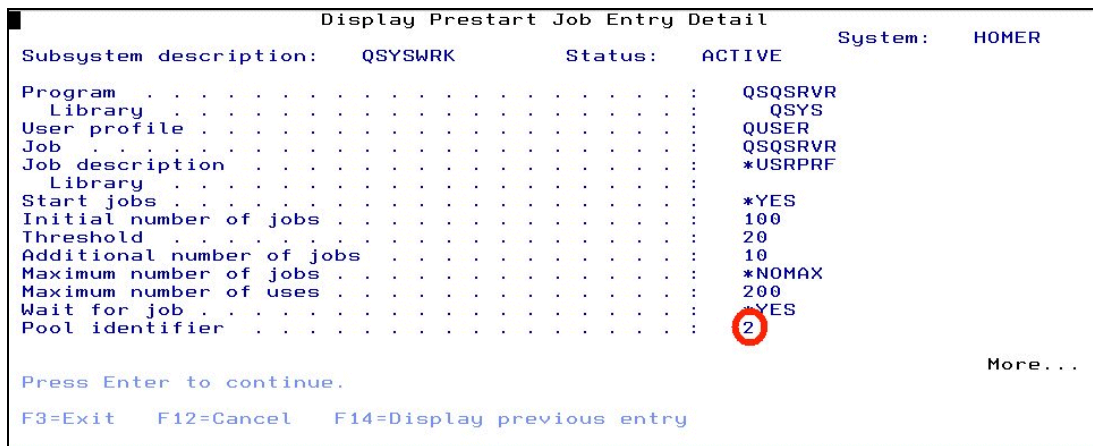
```
                      Display Prestart Job Entry Detail
                                                        System:      HOMER
Subsystem description:    QSYSWRK            Status:    ACTIVE

Program  . . . . . . . . . . . . . . . . . . . . :    QSQSRVR
  Library  . . . . . . . . . . . . . . . . . . . :      QSYS
User profile . . . . . . . . . . . . . . . . . . :    QUSER
Job  . . . . . . . . . . . . . . . . . . . . . . :    QSQSRVR
Job description  . . . . . . . . . . . . . . . . :    *USRPRF
  Library  . . . . . . . . . . . . . . . . . . . :
Start jobs . . . . . . . . . . . . . . . . . . . :    *YES
Initial number of jobs . . . . . . . . . . . . . :    100
Threshold  . . . . . . . . . . . . . . . . . . . :    20
Additional number of jobs  . . . . . . . . . . . :    10
Maximum number of jobs . . . . . . . . . . . . . :    *NOMAX
Maximum number of uses . . . . . . . . . . . . . :    200
Wait for job . . . . . . . . . . . . . . . . . . :    *YES
Pool identifier  . . . . . . . . . . . . . . . . :    2

                                                                  More...

Press Enter to continue.

F3=Exit   F12=Cancel   F14=Display previous entry
```

Figure 5-12   Display Prestart Job Entry Detail for QSQSRVR

If you did not select option 2 in step 7 on page 116, then you should use the same number again here.

18. Restart the prestart jobs by entering the following command:

`STRPJ SBS(QSYSWRK) PGM(QSQSRVR)`

19. Check whether the memory pool started.

20. Enter the `WRKSYSSTS` command.

21. You see that another line has appeared with your specified size and activity level. It may be in the last line since it is newly created. But it may not always be there. Its position after an IPL depends on the sequence that your subsystems are started.

Press F11 until you see the pool name. You should see *SHRPOOL1 or whichever one you used. Note the system pool number of your new pool (this may change after an IPL). If the size or activity level is not the one that you specified, then it may be the affect of the performance adjuster. Figure 5-13 shows the Work with System Status display with the pool names.

```
                    Work with System Status                    HOMER
                                                    20/12/01  12:20:05
% CPU used . . . . . . . . :          1.2    System ASP . . . . . . . :      70.19 G
% DB capability  . . . . :            .0    % system ASP used  . . . :     57.8447
Elapsed time . . . . . . :      01:22:05    Total aux stg  . . . . . :      70.19 G
Jobs in system . . . . . :           795    Current unprotect used . :      2727 M
% perm addresses . . . . :          .013    Maximum unprotect  . . . :      5146 M
% temp addresses . . . . :          .041

Sys      Pool    Reserved      Max                                        Paging
Pool     Size M    Size M      Act  Pool        Subsystem   Library       Option
  1    ■ 174.89    104.93    +++++  *MACHINE                              *FIXED
  2      323.10      1.00      176  *BASE                                 *CALC
  3     1500.00       .53      120  *SHRPOOL2                             *CALC
  5       50.00       .01       35  *INTERACT                            *CALC




                                                                        Bottom
===>  _____
F21=Select assistance level
```

Figure 5-13   WRKSYSSTS display showing the memory pool names

As in the other images, shared pool 2 was used. It is not appearing as the last system pool. That's because the system was restarted since the configuration change was made, and during the following start, QSYSWRK started before QINTER.

22. Check the QSQSRVR jobs by using the following command:

   WRKACTJOB JOB(QSQSRVR)

23. On the Work with Active Jobs display (Figure 5-14), you may need to press F14 to see the jobs, since at this stage, they may all be waiting. Press F11 to see the Pool column. The QSQSRVR jobs should be using the new pool. Note that the number will not be 2 as specified in step 9 through step 11 on page 117, but the system pool number as noted in step 21. If the new pool is not being used, check that you followed step 9 through step 11 on page 117 correctly.

   In this example, the QSQSRVR jobs use pool 3, which matches the system pool for *SHRPOOL2 in Figure 5-13. Therefore, the QSQSRVR jobs use shared pool 2.

```
                    Work with Active Jobs                    HOMER
                                                  20/12/01  15:59:25
CPU %:     1.2      Elapsed time:    04:23:04     Active jobs:     324

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...
                                     --------Elapsed---------
Opt   Subsystem/Job   Type  Pool  Pty    CPU    Int    Rsp   AuxIO   CPU %
 ■      QSQSRVR        PJ    3     25    80.1                    0     .0
 __     QSQSRVR        PJ    3     25   137.5                    0     .0
 __     QSQSRVR        PJ    3     20   628.9                    0     .0
 __     QSQSRVR        PJ    3     20    97.1                    0     .0
 __     QSQSRVR        PJ    3     10     .1                     0     .0
 __     QSQSRVR        PJ    3     10     .1                     0     .0
 __     QSQSRVR        PJ    3     10     .1                     0     .0
 __     QSQSRVR        PJ    3     10     .1                     0     .0
 __     QSQSRVR        PJ    3     10     .1                     0     .0
                                                                   More...
Parameters or command
===> _____
F3=Exit    F5=Refresh      F7=Find      F10=Restart statistics
F11=Display thread data    F12=Cancel   F23=More options    F24=More keys
Already at top of area.
```

Figure 5-14   WRKACTJOB display showing the pool numbers for the QSQSRVR jobs

24. Start the WebSphere subsystem:

   STRSBS SBSD(QEJB/QEJBSBS)

   Notice that this starts the default instance of WebSphere.

25. Check the QEJBSBS subsystem:

   WRKACTJOB SBS(QEJBSBS)

26. On the Work with Active Jobs display (Figure 5-15), press F11 to see the Pool column. All the jobs, except the subsystem monitor QEJBSBS, should be running in your new pool (the number noted in step 21). If the new pool is not being used, then verify that you followed step 9 through step 11 on page 117 correctly.

```
                          Work with Active Jobs                    HOMER
                                                      20/12/01   15:20:35
CPU %:     1.2       Elapsed time:   03:44:14      Active jobs:    321

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...
                                             --------Elapsed--------
Opt  Subsystem/Job   Type  Pool  Pty      CPU   Int    Rsp   AuxIO   CPU %
 ■_   QEJBSBS         SBS    2    0       .0                     0      .0
 __    SALESTEST      BCI    3    25+   9376.3                   0      .6
 __    TESTADMIN      BCI    3    25+   4213.1                   0      .2
 __    TESTMNTR       BCH    3    25       .0                    0      .0




                                                                   Bottom
Parameters or command
===>
F3=Exit    F5=Refresh       F7=Find       F10=Restart statistics
F11=Display thread data   F12=Cancel    F23=More options   F24=More keys
```

Figure 5-15   WRKACTJOB command showing the pool numbers for the QEJBSBS jobs

As in Figure 5-13, the jobs are using pool 3. This matches the *SHRPOOL2 entry in the earlier WRKSYSSTS display. Note that the subsystem job itself is still using system pool 2, which is *BASE. This is because the simplification mentioned in step 8 on page 116 was not taken. Note also that the default instance of WebSphere is not running. However, an alternative instance TEST is running, and so is the application SALESTEST within it.

5.2.4 Automatic performance adjustment

You should verify whether automatic performance adjustment is active on your system. Check the system value QPFRADJ by using the following command:

`DSPSYSVAL SYSVAL(QPFRADJ)`

Figure 5-16 shows the Display System Value display for QPFRADJ.

```
                          Display System Value

System value . . . . . . :    QPFRADJ
Description  . . . . . . :    Performance adjustment


Performance adjustment . . . :    0          0=No adjustment
                                             1=Adjustment at IPL
                                             2=Adjustment at IPL and automatic
                                                adjustment
                                             3=Automatic adjustment








Press Enter to continue.

F3=Exit   F12=Cancel
```

Figure 5-16   DSPSYSVAL command showing the QPFRADJ system value

If the setting is not 0, then some form of automatic performance adjustment is active.

If WebSphere is the only significant application on your system, then the performance adjuster may do a reasonable job. In this case, there may not be any value to a dedicated memory pool for WebSphere.

If WebSphere will run alongside more traditional interactive and batch applications, such as System21, then the performance adjuster does not seem to do a good job. It appears to favor interactive jobs and reduces the memory of the WebSphere subsystem so much that WebSphere performs very poorly. This even happens when interactive performance is satisfactory and does not seem to require more memory.

In an extreme case, if WebSphere is idle for an extended period, its memory may be reduced so much that it struggles to restart when someone starts to use one of the WebSphere applications.

Therefore, if you have a mixed WebSphere and traditional workload, you may need to turn the performance adjuster off and use manual tuning. To turn the performance adjuster off, sign on as `QSECOFR` and enter this command:

```
WRKSYSVAL SYSVAL(QPFRADJ)
```

Type 2 next to the QPFRADJ system value, press Enter, type 0 on the next display, and press Enter again. Of course, you may have to watch your system performance, and if necessary, make further manual adjustments.

5.2.5 Manual performance adjustment

Most aspects of manual performance tuning for WebSphere are the same as for any other application. The most useful tool for basic tuning is the WRKSYSSTS command. Watch the faulting level, the wait to ineligible, and active to ineligible rates in the new WebSphere pool.

5.3 Stateless and stateful connections, datasources, connection pools, etc.

The call.connect application uses both stateless and stateful JDBC connections.

5.3.1 Stateful connections

call.connect uses the Advanced Pricing module of System21. This module is stateful. This means that it retains data that is relevant to the client between calls. If you price an order with multiple lines, the price of a line may be affected by preceding lines. Quantity discounts may apply to a group of items. Therefore, several lines ordering different items from the same group may qualify for a discount, even though one line could not qualify by itself.

Stateful connections are not commonly used in WebSphere. call.connect uses special support. The beans are still stateless session beans, but stateful connections are obtained using a connection manager within call.connect. These stateful connections are managed using a client ID so that a client obtains the same connection on each call.

This means that every active call.connect client requires a connection. This connection is not managed by WebSphere and you do not need to consider it when sizing WebSphere connection pools. However, you need to consider it when you configure iSeries SQL server jobs.

vendor.connect does not use stateful connections.

5.3.2 Stateless connections

The majority of call.connect beans and all vendor.connect beans are common stateless session beans and use stateless connections. One method call (typically one interaction between the client and the server) obtains a stateless connection from the WebSphere datasource, performs the necessary work, and then returns the connection to the pool.

If all clients would execute a method at exactly the same time, then you would need one connection per client. A single call.connect client sometimes obtains two independent connections for reasons of transaction boundaries.

In an extreme case, the number of stateless connections required could be twice the number of clients. However in practice, this is extremely unlikely. You should attempt to estimate the likely number of simultaneous connections. Once the system is running, you can verify your estimate with the WebSphere Resource Analyzer. If your value is too low, then clients will have to wait for connections and response times may become unpredictable. If the value is too high, then you will use more iSeries resources than required. If you are in doubt, err on the high side, a small number of active but idle connections is unlikely to be a problem.

If call.connect and vendor.connect are on the same iSeries, then you can configure them to use separate datasources or a common one. The pros and cons of the choice are complex. A common source may reduce the number of connections required since it is unlikely that both applications will reach their maximum usage levels at the same time. On the other hand, sharing the connections means that all connections accumulate prepared statements from each application. Therefore, they have a larger memory requirement per connection than if they were separated. A simple setup uses a common pool. Unless you have a reason to do otherwise, you should retain this.

The size of the datasource connection pool can be controlled using the WebSphere console. Start the console and select the datasource used by your application. On the WebSphere Console's DataSource panel (Figure 5-17), select the **Advanced** tab. Note the Minimum connection pool size and the Maximum connection pool size fields.

The maximum pool size has the most significant affect on performance. If it is insufficient, then clients are forced to wait when they need a connection. In extreme cases, the application can lock up. The ideal tuning technique is to use the WebSphere Resource Analyzer to monitor actual use. A simpler approach is to make a high estimate and drop it down gradually until it impacts performance. Except for extreme values, performance is not greatly affected by this setting.

The minimum pool size is less significant but worth considering. This is the number of connections that WebSphere will retain even if they are not in use. The default of 1 is reasonable if there are extended periods in which WebSphere is not in use but other applications are. However, if there will rarely be times when WebSphere is idle, or when WebSphere and most other applications are idle, then a larger value is desirable. Larger values should improve startup times after periods of inactivity. Try to estimate the lowest level of WebSphere usage while other applications are busy. Ignore periods when WebSphere and other applications are not busy.
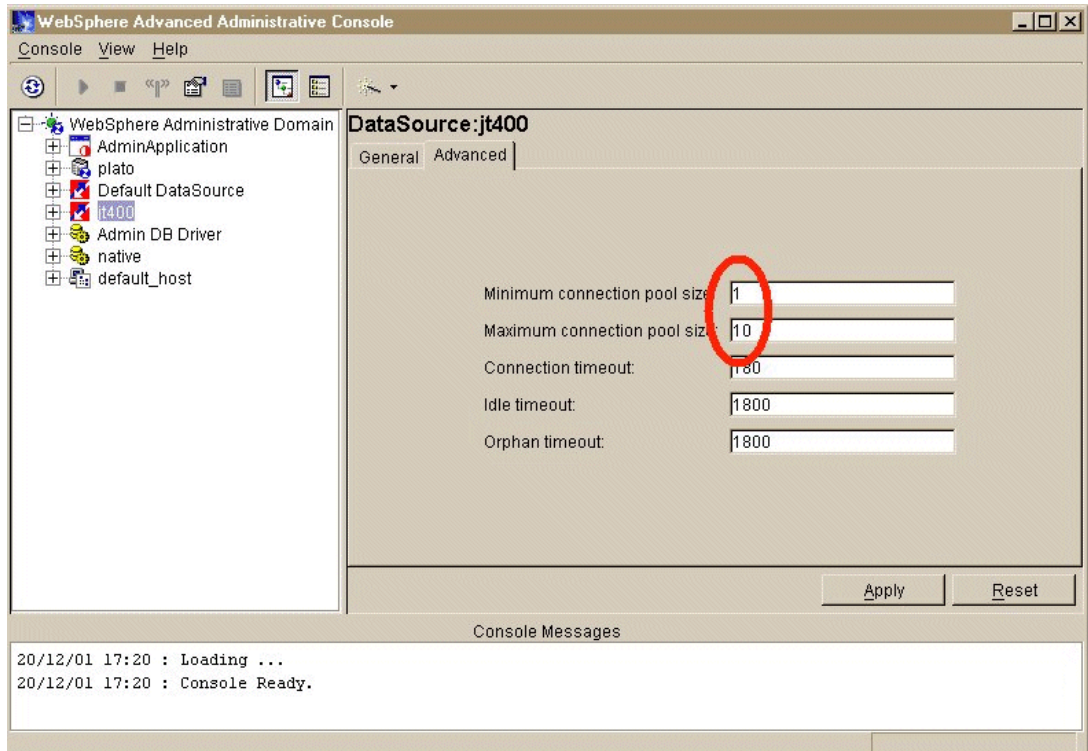
Figure 5-17   The DataSource panel Advanced page of the WebSphere console

5.3.3 Total connections and SQL server jobs

If you estimated the number of stateful and stateless connections required by your applications, then you can easily estimate the total number of connections. You should add these two and the five connections used by the WebSphere administration server job QEJBADMIN. If you are using multiple applications (for example, call.connect and vendor.connect in different pools), then you should also add them. If you are running multiple instances, then count each separately and then add them together. Each instance has its own administration server job and, therefore, five connections.

It is important that sufficient SQL server jobs are available for these connections. See 5.2.1, "SQL server job configuration" on page 109. If WebSphere attempts to use more connections than the number of prestarted SQL server jobs, then performance may be poor.

Any time you adjust the size of your WebSphere connection pools, review the configuration of your SQL server jobs. Remember that if you have other applications that use these server jobs, then you should include them in the calculation of the number of server jobs required. The normal server QSQSRVR jobs are only used by JDBC connections using the native JDBC driver. Therefore, if do not have any other Java applications on your iSeries, then no other applications are using these jobs. If you are using the Toolbox driver QZDASOINIT, remember that Java applications running elsewhere and ODBC clients may use it.

5.4 Performance topics for Java virtual machine (JVM) settings

This section discusses performance-related topics regarding to JVM settings. You can specify command line parameters for the application's JVM using the WebSphere console. Simply select the application and then select the **General** tab.

5.4.1 Initial Java heap size

–Xms is an important parameter that controls the initial Java heap size. It is followed by the size, which is usually an integer followed by m (represents megabytes). The default is 32 Mb, which would appear as –Xms32m.

Figure 5-18 highlights the Xms parameter on the WebSphere Console's Application Server panel.
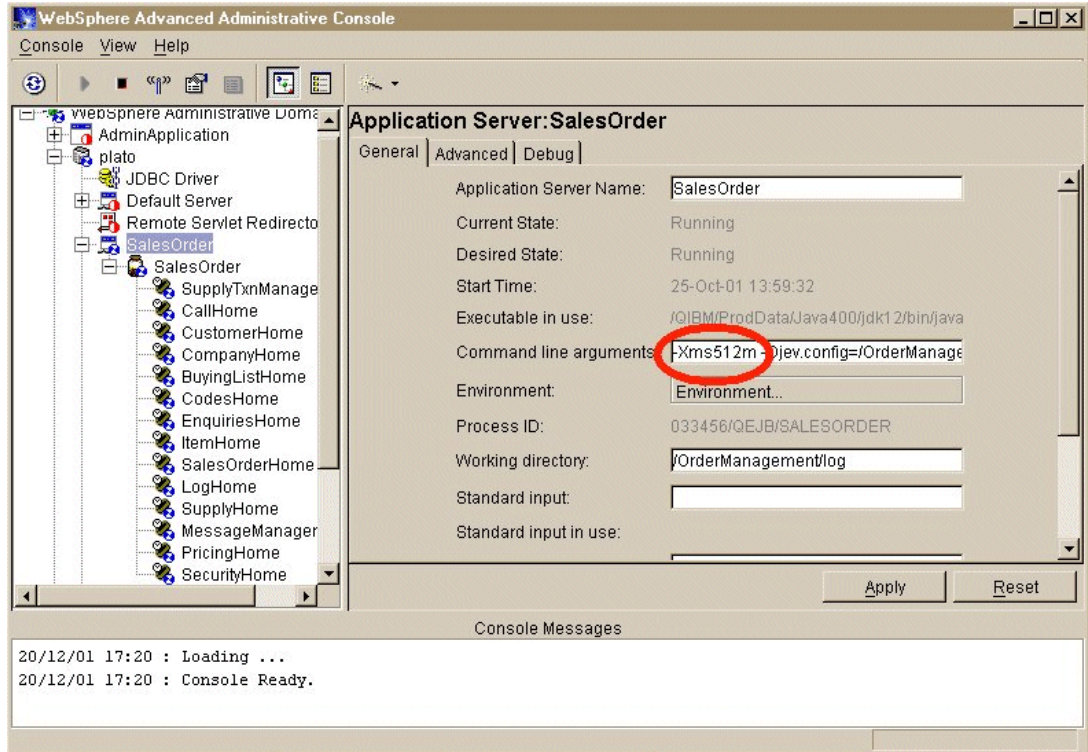

Figure 5-18   Application Server panel of the WebSphere console showing the Xms parameter

If you have only a few users who will not be very busy, then this setting may be sufficient. If, however, you have many users, especially active users, you may benefit from a larger value.

Increasing the value can affect the iSeries in two ways. First, the initial size of the Java heap will be greater. This should speed the startup of the application. Second, it causes garbage collection to run less frequently. The less frequently it runs, the better the average performance should be. If it is too infrequent, then the impact may be significant when it runs.

5.4.2 Maximum Java heap size

It is also possible to specify a maximum Java heap size. The parameter is –Xmx and is followed by a value in the same form as the –Xms parameter.

For the iSeries, we recommend that you do not set this parameter because we have not found it to be useful. If you do not specify it, then the heap grows as required including beyond physical memory.

5.4.3 Verbose garbage collection

The JVM has an option called *verbose garbage collection*. This means that the garbage collector outputs statistics on its activity. To activate the option, add the parameter `–verbosegc` to the command line arguments in the Properties window of the application server. You must keep this option separate from other options by adding a space but the sequence is not important.

The messages are sent to the standard output file, which was defined when the application was created. You can verify this setting on the General tab of the application server settings. The file name is specified in the standard output field. If it does not specify the directory, then the file is in the working directory specified on the same tab. Note that if the standard output file starts with an exclamation mark (!), then the file is overwritten at the time that the application is started. Otherwise the file is appended to.

Figure 5-19 highlights the verbosegc and related parameters on the WebSphere Console's Application Server panel.
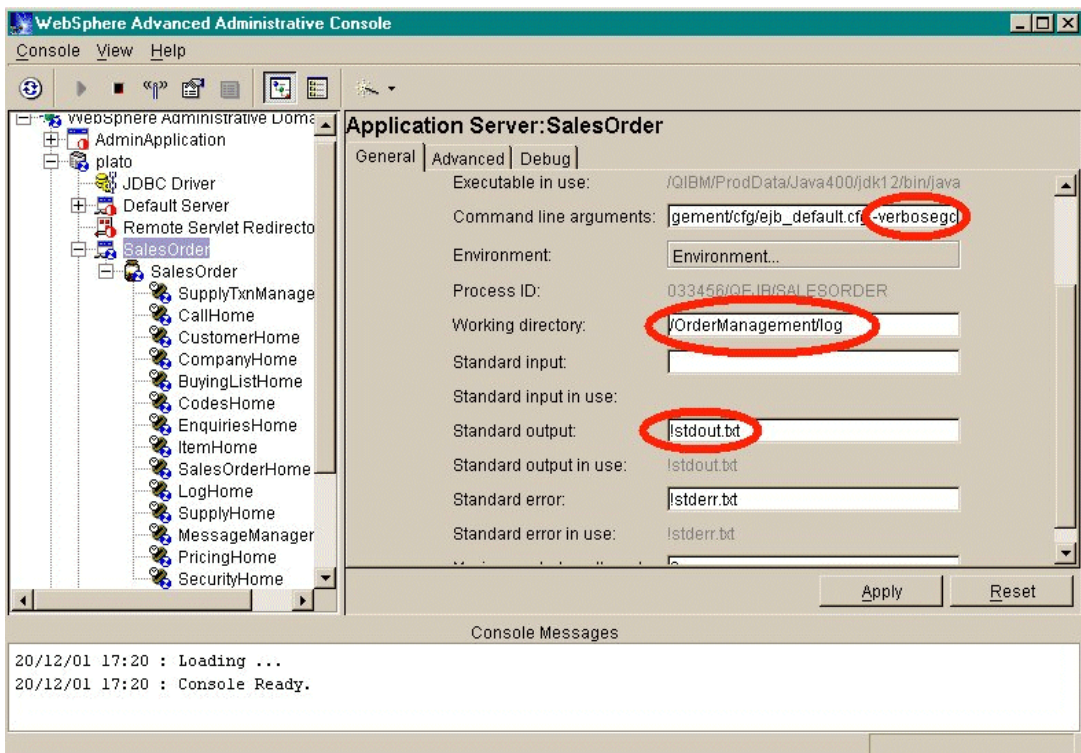
Figure 5-19   Application Server panel of the WebSphere console showing the -verbosegc parameter

In this example, the output goes to /OrderManagement/log/stdout.txt. Since the exclamation mark (!) is placed at the beginning of the file name, it is rewritten when the application is started. If you want to save the data from a previous run, then either do not use the !, or rename the file before you restart the application.

The example in Figure 5-19 shows the page scrolled down slightly so that you can see all three relevant fields. The command line arguments field is very long. In this case, the –verbosegc parameter is at the end and appears only because the cursor was placed in the field and the End key used to move to the end of the value. For a tip on editing the command line arguments, see 5.4.4, "Static compilation" on page 127.

A simple way to view this file is to map a PC drive to the iSeries integrated file system (IFS) and then use a PC editor. Another simple way to view this file is to use the OS/400 Edit File (EDTF) command. An another less friendly but useful technique is to use the Qshell `tail` command. You can learn more about each of these techniques in Chapter 6, "Tips and techniques" on page 137.

The following example shows sample output produced by verbose garbage collection.

```
GC: initial heap(KB) 262144; maximum heap(KB) 0; virtual machine identifier
F246AFEE6B000048;  heap identifier F246AFEE6B000048.…
GC 1: starting collection, threshold allocation reached.…
GC 1: live objects 149160; collected objects 1810659; collected(KB) 240977.…
GC 1: queued for finalization 0; total soft references 77; cleared soft references 49.…
GC 1: current heap(KB) 311328; current threshold(KB) 262144.…
GC 1: collect (milliseconds) 2106.…
GC 1: current cycle allocation(KB) 17133; previous cycle allocation(KB) 262172.…
GC 1: total weak references 6; cleared weak references 1.…
GC 1: total final references 516; cleared final references 47.…
GC 1: total phantom references 0; cleared phantom references 0.…
GC 1: total old soft references 0; cleared old soft references 0.…
GC 1: total JNI global weak references 0; cleared JNI global weak references 0.…
```

The first line is output once. It confirms the application settings. At the time this output was run, the Xms setting was 256m, which is 262144 bytes. Xmx was not specified as indicated by the maximum size of 0.

The lines that follow have a number after the GC. This is the cycle number, so all these lines relate to the first garbage collection. Later a set of lines beginning with `GC 2` appear and so on.

The value 240977 at the end of the second GC 1 line indicates how much heap was recovered by the collection. The value 311328 on the fourth line indicates the size of the heap after the collection. The other value on this line 262712 is the threshold, which specifies how much the heap needs to grow before the next collection. This is based on the Xms parameter. If the application was running for some time and this is a typical workload, then these figures give a good indication of the ideal heap size. It varies between the size after collection (311328 in this example) and this size, plus the threshold (311328 + 262144).

If you increase Xms, the threshold increases, so the collections occur less often. In general, this improves performance. However if you increase it too much, then the collection has a greater impact on performance when they do occur. Also if you let the heap grow too large, then you may exceed real memory in the pool and cause faulting. On the other hand, if you have insufficient real memory, you may improve performance by reducing Xms and keeping the heap smaller to reduce faulting.

5.4.4 Static compilation

OS/400 offers the ability to statically compile Java classes and JAR files. This is done using the Create Java Program (CRTJVAPGM) command. A service program (similar to a familiar iSeries program but with multiple entry points) is created. Unlike traditional languages (for example, RPG or Cobol), this program does not appear in any library. Instead, it is attached to the Java class or JAR file. The iSeries class loader detects the presence of this service program and uses it if it is present. This saves the overhead of Just In Time (JIT) compilation and the level of optimization may be greater. This step used to be extremely valuable in the early releases of Java on the iSeries, but it is less important now since the JIT compiler has improved greatly.

If you have addressed most other performance issues, then you may want to try this technique. At V5R1, the JIT compiler was greatly enhanced, although the benefit may not be significant. Although some improvement in application startup may still be achieved.

Refer to Chapter 3, "Installing and setting up call.connect" on page 19, and Chapter 4, "Installing and setting up vendor.connect" on page 73, for the names and locations of the JAR files used by the application. These include ones that are described as "deployed" JAR files that contain the EJB classes, "logic" JAR files that contain the application logic used by the EJBs, and some utility JAR files.

You can check whether these JAR files already have a static program attached by using the Display Java Program (DSPJVAPGM) command as shown in this example:

```
DSPJVAPGM CLSF('/OrderManagement/Deployed/roseao_deployed.jar')
```

Then you see the Display Java Program Information display (Figure 5-20). If you receive the message "No Java program associated with the file", then there is no hidden static program and JIT compilation is used.

If some data is displayed, then look at the Optimization level. The highest level is 40. Also look at the value for the Classes without current Java programs parameter.

```
                       Display Java Program Information
 File name  . . . . . . . . . . . . . . . . . :   < nt/Deployed/roseao_deployed.jar
 Owner  . . . . . . . . . . . . . . . . . . . :   QSECOFR



 Java program creation information:
   File change date/time  . . . . . . . . . . :   27/11/01   10:26:19
   Java program creation date/time  . . . . . :   21/12/01   10:55:13
   Java programs  . . . . . . . . . . . . . . :
   Total classes in source  . . . . . . . . . :   182
   Classes with current Java programs . . . . :   182
   Classes without current Java programs  . . :   0
   Classes containing errors  . . . . . . . . :   0
   Optimization . . . . . . . . . . . . . . . :   40
   Enable performance collection  . . . . . . :   *NONE
   Use adopted authority  . . . . . . . . . . :   *NO
   User profile . . . . . . . . . . . . . . . :   *USER
                                                                    More...
 Press Enter to continue.

 F3=Exit    F12=Cancel
 (C) COPYRIGHT IBM CORP. 1980, 2000.
```

Figure 5-20 The DSPJVAPGM command used with a compiled JAR file

If there is no static program, the optimization level is less than 40, or some classes do not have Java programs, then the compile may be worth trying. Compilation is performed using the CRTJVAPGM command as shown in this example:

```
CRTJVAPGM CLSF('/OrderManagement/Deployed/roseao_deployed.jar') OPTIMIZE(40)
```

This command can be slow, especially for large JAR files, and you may want to submit it to batch. Repeat this command for all JAR files used by the application. You can use generic names, for example, *.jar at the end of the path. If you compile all the JAR files in the application directory at one time, then it will be very slow so then you certainly submit them to batch. Check the job log periodically. You will see a message each time it completes a JAR file so you can check the progress.

You can also apply this to individual class files but none that are used by call.connect or vendor.connect at the moment.

Outside of WebSphere, this program would be detected and used with no further effort. Unfortunately, WebSphere adds a complication. It may use its own *custom class loader* in

place of the OS/400 *system class loader*. This custom class loader does not detect the static programs.

To enable the system class loader to load the classes and detect the static programs, you must add the classpath parameter to the JVM settings. Do this in the Command line settings field of the application server settings (the same field as the –Xms parameter) as shown in the example in Figure 5-21. The parameter is –classpath followed by a space and the classpath value. The classpath value is the fully qualified name of all the JAR files separated by colons (:). There should not be any spaces in the value. It is similar to the Dependant classpath parameter in the application node settings except that it includes the deployed JAR files. Notice that the slashes in the qualified names are forward slashes (UNIX rather than Windows NT style) and that the separator is a colon rather than a semi-colon (;) (again UNIX rather than Windows NT style).
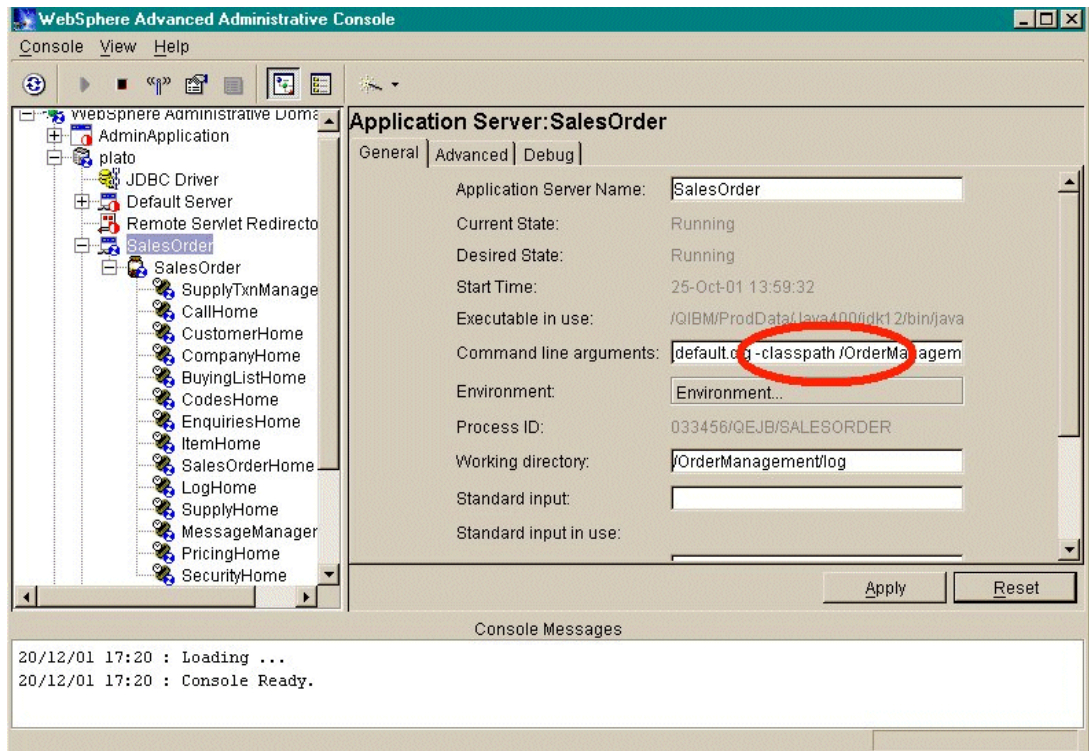
Figure 5-21   Application Server panel of the WebSphere console showing the classpath

It is possible that the parameter is too long and will be difficult to edit directly in the field of the dialog. Therefore, copy the parameter to an editor and modify it there. To copy the entire parameter, follow these steps:

1. Place the cursor in the field, and press the Home key to ensure that you are at the beginning of the field.

2. Press and hold the Shift key and then press the End key to select the entire field.

3. Press Ctrl+C to copy the field to the clip board and then press Ctrl+V to paste the parameter into a text editor (for example, Windows Notepad).

Since the line may be long, check how the editor handles word wrap. Some editors insert line feeds into long lines of text, which may be a problem. In our example, we turned off word wrap because the line is too long. To turn off word wrap in Notepad, select **Format-> Word Wrap** if a checkmark precedes the option. If there is no checkmark in front of the Word Wrap option, then Word Wrap is already turned off.

If you have a powerful editor, you may be able to replace all the colons with line feeds. This makes the parameter much easier to edit. You would rejoin the lines with the colons between each entry.

You may also want to consider starting a Windows command prompt, navigate to the directory that contains the JAR files (using a mapped drive), list the JAR files using the DIR command, and output the result to a file using redirection. he following command creates a file called jars.txt in the c:\temp directory with the names of the JAR files. The /b option suppresses all data (for example, size and date) except the names (which is convenient).

```
dir *.jar /b > c:\temp\jars.txt
```

You can then edit this file, preferably in a second session alongside the previous edit session as explained here:

1. Add the qualification, for example, /OrderManagement/deployed/ to the beginning of each line (copy and paste or possibly scan and replace).

2. Add a colon to the end of each line, except the last.

3. Delete the line feeds to join all the lines into one long line and copy the resulting long line to the clipboard.

Return to the editor session with the parameter copied from the console, add the parameter name –classpath and a space at the end of the current settings, and paste the list of JAR files from the other editor session.

An editor more powerful than Notepad is desirable, but even using Notepad is a lot easier than editing the field in the WebSphere console.

Figure 5-22 shows a command prompt and two Notepad sessions.
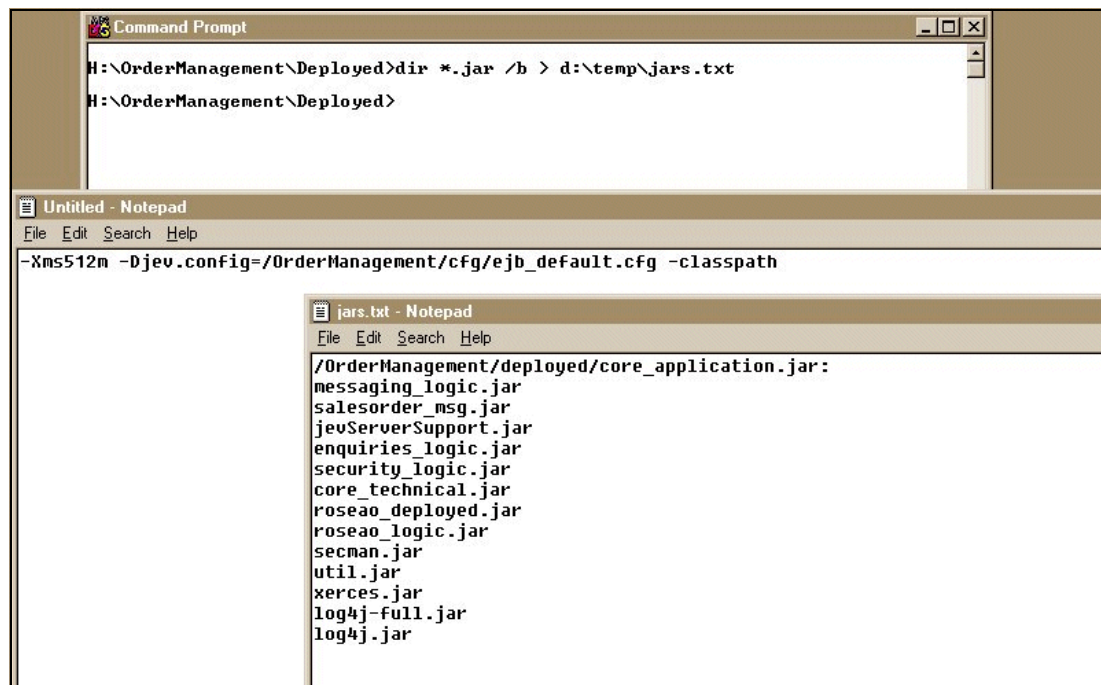
Figure 5-22   Using a command prompt and two Notepad sessions to edit the classpath

In this example, we completed the following tasks:

1. The command line arguments were copied to one Notepad session and the parameter name –classpath and a space were added at the end.

2. A file with a list of the JAR files was created and opened in another Notepad session.

3. The qualification was added to the beginning of the first line and the colon was added to the end.

The lines that follow should be amended in the same way (no colon on the last), the lines joined into one long line with no spaces, and then copied to the clipboard. Finally the long string should be pasted into the other session after –classpath.

An alternative technique is not to run the CRTJVAPGM command, but to add the following line to the command line arguments:

```
-Dos400.optimization=40
```

This sets the default optimization level to 40. If there are no attached Java programs, then they are created the next time the application is started. This technique means that the next application start will be slow, possibly so slow that it times out. But if the Java programs are created, then it confirms that your configuration was correct and they will be used.

You can verify whether the Java programs exist by using the DSPJVAPGM command. If the programs exist and you want to use this technique, then delete them by using the Delete Java Program (DLTJVAPGM) command. Note that this command does not delete the Java class file, but the hidden statically compiled program associated with it.

Enterprise bean settings

Similar to connections, beans are pooled. The default minimum pool size is 2, and the maximum is 100. Unless you have an extremely large installation, these values should work. Even if you had hundreds of clients, you would need more than 100 beans only if they were all active and more than 100 were frequently using the same bean at the same time.

However, on a moderately busy system, you may improve performance by increasing the minimum pool size and possibly by reducing the maximum. Ideally you would watch usage by using the Resource Analyzer.

These settings are controlled on the Advanced page of a bean's settings as shown in Figure 5-23.
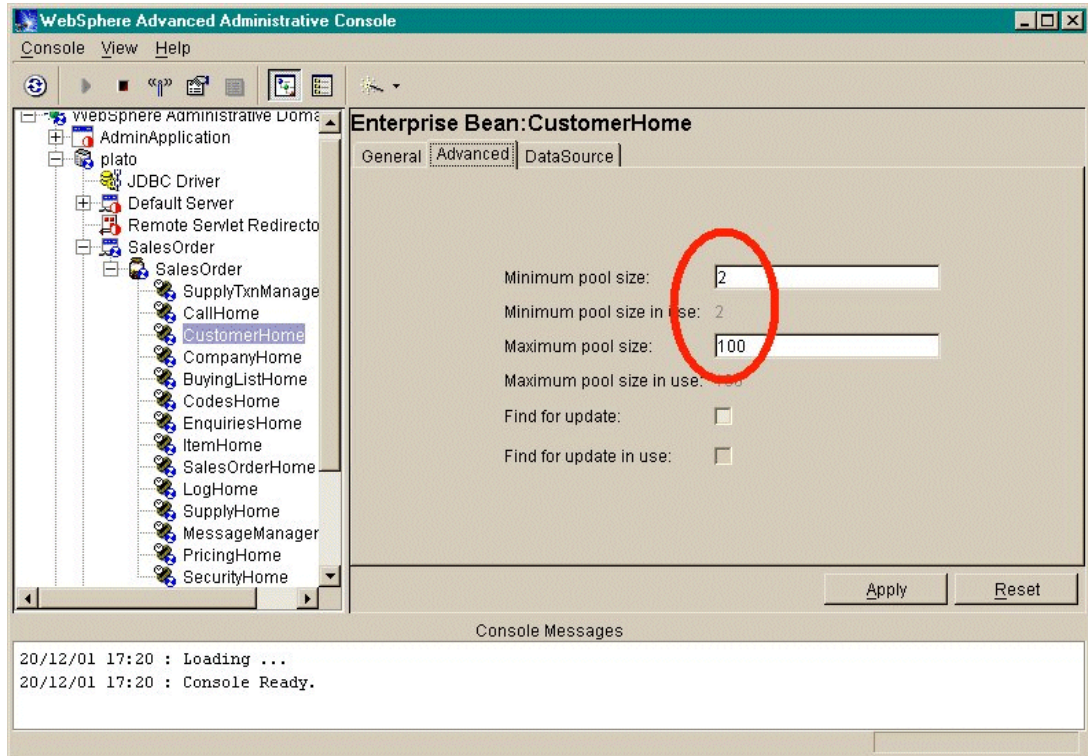
Figure 5-23   Advanced tab of the Enterprise Bean panel

Performance requirements

We have not yet determined precise guidelines for sizing these applications. Our standard tests normally use a test harness that emulates a specified number of users entering orders. A 20 second think time is used. For example, the simulated users wait for 20 seconds between creating the order header and the first line, then wait another 20 seconds between this line and the next, etc. You need to estimate whether this is similar to your usage or whether your users are likely to be enter data faster or slower than this.

For our simulated users, approximately 8 CPW is required per user. For 30 such users, a CPW of 240 is required. This assumes that sufficient memory is available. Insufficient memory constrains performance so that a system with insufficient memory may not support as many users as its CPW would imply.

Constrained performance

WebSphere has many time limits that are intended to detect and clear up failed transaction and applications. If the system can cope with the workload and it is tuned reasonably, then the default settings are normally acceptable. However if performance is poor, then there is a danger that a transaction or application exceed a time limit and be cancelled even if it has not failed.

Application settings

These limits are specified on the Advanced page of the Application Server panel.

Transaction time out

This parameter defaults to 120 seconds, that is two minutes. This means that if a transaction (one method call) has not completed in two minutes, it is cancelled by WebSphere.

If transactions fail for this reason, then the client receives such messages as `JTA Transaction Aborted`. If you see this message on the client or transactions fail, but you have not determined why, then you may want to increase this value. If you prefer long waits to failures, then it is worth it to try a large increase such as adding a zero (multiplying by 10).

Transaction inactivity time out

This is another WebSphere timing parameter that can stop a long running transaction. The default is 60000, which seems large. But unlike the other timers, it is measured in milliseconds and is, therefore, only one minute. You may want to increase this for similar reasons as the previous setting.

Figure 5-24 shows Transaction time out and Transaction inactivity time out parameters in the top half of the Advanced page of the Application Server panel.
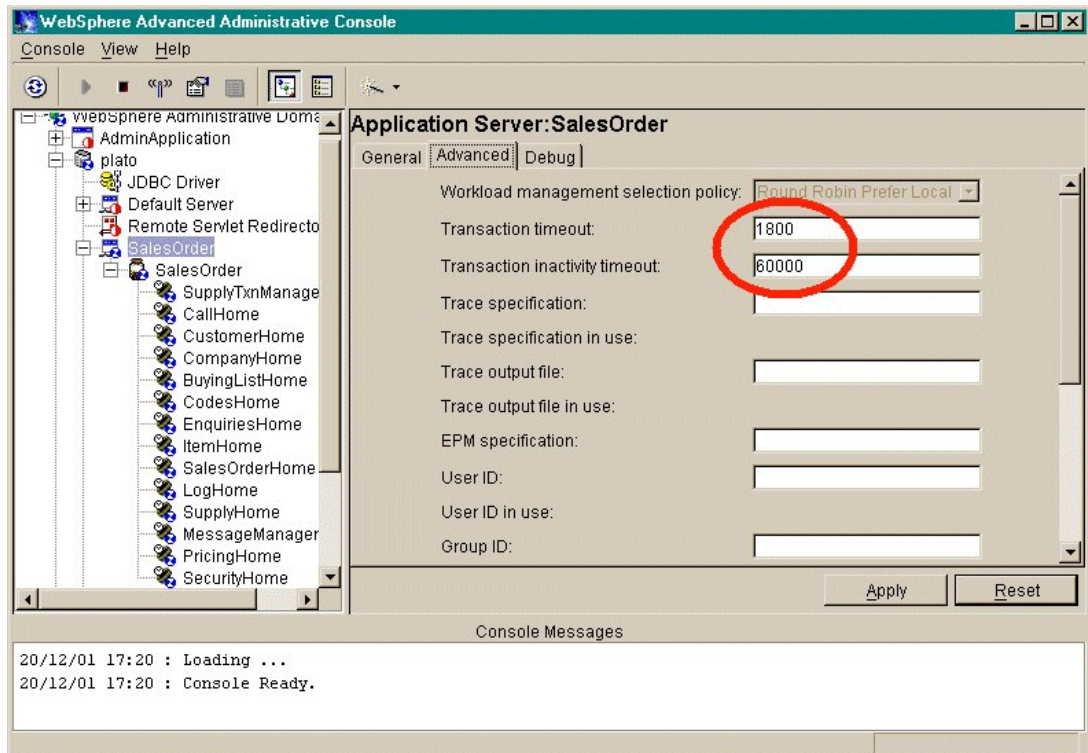
Figure 5-24   Advanced page of the Application Server panel

Ping interval and timeout

The administrative server pings your application regularly to check whether it is still operating. If it fails, then it is ended and restarted. The *ping interval* controls how often this is performed. The *ping timeout* controls how long the administrative server waits before it assumes failure. The *ping initial timeout* is similar to ping timeout except that it covers the initial ping after the application is started. It is usual to use a larger value for the initial time out.

If performance is poor, then increase all of these parameters to avoid a slow application being mistaken for a failed application. If your application server is periodically ended and restarted, then this may explain the poor performance. A "`SIGKILL received`" message in the job log would be normal in this case.

When performance is poor, consider simply adding a zero to the timeout values. Note that the time out values are most important. However, there is no point in frequent pings if the time out is large. Therefore, increase the ping interval as well.

"Ping" in this context refers to sending a simple test message, which requires only a simple acknowledgement. It is a similar idea to the network command PING, which can be used to test a TCP/IP network.

Figure 5-25 shows the Ping interval, Ping timeout, and Ping initial timeout parameters at the bottom half of the Advanced page of the Application Server panel.
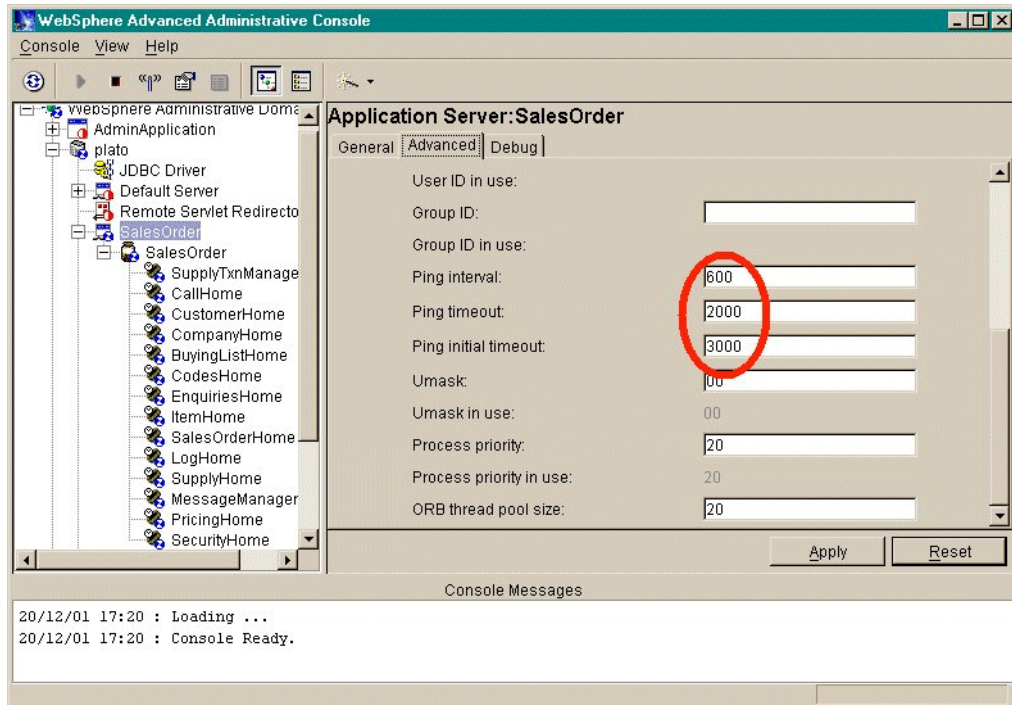

Figure 5-25   Advanced tab of the Application Server panel showing the ping settings

Datasource settings

The datasource settings are controlled using the WebSphere console. To access these settings, select the datasource and then select the **Advanced** tab.

Refer to 5.3.2, "Stateless connections" on page 123, to learn about the important datasource settings of minimum and maximum connection pool sizes.

Other settings are occasionally useful, especially when the system is performing poorly, as explained in the following sections.

Figure 5-26 shows the Advanced page of the DataSource panel.
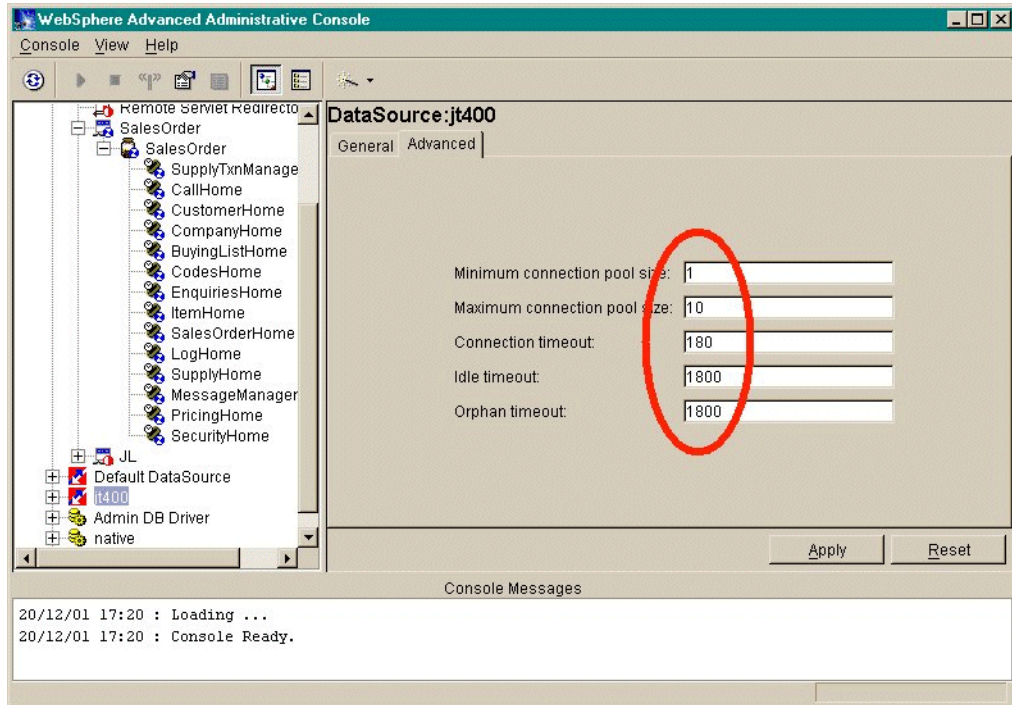
Figure 5-26   The Advanced page of the DataSource panel

Connection time out

This is the maximum time that WebSphere waits for a connection when all the connections in the pool are in use.

If you frequently encounter connection time outs, then consider increasing the size of the connection pool. See 5.3.2, "Stateless connections" on page 123, and 5.3.3, "Total connections and SQL server jobs" on page 124, for more information.

If the problem occurs only occasionally during extremely busy periods and you do not want to use additional SQL server jobs, then increase this limit. In general, the problem is better addressed by increasing the pool size.

Idle time out

If the current size of the connection pool is above the minimum size, then connections are released after they are idle for this time. They then return to the OS/400 pool. See 5.2.1, "SQL server job configuration" on page 109, and 5.3.3, "Total connections and SQL server jobs" on page 124, for more information. Depending on the SQL server job configuration, they may then be as idle prestart jobs or released later in a similar way.

If there are times when WebSphere and non-WebSphere applications are busy, then setting the minimum pool size less than peak usage may improve the performance of other applications when WebSphere is below its peak. However, if WebSphere usage is fairly constant or when it is idle and so are other applications, performance may be better with a larger minimum pool size. Then this parameter will not be so important.

Orphan time out

This is the maximum amount of time that a transaction can hold a transaction. If it is exceeded, then the connection is returned to the pool anyway. This is designed to prevent failed or hung transacting monopolizing connections. However, if performance is poor, then a valid transaction may have its connection removed and when it tries to use it the next time. It

receives an exception indicating that the connection was pre-empted. If valid transactions suffer from this problem, then you may want to increase this setting.

6

Tips and techniques

This chapter offers various tips and techniques for using Geac System21 products.

137

6.1 The iSeries integrated file system

Several releases ago, IBM reorganized considerably the file system of the iSeries server. A new UNIX- or Windows NT-like file system called the integrated file system (IFS) was introduced. The original file system, for example QSYS, other libraries, and their contents became part of this new system.

Files in new areas of the IFS are quite different than the familiar database files of the iSeries. They do not have fixed length records. In fact, they do not have a structure known to the operating system. They are simply stored bytes. The data may be readable text in ASCII, EBCDIC or Unicode. It could be Java byte code, PC executable code, or many other types. Some may be suitable for viewing and others may not be. You need to know which programs are intended to read and write these files. The extension (the portion of the name after the dot) often suggests the nature of the contents but it is not exact. A file whose name ends in `.java` may be Java source code, but it could be anything.

When necessary, these files are called *stream files* to distinguish them from *database files*. They are also commonly called *IFS files*. To a UNIX or Windows NT user, they are simply files even if they contain executable program code.

WebSphere and its applications make considerable use of stream files. Java source and byte code is stored in stream files. The source code is in ASCII, and the byte code is not humanly readable. Configuration and log files used by these applications are typically stream files, which are usually ASCII, but may be EBCDIC.

Many experienced iSeries users are not familiar with these stream files, so we added a number of topics that deal with them in this section.

The IFS is similar to UNIX, except that in general, file names are not case sensitive. The system remembers the case that is used when the file is created, but it can be addressed later using a different case. Although this is not always true, we recommend that you use consistent case. (Some UNIX knowledge will be valuable for installing and managing a WebSphere application.) The IFS even supports links, both hard and soft, but they are not used by System21 applications.

The IFS is similar to the file system of Windows NT, but less so than UNIX. Here are some significant differences from Windows NT:

► Although the iSeries has disk drives, they are not reflected in file naming so there is no C: drive, D: drive, etc.

► The directory separator is / rather than \.

► The root directory is called /, not C:\. All iSeries storage is somewhere within this directory.

One aspect in which the IFS is more like Windows NT than UNIX is regard to case. Outside the special directory QOpenSys (see the following list), file names are not case sensitive. For example, FRED, Fred, and fred all refer to the same file. Like Windows NT, the IFS remembers the case but allows access through names with different cases.

Certain top-level directories have special properties, which include:

► **QDLS**: The old folder / document file system. This exists only for compatibility. This is not used by System21 applications and we do not recommend using it.

► **QOpenSys**: Within this directory, names are case sensitive. This is intended for porting UNIX applications. It is not used by System21 applications.

► **QOpt**: Optical storage, such as CD drives. This is used during WebSphere installation.

► **QIBM**: This directory does not have any special property, but most IBM supplied files are within this directory. Two important subdirectories are ProdData and UserData. Within each of these, there is usually a directory for each IBM product that uses stream files.

For example, both contain a directory called WebASAdv. ProdData normally contains files that would not change in normal usage. UserData contains files that may change during normal usage. If the product is deleted, then the directory within ProdData is deleted, but the one within UserData is not deleted. If you want to completely remove the product, then you need to manually delete. Be sure to first back it up.

► **QNTC**: This directory enables the iSeries to be a client of Windows NT file shares. It may contain subdirectories named after Windows NT servers. Within these directories, you may see directories named after shares offered by that system. And within these directories, you may see files from the Windows NT system. This feature is not used by System21 applications.

► **QSYS.LIB**: All traditional AS/400 objects are within this directory. If you explore this directory, you will find all your familiar libraries, files, programs, etc. but with new style names. In general, it is better to use the traditional commands to deal with objects in this area of the file system.

There are additional special Q directories with special properties. With each release, IBM may add more of these directories.

Files directly in the root or in normal directories should be quite UNIX or Windows NT like.

6.1.1 Using File Transfer Protocol (FTP) with the iSeries IFS

If you configured FTP on your iSeries, then you may want to use it to transfer files to and from other systems. FTP on the iSeries is similar to FTP on most systems, except for the naming systems of the iSeries. That is FTP was introduced to the iSeries before the IFS, so it has two naming modes. One mode attempts to match UNIX-type names to the original file system, and the other mode uses the IFS names.

The iSeries defaults to the older non-IFS mode. In the older non-IFS mode, only a few object types, such as physical files, may be accessed. A source physical file is accessed with a name of this form:

```
library/file.member
```

For a typical data physical file with one member, the member may be omitted. It is unlikely that you need to use this form of naming when you deal with the .connect applications.

It is more likely that you will want to access stream files outside the traditional file system using the IFS names. To switch to the new naming mode, use the following command:

```
quote site namefmt 1
```

Some FTP clients may need a slightly different command.

Alternatively if you use **cd** to change directory to an IFS type name beginning with /, then the mode is switched. For example, you can use the following command to switch to IFS names and the root directory:

```
cd /
```

If you ever need to send a save file to or from an iSeries, you must switch the transfer mode to binary first. Otherwise, the file will be corrupted. Use the following command:

```
bin
```

6.1.2 Mapping a PC drive to the iSeries IFS

An easy way to explore and view the IFS is to map a PC drive to some or all of the IFS and use PC tools such as Windows Explorer. To do this, you must set up a Windows-style file share on the iSeries. You must also install Operations Navigator and create a connection to the appropriate iSeries server.

To create a share, follow these steps:

1. Start Operations Navigator.

2. Expand the system you need. Log on if necessary.

3. Expand **File Systems**, and select **File Shares**.

4. Several IBM default shares may be displayed. These map to portions of the IFS. You may use these immediately if you prefer or you can add more. You may find it useful to create a share that allows access to the IFS directories containing the System21 applications.

   The simplest share is one that mapped to the root of the IFS. This allows access to the entire IFS from the PC (subject to authority), but this can be a security exposure. Only do this if you are quite familiar with iSeries and Windows NT security.

   Right-click **File Shares** in the left panel of Operations Navigator and select **Open AS/400 NetServer**.

5. The AS/400 NetServer window opens. In the new window, right-click the **Shared Objects** item and select **New-> File**.

6. The AS/400 NetServer File Share window (Figure 6-1) opens.

   a. Enter a share name. If the share is to the root of the IFS, then use the name ROOT.

   b. Enter a description (optional).

   c. Set the access to **Read only** or **Read/Write** depending on your intended usage. A Read/Write share to the root can be a significant security exposure. Read only is safer to use, but remember that it may still allow viewing of confidential data.

   d. Set the maximum number of users (optional).

   e. Enter the path name as /.

   f. Click **OK**.
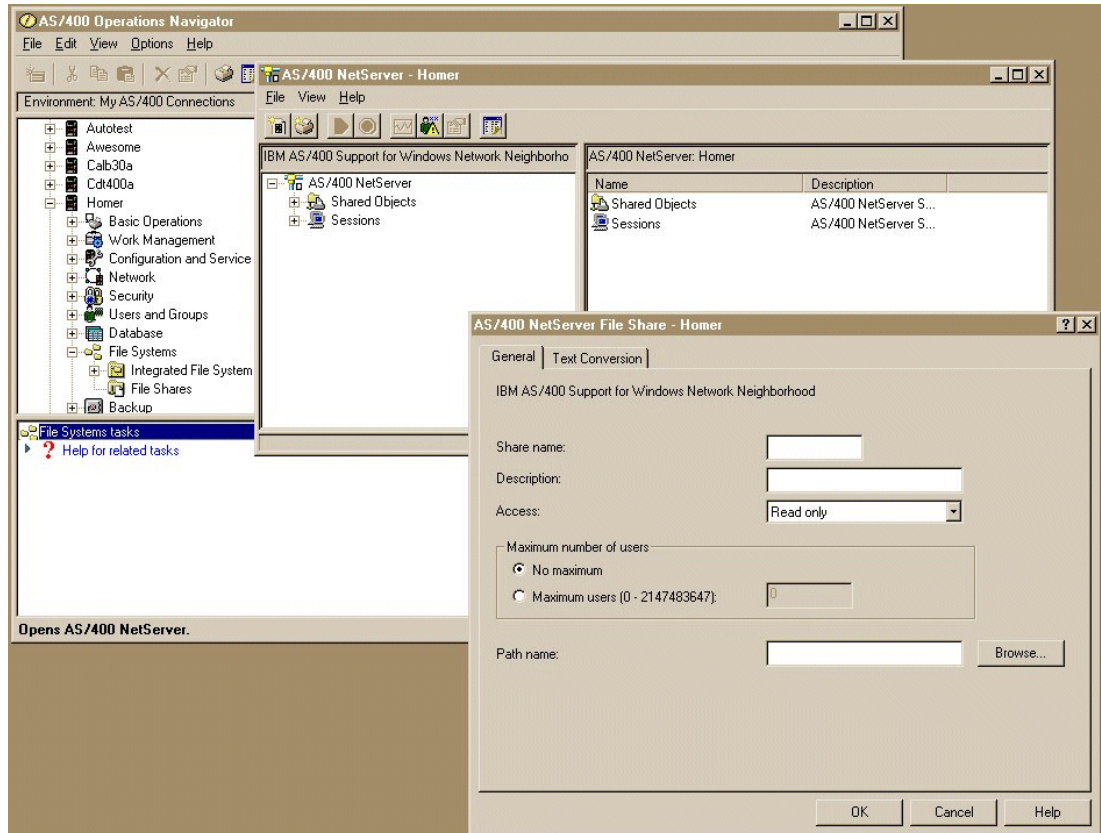
Figure 6-1   Creating a file share

To create a more specific share, use a different name and select an appropriate directory in the path name field. You can find this directory using the Browse button. You may want to create a share to the /OrderManagement directory of the application. A logical name is to use OrderManagement.

Mapping a PC drive to one of these shares is like mapping a drive to a share on a Windows NT server. Successfully accessing the iSeries by name from Windows Explorer is quite sensitive to the configuration of your iSeries and PC. One problem we found is that if you do not have a WINS server configured, then you may be unable to address the iSeries by name in Windows Explorer. In most cases, you can address the iSeries by its IP address even if you cannot address it by name.

You may need to log onto the iSeries server. You can avoid this if your Windows NT and iSeries users and passwords are the same. If you do not log on as QSECOFR, then you may find that your access to certain files and directories is restricted. Consequently, it is tempting to log on as QSECOFR. If you do, then be careful since it is easy to accidentally delete, move, or even damage important files.

If you are using a share that maps to a directory other than the root, for example, /OrderManagement, then the name of this directory does not appear in Windows Explorer. The root of the mapped drive is the contents of the directory on the iSeries. The immediate subdirectories of the iSeries directory appear to be top-level directories in the PC drive.

6.1.3 Editing an iSeries stream file using a PC editor

If you exposed some or all of your iSeries IFS using a file share, then you can edit many configuration, log, and source files with a PC editor. Remember that not all stream files contain readable text. If you accidentally open one and it is not readable text, then be sure to exit without saving. If you save the file, you may corrupt it even though you did not make any deliberate changes.

You must use the editor with care. Ensure that it can read and write simple ASCII files. Word processors may not be suitable or may only be suitable with care.

Notepad (in the Windows Accessories folder) is suitable for some files, but not all files. Some iSeries files use the UNIX convention of a line feed between lines rather than the DOS convention of using a carriage return and line feed. Notepad does not understand these files and presents them in an incorrect way. An advantage of Notepad is that it can open a file that is in use but cannot save that file while it is in use.

Wordpad (another Windows accessory) may also suitable for some files. Wordpad can save in Word document format (be careful not to use this option). It copes with the UNIX style line feeds that some iSeries files use. However, it cannot open files that are in use.

The best option may be to use a PC editor that is intended for programmer use. There are may available. Whichever editor you use, you may find that some files that you expect to be text are unreadable. Be careful to exit these files without saving. If the file contains many @ symbols, then it is possible that it is a text file, but in EBCDIC. For more information, see the following section.

6.1.4 Stream files and CCSID

CCSIDs are labels for a file and some other objects. They indicate the code page of the data within them. When database or stream files are labelled with an appropriate CCSID, OS/400 performs any necessary conversion when the file is read or written by OS/400 programs.

If a stream file is created on the PC, for example, by creating a new text file in Windows Explorer, it is given an ASCII CCSID. Most OS/400 software can use this. That is, although the software may expect EBCDIC, the operating system can convert the data as required.

If the file is created by Java on the iSeries server, it also gets an ASCII CCSID so you should be able to read it on both the PC and the iSeries (if it is a text file). However, if the file is created on the iSeries by something other than Java, it may get an EBCDIC CCSID. This is usable by OS/400 software but not by PC software as mentioned in the previous section. You are likely to see many @ characters because the EBCIDC code for space is the same as the ASCII code for @. A situation in which this may occur is when a file is created using redirection from a Qshell command.

You can create an ASCII file in the Qshell on the iSeries server by using the **touch** command. This command can be useful in scripts. If the file is created with an ASCII CCSID, then data written to it by following iSeries software should be translated. The result should be readable from a PC.

It is possible to change the CCSID of a stream file after it is created without changing the data already in the file.

If it is easy to recreate the file, then the easiest approach may be to delete the file, create it again with an ASCII CCSID, and run the application again. You should now be able to view the file from a PC. To create the file with an ASCII CCSID, either create it from a PC using a mapped drive or by using the **touch** command.

If it is not easy to create the file again, then you can view it by using an OS/400 command, such as Edit File (EDTF), or copy the file to a file with an ASCII CCSID using a command like this example:

```
CPY OBJ(ebcdic-file-name) TOOBJ(ascii-file-name)
   TOCCSID(819) DTAFMT(*TXT)
```

Here *ebcdic-file-name* is the name of the file that you cannot read on the PC and *ascii-file-name* is a new name that does not yet exist. You may need to fully qualify both names (unless you used the **cd** command first). If the file is text in EBCDIC, you should be able to read the new file using a PC editor.

For more information about the **touch** and **setccsid** commands, see 6.2.2, "The touch and setccsid commands" on page 149.

6.1.5 The cd command

An OS/400 job has a current directory in the same way that a Windows or UNIX session has a current directory. This is a similar concept to the current library, but it is not linked. Unlike Windows and UNIX, the current directory does not affect many things. Programs are still found using the library list, which, like the current library, is independent of the current directory.

Like Windows and UNIX, you can set the current directory with the **cd** command. You may also use the more traditional name CHGCURDIR. Like UNIX, but not Windows, the directory separator is / rather than \. Also like UNIX, but not Windows, there is no notion of drives. Only / is the root of all storage. Unlike UNIX or Windows, a qualified name (that is one that contains a /) needs to be in single quotation marks as shown in this example:

```
cd '/OrderManagement/log'
```

An exception is / by itself, which does not need single quotation marks.

As in UNIX and Windows, you can use the special name .. to mean the parent of the current directory.

A home directory is stored in your user profile. This is usually /home/user-name. If this directory exists, then it is set as your current directory when you sign on. If it does not exist, then your current directory is the root. You can change this home directory using the Change Profile (CHGPRF) or Change User Profile (CHGUSRPRF) command.

6.1.6 Managing stream files with the OS/400 WRKLNK command

You can use the Work with Object Links (WRKLNK) command (Figure 6-2) to explore and manage the IFS. It is a dumb terminal equivalent of Windows Explorer. For most purposes, it is easier to use Windows Explorer against a mapped drive, although WRKLNK is occasionally useful.

A time when WRKLNK is useful is when you do not have a share for the IFS directory in which you are interested. In some cases, WRKLNK can be easier or perform better. It allows you to control authority, which is not possible through Windows Explorer.

The WRKLNK command without parameters shows the contents of the current directory. Alternatively, you may follow the command with a parameter that indicates which file or directory to show. If you want to explore the entire file system and you have not set your current directory, use the following command:

```
wrklnk /
```

Like cd, qualified names, except for /, must be in single quotation marks.

```
                          Work with Object Links

Directory . . . . :   /

Type options, press Enter.
  2=Edit    3=Copy    4=Remove   5=Display   7=Rename    8=Display attributes
  11=Change current directory ...

Opt    Object link          Type       Attribute    Text
  ■_      .                  DIR
  __      ..                 DIR
  __      b2b                DIR
  __      com                DIR
  __      dev                DIR
  __      etc                DIR
  __      home               DIR
  __      html               DIR
  __      java               DIR
                                                                   More...
Parameters or command
===> _____
F3=Exit    F4=Prompt    F5=Refresh    F9=Retrieve   F12=Cancel    F17=Position to
F22=Display entire field            F23=More options
```

Figure 6-2   The WRKLNK command showing the root directory

Option 5 (Display) is rather important. If you type it next to a file, the EDTF command is called in display mode. If you type it next to a directory, the contents of that directory are shown. If you enter a directory using option 5, then you can return to the parent directory by pressing F12. Therefore, if you started with the root, you view the entire storage of the system (but it can take a long time). If you go into the /QSYS.LIB directory, then you see all the traditional objects, but this can be slow.

Option 2, which calls EDTF in edit mode, is also useful. Use this option only if you know that it is appropriate to edit the file.

There are also options to copy, move, rename, and delete files and directories, but you must use these options with caution. Although these options may be less friendly than the equivalent options in Windows Explorer, they can be faster, especially with large files. If you move or copy a stream file using Windows Explorer, then it sends the file down to your PC and back. However, if you use the WRKLNK command, it moves only with the iSeries. In some cases, it does not move the object but only updates the directories. If you move a Java class or JAR file that has an associated static program, then it is retained if you use the WRKLNK command, but not if you use Windows Explorer.

The option to delete a directory only deletes an empty directory.

Some options, among these of which is these option 9 to control authority, are not shown until you press F23. See the following section to learn more about stream file authority.

For more information about using the EDTF command, see 6.1.9, "Editing an iSeries stream file using the OS/400 EDTF command" on page 146.

6.1.7 Other stream file commands

There are many other stream file commands. Most of them are similar to traditional commands that deal with objects in general. For a traditional command with OBJ in its name, there is often a stream file command that is similar but that does not have OBJ. For example, MOV, RNM, and SAV are commands to move, rename, and save stream files.

A few commands do not follow the pattern. For example, the command to delete a stream file is RMVLNK, which may also be called DEL. The command to manage authority is WRKAUT.

The Save Object (SAV) command is slightly confusing since it does not want only the save device name (for example, TAP01), but the name of its device description in the new IFS syntax. An example is /qsys.lib/tap01.devd.

Figure 6-3 shows the Save Object display.

```
                              Save Object (SAV)

 Type choices, press Enter.

 Device . . . . . . . . . . . . . > █/qsys.lib/tap01.devd'_____
 _____
                 + for more values   _____
 _____
 Objects:                           _
   Name . . . . . . . . . . . . . > '/OrderManagement'_____
 _____
   Include or omit  . . . . . . .   *INCLUDE       *INCLUDE, *OMIT
                 + for more values _
 Directory subtree  . . . . . . .   *ALL           *ALL, *DIR, *NONE, *OBJ
 Save active  . . . . . . . . . .   *NO            *NO, *YES, *SYNC




                                                                       Bottom
 F3=Exit    F4=Prompt    F5=Refresh    F10=Additional parameters    F12=Cancel
 F13=How to use this display         F24=More keys
```

Figure 6-3   Using the SAV command to save the OrderManagement directory

You can create directories using the CRTDIR command, which may also be called MKDIR or MD. And you may delete directories using the RMVDIR command, which may also be called RMDIR or RD.

6.1.8 Stream file authority

Stream files and directories are subject to authority in a similar way to traditional objects. The authority of a stream file or directory may be controlled using the WRKAUT command or by finding it with WRKLNK and typing option 9 next to it.

The most important authorities are the data authorities. These are UNIX style authorities. For each user, these may be a combination of R, W, and X. R indicates permission to read. W indicates permission to write (update) to the file and also to delete the file. For UNIX, X indicates execute permission, which is not relevant to the iSeries, since programs are not stored in stream files. However, on both UNIX and the iSeries, X for a directory indicates permission to set the directory as your current directory.

Lack of X permission to a directory does not prevent access to its contents. If a user has permission to a file within the directory, then they can still access it using a qualified name. Lack of R permission to a directory prevents access to its contents.

When a directory is created, its authority is the same as its parent. Therefore, if you create a directory in the root, it has the same authority as the root directory. Before you change the authority of the root directory, we recommend that you check the authority of your directory after it is created. If the authority is inappropriate and you already created subdirectories, then you must correct their authority as well. If the authority of the parent directory is changed after the subdirectories are created, then its authority is not changed.

A common authority mistake is to fail to allow QEJB or OMUSER write authority to the log directory and others in which they need to write files. QEJB is the user used by WebSphere jobs. OMUSER is a typical user for jobs such as JMS. It can be particularly confusing if the problem is lack of authority to the log directory since the application will be unable to log the problem.

Another common problem can be caused by inconsistent use of users. If you run jobs, such as JMS as QSECOFR and then later as OMUSER, you may have an authority problem. It is likely that the log files created when QSECOFR was used cannot be overwritten by the normal user used later.

6.1.9 Editing an iSeries stream file using the OS/400 EDTF command

In most cases, the easiest way to edit a stream file is with a PC editor on a mapped drive. The ETDF command is useful in a few circumstances:

► The file may be in a directory that is not accessible through any share.
► The file may be EBCDIC.
► It may simply be convenient, for example, using the WRKLNK command.

There are several ways to start EDTF. For example, one way is to enter the EDTF command followed by the stream file name. If the name is not qualified, then the current directory is searched. It is not common to set the current directory. Therefore, you may need to full qualify the file name. As for most stream file commands, if the name is qualified, then you need to add single quotation marks around it as shown here:

```
edtf '/OrderManagement/log/stdout.txt'
```

Another way is to enter the EDTF command followed by a directory name such as /. Then you can explore the IFS in a way that is similar to the WRKLNK command. When you find the file, type option 5 (display) or option 2 (edit). You may perform some operations on files from the EDTF file selector. There is even an option that WRKLNK lacks, which is *recursive delete*. This deletes a directory and its contents. The delete within WRKLNK deletes only an empty directory. Of course, you must use recursive delete with caution.

You can also find the file using the WRKLNK command and type option 5 (display) or option 2 (edit) next to it the file.

EDTF is a simple dumb terminal editor that should be familiar to anyone who has used SEU (the dumb terminal source editor on the iSeries). There are no sequence numbers but there is an entry field at the left of each line into which SEU like commands may be typed.

If you do not make any changes, then press F3 or F12 to exit the editor. Unlike SEU, there is no exit screen. If you make a change, when you press either key, you see a message indicating that the file has changed. If you press F3 again, the file is saved and you exit. If you press F12, then the file is not saved and you exit.

As for editing files with a PC editor, ensure that it is safe and appropriate to edit them. If you start EDTF and the file is not readable or you do not understand its contents, then be careful to exit without saving.

To use EDTF, you should be familiar with SEU and other OS/400 commands.

6.2 The Qshell

Java and other non-traditional iSeries software often requires programming techniques that do not fit well into traditional iSeries facilities such as its structured commands and its command entry screen. The UNIX or Windows NT equivalent of the command entry screen is rather different.

The DOS command screen is probably familiar to most computer users. The output of simple commands is sent back to the command screen and when the screen is full, it automatically

scrolls. There are few restrictions on commands and their parameters. The commands can be rather inconsistent and help can be non-existent or very basic. On the other hand, there are some useful features such as input and output redirection and piping.
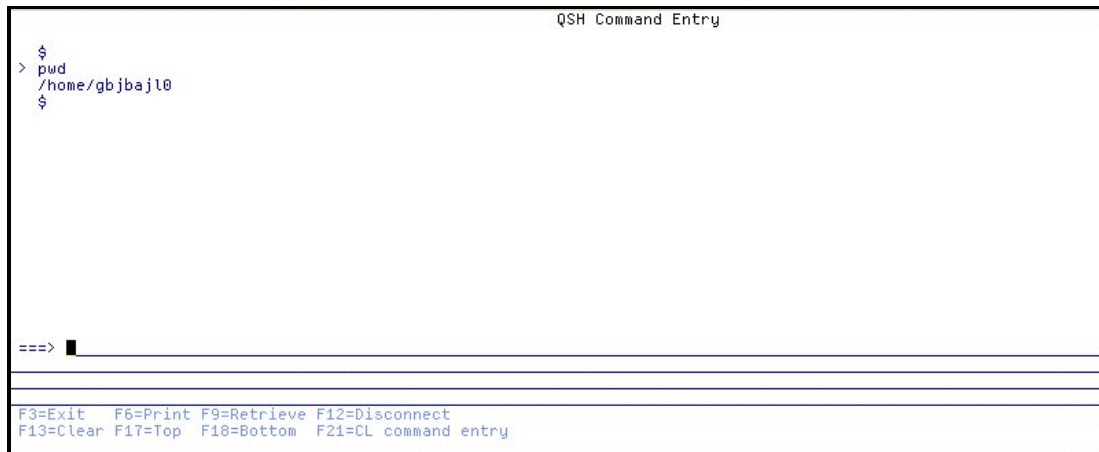
A typical UNIX command screen, usually called a *shell*, is similar although the range of commands and their syntax are not the same as DOS.

For the sake of software that expects commands or programming techniques of this sort, IBM has added a utility called the *Qshell* to OS/400.

UNIX users will find Qshell very familiar. Experience of the DOS command screen helps but not as much as UNIX experience helps. Qshell is not identical to either UNIX or DOS, mainly due to limitations of the 5250 data stream that does not permit a scrolling effect like the UNIX or DOS screens.

You can start Qshell from an ordinary OS/400 command entry display or menu by using either the Start Qshell (STRQSH) or QSH command. Then you see the QSH Command Entry display (Figure 6-4), which a mostly blank display with a line beginning with a $ and a large entry field at the bottom. In our example, the **pwd** command was already executed.

This display is similar to the normal Command Entry (QCMD) display, but it recognizes a completely different set of commands. You cannot use normal OS/400 commands in QSH, and you cannot use QSH commands in the Command Entry. There are few commands common to both (for example, CD and JAVA), but even in these cases they are not quite identical.

```
                                    QSH Command Entry

   $
>  pwd
   /home/gbjbajl0
   $

















 ===> █

 F3=Exit   F6=Print F9=Retrieve F12=Disconnect
 F13=Clear F17=Top  F18=Bottom  F21=CL command entry
```

Figure 6-4   The QSH Command Entry display

If you do not see a $ but instead another character such as ¢ or £, then you may have a code page error in your terminal or emulator setup. Ensure that your emulator configuration and the device description that you are using match your terminal. The problem may also be caused by using a user profile or system value CCSID of 65535. If you only plan to execute simple commands, then you may be able to ignore the problem. For anything more complex, we recommend that you correct the problem.

The $ is called a prompt. It is output at the beginning of the session and after the completion of each command. It indicates that the shell is ready for a command. In general, the keyboard is not locked during the execution of a command. You may be able to type another command, but it cannot not execute until the current command has completed. This can be confusing since if the current command generates any output, because it follows the waiting command. Be sure to wait for the $ prompt after each command. UNIX shells often also use $ as the

prompt. DOS command screens commonly use the currently drive and directory name followed by a >. You can usually type additional commands before the current one has completed, but they may not appear until the current command has completed.

The Qshell has a current directory that is similar to that of UNIX and OS/400. A DOS command session also has a current directory but one that Qshell does not have. The current directory is controlled with the **cd** command. This is similar to the OS/400 CD command, but you do not need the single quotation marks when you use qualified names. To check the current directory, use the **pwd** command.

You can exit Qshell by pressing either F3 or F12. If you use F3, then your session is ended. Then if you use QSH again, you start another session. If you use F12, then your session is detached from the OS/400 job that started it and you can perform other OS/400 work. If you use QSH later again from the same OS/400 job, you resume the session where you left it.

> **Note:** QSH commands do not actually execute in your interactive job. When you start QSH, a job called QZSHSH is started. The commands are sent to this job for execution and the results are sent back. This job is ended by F3, but not by F12. Some commands may cause further jobs to be used. Some of these jobs may send completion messages to your user message queue. If this is set to break, they may interrupt your session.

The STRQSH and QSH commands have an optional parameter. This can be a single Qshell command or several commands separated by a semi-colon (;). This is of little use interactively since it may be easier to start QSH normally and then type the command. However, it can be useful if you want to execute some Qshell commands inside a CL program. If the series of commands is not very simple, then it may be better to store them in a Qshell script and use the QSH command to execute this script.

If you use the QSH command interactively with a parameter and the command generates some output, the QSH command display appears. You need to press Enter to proceed. If the command does not produce any output, then the screen is not shown. If you do not want QSH to stop, then make sure that your command does not produce any output, possibly by using redirection.

6.2.1 Managing stream files with Qshell commands

You can use many UNIX-like commands in Qshell. Not many of these are useful unless you happen to be familiar with UNIX. In unusual circumstances, these commands can be useful.

One such case is if you accidentally create a file with a character that has a special meaning to Windows but not to OS/400. Examples of such characters are the colon (:) and backslash (\). You may create a file with these characters in its name, but and this file may be difficult to access using Windows Explorer and even when using WRKLNK.

An example is if you accidentally specify the log file as c:\OrderManagement\log\OrderManagement.log. This may cause a file of this name to be created in whichever directory was current at the time (probably specified as the working directory in the WebSphere application server properties).

You can see this file in Windows Explorer, but you cannot delete, move, or rename it. When you try, you may see an error message telling you that the file does not exist. Worse still, you may process a local PC file that happens to have this name.

The WRKLNK command also shows you the file but does not let you delete or process it. This is more surprising since the characters do not have a special meaning to OS/400.

Even Qshell may be confused by the unusual characters. However in this case, there is a solution. That is you may quote the name. The easiest way to delete the problem file is to navigate into its directory using cd and then delete it by using the `rm` command. Enter the file name in double quotation marks to prevent Qshell being confused by any special characters, for example:

```
rm "c:\OrderManagement\log\OrderManagement.log"
```

If you want to use the file, then you can use Qshell commands. It may be easier, however, to rename it to a normal name and go back to familiar techniques. Oddly the UNIX command for rename is `mv` (it is also the move command), for example:

```
rm "c:\OrderManagement\log\OrderManagement.log" OrderManagement.log
```

6.2.2 The touch and setccsid commands

These `touch` and `setccsid` Qshell commands are occasionally useful.

`touch` is a UNIX-like command. On UNIX, it simply sets the last update time of the stream file to the current time. Or if the file does not exist, it creates a new empty stream file.

An OS/400 enhancement to the `touch` command makes it more useful. It has an optional parameter –C that you can use to set the CCSID of the file. The option must be an uppercase C, that should be followed by a space, the desired CCSID value, and the file name. The following command creates the file jms.log in the directory /OrderManagement/log. The file will be empty and have the ASCII CCSID 819.

```
touch –C 819 /OrderManagement/log/jms.log
```

Further data written to the file by OS/400 commands should be translated to the specified code page. This should enable the data to be easily read on a PC using a mapped drive (if it is textual data).

If the file already exists, then the code page is not changed. You can use the `setccsid` command instead. But it may be appropriate to delete the file and create it again empty using the `touch` command. You can delete the file by using the `rm` command.

The `setccsid` command changes the CCSID of a stream file to a specified value. Note that the syntax is a little different. The command is followed by the desired code page and then the file name as shown in this example:

```
setccsid 819 /OrderManagement/log/jms.log
```

This command does not translate any data already in the file. If the file already exists and contains data that you cannot read on a PC, then this command will not help. In fact, it can make the situation worse, since after the command, the data may be misinterpreted on the iSeries server.

6.2.3 Viewing an iSeries stream file using the Qshell tail command

There are a number of Qshell commands for viewing files.

▶ `cat`: Displays the contents of the file in Qshell.
▶ `head`: Displays the first few lines of a file.
▶ `tail`: Displays the last few lines of a file.

These commands are rarely useful. However, there is an option in the `tail` command that can be useful. If you specify the option –f between the `tail` command and the file name, then you can see the last few lines of the file and then monitor the file for more lines being written.

This can be useful for monitoring log files in real time. The session in which you use this command remains in use. When you are finished, use System Request 2 to stop the command as shown here:

```
tail -f /OrderManagement/log.stdout.txt
```

This command shows the last few lines of the standard output file of the Order Management application. If more lines are written, they are output. This is mentioned in 5.4.3, "Verbose garbage collection" on page 126.

6.2.4 Qshell scripts

It is possible to save Qshell commands in stream files for later execution. This is similar to script programming in UNIX and to DOS batch files. The concept is similar to CL programs, but the details are different.

This is a complex subject considering that there are some powerful programming techniques available. We only briefly describe a few simple abilities. Familiarity with DOS batch file programming can help, but there are quite a few differences.

Qshell scripts are stored in stream files in the IFS. You can create and edit them using any of the editing techniques described earlier such as a PC editor against a mapped drive or the OS/400 EDTF command. If you create the script with EDTF first, then it may be given an EBCDIC CCSID and you won't be able to edit it from the PC later.

The names, or even the extensions of the names, of scripts are not restricted as DOS batch files are. There are two common conventions. Names without any extension may be used, for example StartSalesApplication, but some PC editors and tools may have problems with such files. The extension .sh is often used, but this is required when the script is executed, unlike DOS where the .bat extension is not required when executing the file. The file names are not restricted to 10 characters and may be very long. Unlike UNIX, case is not significant. It is common to use all lowercase or all uppercase letters for the initial letter of each word of the name to make it easier to read.

At their simplest, script files are one or more Qshell commands that are executed in sequence. Each command should be typed on its own line. Lines that begin with # may be used for comments. The scripts do not need to be compiled, which is like UNIX and DOS but unlike CL.

You execute scripts from Qshell by typing their name. If an extension, for example, .sh, was used, then you must include this as well. You should either be in the directory containing the script, fully qualify its name, or use the PATH facility. You may also execute the script from an OS/400 command display by using QSH with its optional parameter as shown in an example later in this section.

The name of the script may be followed by one or more values that will be available to the script as parameters. The parameters should be separated by spaces. If the parameter values include spaces, then they should be enclosed in single quotation marks. The parameters may be used inside the script using the special variables $1 for the first parameter, $2 for the second, etc. This is similar to DOS, except that in DOS, the variables are %1, %2, etc.

You can define additional variables within the program. A variable is set simply by choosing a name and writing a line with that name followed by = and then by the desired value as shown in the following example. If the value contains spaces or certain special characters, then it should be in quotation marks.

```
DIRECTORY=/OrderManagement
```

This creates the variable DIRECTORY if necessary and set its value to /OrderManagement. There are no types for variables. They are all strings, although some commands may attempt to interpret the contents as numbers.

Names are case sensitive. The variable defined in the previous example is not the same as Directory or directory. It is typical to use names entirely in uppercase, but this is not required.

You may reference a variable by using $ followed by its name. This is replaced by its current contents. You can use this as a parameter to a command, part of a parameter, or even all or part of the command name. Consider this example:

```
cd $DIRECTORY/log
```

This changes the current directory to the subdirectory log of the directory whose name was in the variable DIRECTORY. This is similar to DOS, except that DOS would use %DIRECTORY%.

This is a typical example of use. It is common to need to refer to the directory of a product multiple times within a large script. To avoid repeating the name and making it difficult to change later, it is common to set the name once in a variable and then use that as required later in the script.

If one script (the parent) executes another (the child), then the value of the variables of the parent will not normally be available to the child. Also, if the child changes a variable, then the change will not be seen by the parent.

If the parent uses the **export** command to set a variable, then it will be available to the child.

```
export DIRECTORY=/OrderManagement
```

A child script that is run later would see the variable DIRECTORY with the value set by its parent. Nonetheless, if the child changes the variable, the parent will not see the change.

If the parent script executes the child script with the command . (dot), then the commands in the child script will behave as if they were executed by the parent. In this case, variables set in either way by the parent (export or not) are seen by the child and changes made by the child are seen by the parent.

A useful feature of Qshell commands is redirection of standard output. Standard output is the output from Qshell programs that is sent to the Qshell display. If you follow a command or script with a > (greater than) sign and a file name, then output that would go to standard output instead goes to that file. This can be useful for saving the output in a file as a log. It is also useful if you are executing your script from a CL program using the QSH command and you do not want the program to stop with a Qshell display. As mentioned earlier, if a command is supplied to the Qshell as a parameter and it produces no output, the display is not shown and the program proceeds without operator intervention. Redirecting output to a file is a way to arrange that no output is produced.

```
myscript.sh > redirect.out
```

A second output stream is called standard error. This is similar to standard output and is shown on the Qshell display by default. It is not affected by the redirection of standard output but can be redirected in a similar way by using the operator 2> followed by a file name. The ability to redirect error output separately can be useful. In testing, you may want to redirect only normal output so that any error output is obvious. In production, you may want the error output also to go to a log file. You may have errors if you attempt to redirect both standard and error output to the same file, but you can solve this by using the unusual operator 2>&1. This means that stream 2 (another name for standard error) should be merged with stream 1 (another name for standard output). Therefore, it is affected by the same redirection as standard output.

The following Qshell command runs the script `myscript.sh` and sends both standard output and standard error to the file myscript.out:

```
myscript.sh > redirect.out 2>&1
```

If the file used for redirection already exists, then it is overwritten. If you want to append to the file rather than to overwrite it, use `>>` or `2>>`. The following command is similar to the previous example, but appends the output to the file:

```
myscript.sh >> redirect.out 2>&1
```

If you want to run this script from a CL program or an OS/400 command line, then you can use the following command:

```
QSH CMD('/mydirectory/myscript.sh >> redirect.out 2>&1')
```

The script is qualified since it is usually easier in this context to do so rather than to set the current directory. An alternative is to use:

```
QSH CMD('cd /mydirectory ; myscript.sh >> redirect.out 2>&1')
```

This uses the QSH ability to take multiple commands separated by semi-colon (;). The output of **cd** is not redirected in this example. The redirection only refers to the second command. Redirection can be added to the **cd** command, but it does not produce standard output and only produces error output if the directory is invalid (for example, does not exist, no authority etc.).

6.3 Checking the QEJBSBS subsystem

WebSphere 3.5 jobs run in the QEJBSBS subsystem. You can use the WRKACTJOB command to view this subsystem as shown here:

```
WRKACTJOB SBS(QEJBSBS)
```

On the Work with Active Jobs display (Figure 6-5), for each WebSphere instance, there is a monitor and an administration job. There is one job for each application server that is stated in the instance. There may be further jobs that are not part of WebSphere but that are closely related to the application.

For the standard instance, the monitor job is called QEJBMNTR. It is the first job to start when the subsystem is started. The administration job is called QEJBADMIN. It is started by QEJBMNTR just after QEJBMNTR starts.

For any additional instances, there is another monitor and administration job. The names of these jobs depend on how the instance was created, but it is common to use a short name for the instance. An example is TEST followed by MNTR and ADMIN.

This example shows the standard instance and an additional instance called TEST running. Two applications are also running.

```
                          Work with Active Jobs                          HOMER
                                                            21/12/01  13:33:04
 CPU %:        .0     Elapsed time:    00:00:00     Active jobs:     331

 Type options, press Enter.
   2=Change    3=Hold    4=End    5=Work with    6=Release    7=Display message
   8=Work with spooled files    13=Disconnect ...

 Opt   Subsystem/Job   User        Type   CPU %   Function        Status
  ■     QEJBSBS         QSYS        SBS      .0                    DEQW
  __      QEJBADMIN     QEJB        BCI      .0    PGM-QEJBADMIN   JVAW
  __      QEJBMNTR      QEJB        BCH      .0    PGM-QEJBMNTR    EVTW
  __      SALESORDER    QEJB        BCI      .0    PGM-QEJBSVR     JVAW
  __      SALESTEST     QEJB        BCI      .0    PGM-QEJBSVR     JVAW
  __      TESTADMIN     QEJB        BCI      .0    PGM-QEJBADMIN   JVAW
  __      TESTMNTR      QEJB        BCH      .0    PGM-QEJBMNTR    EVTW


                                                                     Bottom
 Parameters or command
 ===>
 F3=Exit    F5=Refresh        F7=Find       F10=Restart statistics
 F11=Display elapsed data    F12=Cancel    F23=More options    F24=More keys
```

Figure 6-5   WRKACTJOB command showing the subsystem QEJBSBS

6.3.1 Instance monitor jobs

If you want to confirm that a particular job, QEJBMNTR for example, is an instance monitor and for which instance, look at its job log. To look at the job log, type 5 next to the job in the WRKACTJOB display and then select option 10. If it is a monitor job, you see a call to the QEJB/QEJBMNTR program. The second parameter to this call is the name of the admin.properties file of the instance. The instance name is normally the name of the directory between WebASAdv and properties. If you look at the job log of the default instance, you see the following command:

```
CALL PGM(QEJB/QEJBMNTR)
    PARM('-p' '/QIBM/UserData/WebASAdv/DEFAULT/properties/admin.properties')
```

You can press F10 for detailed messages. Then you see a message indicating that the administration job was started, using its full name. Here is an example from the monitor of the default instance:

```
WebSphere administration server job 033446/QEJB/QEJBADMIN started.
```

Occasionally you may see multiple messages indicating that the administration job was stopped and started multiple times. This is usually because of time out problems caused by poor performance. See "Constrained performance" on page 132 for more information.

6.3.2 Instance administration jobs

It is useful to look at the job log of an administration job, for example, QEJBADMIN. Type 5 next to it in the WRKACTJOB display and then select option 10.

On the Display Job Log display (Figure 6-6), near the top of the display, you see this message:

```
WebSphere server started with JDK 1.2.
```

By itself, this does not confirm that this is an administration job since it also appears in application server jobs. This message is typically followed by five messages, indicating that SQL server jobs are being used. Here is an example:

```
Job 031345/QUSER/QSQSRVR used for SQL server mode processing.
```

After these messages, you should see:

```
WebSphere administration server QEJBADMIN ready.
```

```
                              Display Job Log
                                               System:     HOMER
Job . . :     QEJBADMIN     User . . :     QEJB         Number . . . :     087923

■    Job 087923/QEJB/QEJBADMIN started on 20/12/01 at 16:15:07 in subsystem
       QEJBSBS in QEJB. Job entered system on 20/12/01 at 16:15:06.
     Output file RDBENTRIES created in library QTEMP.
     Member RDBENTRIES added to output file RDBENTRIES in library QTEMP.
     WebSphere server started with JDK 1.2.
     Job 086219/QUSER/QSQSRVR used for SQL server mode processing.
     Job 087106/QUSER/QSQSRVR used for SQL server mode processing.
     Job 086225/QUSER/QSQSRVR used for SQL server mode processing.
     Job 087107/QUSER/QSQSRVR used for SQL server mode processing.
     Job 086218/QUSER/QSQSRVR used for SQL server mode processing.
     WebSphere application server job 087924/QEJB/SALESORDER started.
     WebSphere administration server QEJBADMIN ready.

                                                                    Bottom
Press Enter to continue.

F3=Exit    F5=Refresh    F10=Display detailed messages    F12=Cancel
F16=Job menu              F24=More keys
```

Figure 6-6   Job log of the QEJBADMIN job

This confirms that this is an instance administration job and that it is ready. You should now be able to successfully connect a console to the instance. This message may be followed by some more messages about SQL server jobs.

If you are using multiple instances, then you may need to check the TCP/IP port that the instance is using. In the job log, position the cursor on the ready message and press F1 for help. You see a message like this example:

```
Message . . . . :    WebSphere administration server QEJBADMIN ready.
Cause . . . . . :    WebSphere administration server QEJBADMIN is ready to
  handle requests from the WebSphere administration console on port 900.
  Diagnostic information is controlled on port 36899.
```

Since this is a default instance and its configuration is not modified, the port for the console is the default 900.

If you are familiar with the configuration of your system and need to quickly check which instances are running, use the NETSTAT command. If you use the following command, you will see the status of TCP/IP sockets on your system:

```
netstat *cnn
```

On the Work with TCP/IP Connection Status display (Figure 6-7), page down until you see 900 or your expected port number in the Local Port column. The ports are in numeric sequence, but some well-known ports are replaced with the names of applications that typically use them. To see the numbers in all cases, press F14. If you find the port for one of your instances, look at the State column, in which you should see Listen. This is not proof that this is a WebSphere administration server and that it is ready. However, if you are familiar with your configuration and the system is generally reliable, then it is a good indication.

```
                    Work with TCP/IP Connection Status
                                              System:    HOMER
Type options, press Enter.
  3=Enable debug    4=End    5=Display details    6=Disable debug
  8=Display jobs

      Remote             Remote        Local
Opt   Address            Port          Port      Idle Time    State
  ▮   *                  *             drda      861:32:42    Listen
  _   *                  *             ddm       861:32:42    Listen
  _   *                  *             ddm-ssl   861:32:42    Listen
  _   *                  *             as-svrmap 000:04:47    Listen
  _   *                  *             ███       861:28:45    Listen
  _   *                  *             900       021:16:25    Listen
  _   *                  *             902       002:03:04    Listen
  _   *                  *             telnet- > 861:27:19    Listen
  _   *                  *             as-admi > 000:03:32    Listen
  _   *                  *             5002      222:18:33    *UDP
  _   *                  *             as-mgtc > 595:59:52    Listen
  _   *                  *             as-mgtc > 363:29:19    Listen
                                                             More...
F5=Refresh    F11=Display byte counts    F13=Sort by column
F14=Display port numbers    F22=Display entire field    F24=More keys
```

Figure 6-7   The NETSTAT command showing ports 900 and 902

If you have trouble connecting a console to the administration server, although the job says that it is ready, then it may be a name problem. If a console is started, then you can find the full name of the administration server job by selecting the node and looking for Process ID on the General page.

The job log of the administration server job has messages that indicate when individual application servers were started and stopped, for example:

```
WebSphere application server job 033456/QEJB/SALESORDER started.
```

If these messages do not correspond to an explicit start or stop request, then there may be a time out problem. For more information, see "Constrained performance" on page 132.

6.3.3 Specifying ports

The WebSphere console and some other IBM tools, for example the Resource Analyzer, have a special provision for alternative instances and ports. The console takes two parameters. The first parameter is the name of the WebSphere server, and the second parameter is the port. Both are optional. The first defaults to the current system and is never appropriate if you are running WebSphere on the iSeries. The second defaults to 900 and may be omitted for the standard instance.

If you are using an additional instance, you need to know the port that it is using and specify this as the second parameter to the console.

Many clients may lack this explicit support for alternative instances. However, it is still normally possible to attach the clients to an alternative instance. Using system:port in place of the system name works with most clients.

6.3.4 Application server jobs

For each application server that is started by any instance, there is a job running in the QEJBSBS subsystem. The name of the job is the name of the application if it is acceptable as an OS/400 job name. If the name is too long, then it is truncated and unusual characters may be omitted.

The name of the instance is not used as part of the job name. If you have the same application server name in multiple instances, it is not obvious which job belongs to which instance. To see the full name of the job (including its OS/400 job number), enter 5 next to it.

Then look in the job log of the administration job of the instance where you will see the full name of any application servers that it has started.

Alternatively, if you select the application server in the console and look on the General page, you will see the process ID, which is the full ID of the job. However, the simplest method is to choose application server names that give easily recognizable and unique job names, for example, prefix applications within the Test instance with Test.

Looking for the application server job in the QEJBSBS subsystem is a quick check of whether the application is running. If the job is missing, then the application is certainly not running. If the job is running, then the application may be is running, but there is a chance that there is a problem with it. The job log of these jobs is rarely useful. For further information about the application, you need to start the console.

In the previous above, you can guess that SALESORDER is an application within the default instance and SALESTEST is an application within the TEST instance. You can confirm these guesses by looking at the job messages of the administration jobs or by using the console for each instance.

Occasionally, an application server job may remain running, although the console indicates that the application is stopped. In this case, the application is unlikely to start successfully unless this job is ended.

6.4 WebSphere versions on the iSeries, your console system, and clients

It is important that the exact release of WebSphere on the iSeries, the PC that you use for the console, and any client PCs, are the same. For example, if you loaded Fix Pack 3.5.4 on the iSeries, then you should also load it onto your console system and update the WebSphere files (for example, ujc.jar and ejs.jar) on your client systems. A mismatch in either direction, server above client or server below client, may cause a problem.

In some cases of a mismatch, the console and applications may work correctly but this is rare. In other cases, the console and applications will work to some extent but not completely. Even if your application appears to work despite a mismatch, it is not advisable to do so.

A common symptom of a release mismatch between the WebSphere release on the iSeries and your console PC is that the console window appears and the message `Loading …` appears but the `Console Ready` message never appears. Another symptom is that simple operations, such as starting and stopping applications, work but more complex ones, such as creating an EJB, do not.

6.4.1 Checking the WebSphere PTF level on the iSeries

There are two ways to check the PTF level of WebSphere on the iSeries.

The first way (usual way) is to check OS/400 group PTFs. The group PTF for WebSphere 3.5 on V5R1 is SF99147, and on V4R5 it is SF99138. In all cases, including V4R4, the data area that indicates the PTF level is in the QEJB library. You can find it by using the following command:

```
WRKDTAARA QEJB/SF*
```

Then you see the Work with Data Areas display (Figure 6-8). If you do not see a data area, then you have not applied any group PTFs and your WebSphere level is 3.5.0. We recommend that you do an upgrade since many important fixes have been issued since then.

```
                          Work with Data Areas

Type options, press Enter.
   1=Create    2=Change    4=Delete     5=Display     13=Change description

        Data
Opt    Area            Library      Text
 ▄
 ___   SF99147        QEJB



                                                                  Bottom
Parameters for options 1, 2 and 5 or command
===>
F3=Exit        F4=Prompt     F5=Refresh    F9=Retrieve    F11=Display names only
F12=Cancel     F16=Repeat position to    F17=Position to
```

Figure 6-8   The WRKDTAARA command showing the SF99147 data area

If you see a data area, type 5 next to it to display its contents. The Display Data Area display in Figure 6-9 shows a typical result.

```
                           Display Data Area
                                                   System:     HOMER
Data area . . . . . . . . :     SF99147
   Library . . . . . . . :       QEJB
Type  . . . . . . . . . . :     *CHAR
Length  . . . . . . . . . :     50
Text  . . . . . . . . . . :

           Value
Offset     *...+....1....+....2....+....3....+....4....+....5
   0        'Group PTF#: SF99147 04   V5R1M0 07/04/2001 3.5.4   '




                                                                  Bottom
Press Enter to continue.

F3=Exit    F12=Cancel
```

Figure 6-9   Option 5 within the WRKDTAARA command showing data area contents

The PTF level is indicated in two ways:

► The number after the PTF ID
► 3.5.x number at the end

Also, verify that the OS/400 release level (V5R1M0 in this case) matches your current release level. If it does not, then this data area remains from before an OS/400 upgrade.

Another way to check the PTF level is to view the stream file /QIBM/ProdData/WebASAdv/properties/com/ibm/WebSphere/product.xml. You can view this in any of the ways described earlier. The contents are similar to those of the equivalent Windows NT file.

6.4.2 Checking the WebSphere PTF level on a PC

The simplest way to check the PTF level on the console system is to start the console and select the About option on the Help menu. This clearly shows the WebSphere version for example, 3.5.4.

An alternative way is to view the product.xml file. If you used the default directory for your WebSphere installation, then this is C:\WebSphere\AppServer\properties\com\ibm\WebSphere\product.xml. However, the C: drive and the initial directories \WebSphere\AppServer may be overridden while installing WebSphere on your PC. If you know how to start the console, then check from where the console is running. The default location of the console command is C:\WebSphere\AppServer\bin\adminclient.bat. The prefix of the directory is the same. This file is an XML file, but even if you do not understand XML, it is easy to read. You should see the version quite clearly on a line as shown in Figure 6-10.
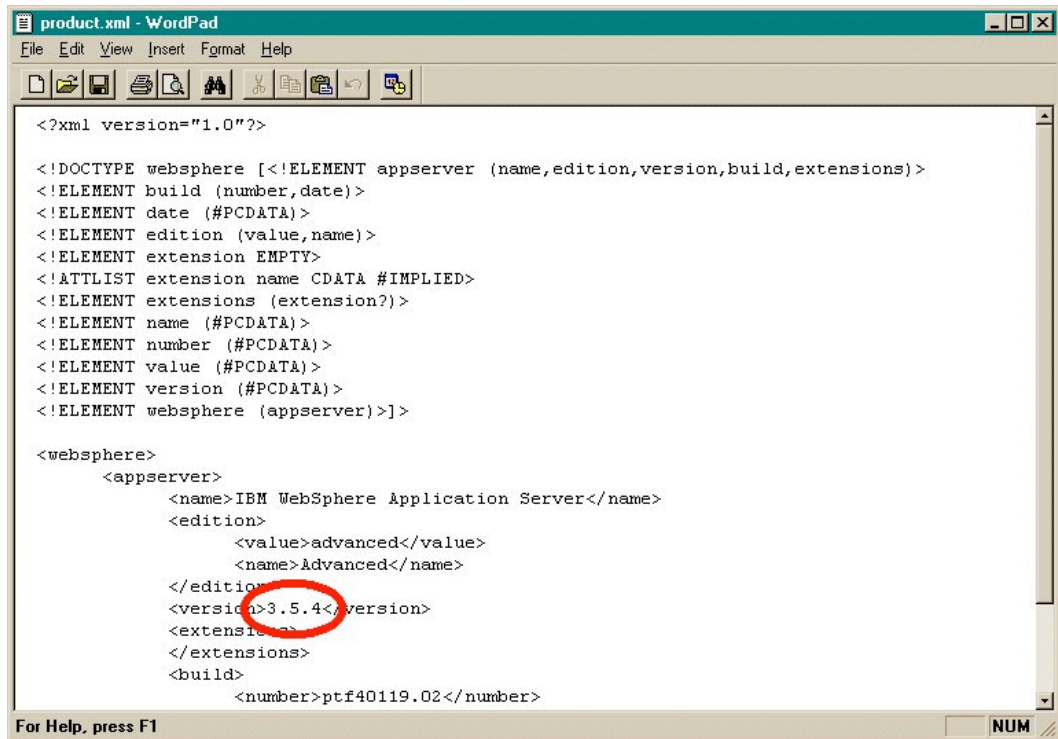
Figure 6-10   WordPad editor showing the Product.xml file

Note that this file was opened with WordPad rather than Notepad. This is because it contains UNIX-style line feed only between lines, Notepad does not cope with.

6.5 Common problems with commerce.connect on iSeries

This section covers some common problems when running commerce.connect application on the iSeries server.

6.5.1 Problems connecting the console to WebSphere on the iSeries

If you see a message saying that the Admin Client failed to connect to the Admin Server, then you should perform the checks that are explained here.

Check that WebSphere is started on the iSeries. Check whether the instance administration job is running, that it is ready, and on which port it is listening.

Check that you specified the correct system name and port when you started the console. If you start the console from the command line, then it takes two parameters. The first is the system name, and the second is the port.

If you start the console using a menu item, desktop icon, or batch file, then look into the definition of the item to ensure that the system name and port are correct. As a diagnostic step, you can try to run the console from a Windows command prompt:

1. Start a command prompt.

2. Switch to the drive and directory in which you installed WebSphere. By default, this is C:\WebSphere\AppServer, but you may have overridden it during the WebSphere installation.

3. Switch to the `bin` directory within this directory.

4. Use the **adminclient** command followed by the system name and the port.

If this still fails, check the system name more carefully as explained in the following section.

You should also check if the version of WebSphere on your console system and the iSeries match. However, if the system name and port are correct but the version is wrong, the console normally fails at a later point.

6.5.2 Checking the iSeries name as required by WebSphere

WebSphere uses the TCP/IP name of the system. Typically this is the same as the normal system name (as shown by the sign-on screen), but it can be different. The TCP/IP name is the one you would use for commands such as PING, TELNET, FTP, etc. Check whether you can ping the iSeries from the console PC. If you cannot, then you need to check your network and maybe your TCP/IP configuration. If you are sure of the TCP/IP name of your iSeries then continue.

WebSphere adds an extra complication. It treats the system name as case sensitive, although TCP/IP does not. Therefore, even if you can ping the iSeries successfully, you may not be using the correct name for WebSphere. Usually the WebSphere name is entirely lowercase, but occasionally it is entirely uppercase.

The easiest way to check the system name as required by WebSphere is to view one of its configuration tables. The configuration tables for the default instance are in the EJSADMIN library. For other instances, the library name was chosen when the instance was created. A typical name is TESTADMIN. The following example assumes the default instance and that the SQL tool in Operations Navigator is being used. If you are using an alternative instance, then you need to amend the library name. If you are using interactive SQL (for example, the STRSQL command) or you changed your naming convention in the Operations Navigator tool, then you have to change the qualifier separator.

Sign on as `QSECOFR` and run this SQL statement:

```
select NAME
      from EJSADMIN.NODE_TABLE
```

This shows the node name as expected by WebSphere. Do not change this value. This does not cause WebSphere to accept a different system name and may cause problems that are more serious. Do not make any other changes to this table or any other table in the WebSphere libraries.

6.5.3 Checking the iSeries database name

In a few contexts, the name of the DB2 database on the iSeries server is required rather than the system name. Two such contexts are creating a datasource and specifying the database for stateful connections.

The database name is normally the same as the system name, but it can be different. It is even possible that it is not set at all. If the database name was not set, then SQL software may fail, but RPG or Cobol software that did not use SQL would still work.

To check the database name, use the following command:

```
WRKRDBDIRE
```

In the Work with Relational Database Directory Entries display (Figure 6-11), look for a line with *LOCAL in the Remote Location column. The entry in the Relational Database column of this line is the name of the database on the system. There may be other lines with entries other than *LOCAL in the location column. You can ignore these for a simple setup.

```
                    Work with Relational Database Directory Entries

Position to  . . . . . . .  _____

Type options, press Enter.
  1=Add    2=Change   4=Remove   5=Display details   6=Print details

            Relational             Remote
Option      Database               Location                 Text
    █       HOMER                  *LOCAL
    _       NEEDJAVA               needjava
    _       PLATO                  plato






                                                                  Bottom
F3=Exit    F5=Refresh    F6=Print list    F12=Cancel
(C) COPYRIGHT IBM CORP. 1980, 2000.
```
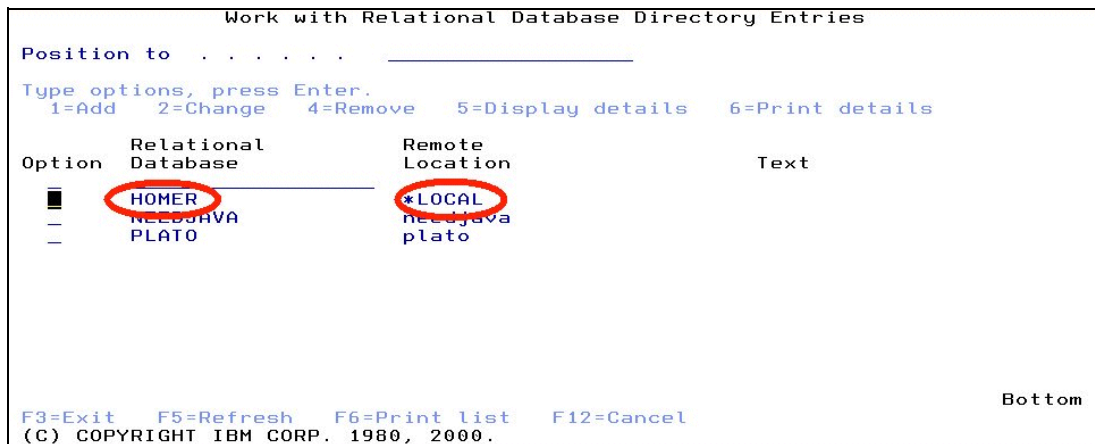
Figure 6-11   The WRKRDBDIRE command file

If there is no line with *LOCAL in the location column, then you need to add one. Type 1 in the Option column of the first line and your desired database name in the name column (we recommend that you name the database the same as the system itself), and press Enter. On the next display, enter the remote location as *LOCAL and leave the other fields alone. Press Enter and the database should now be setup correctly.

> **Attention:** Verify the local database before you install WebSphere.

If you are running WebSphere on the same system as the database, then you can usually enter the database name as *LOCAL to avoid problems with database naming. This may be done in a datasource and in the configuration of stateful connections.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 161.

▶ *Geac System21 Implementation for AS/400,* SG24-5967

Other resources

These publications are also relevant as further information sources:

▶ *OS/400 Work Management,* SC41-5306

▶ *Backup and Recovery,* SC41-5304

The following publications are available through Geac:

▶ *Geac call.connect Installation Guide*

▶ *Geac System21 Installation and Setup Guide*

▶ *Geac System21 Administration Functions Active Enterprise Framework*

▶ *System21 Setup for call.connect*

▶ *Basic System21 Setup for call.connect*

▶ *Geac System21 Installation Guide*

Referenced Web sites

The Geac Web site is also relevant as a further information source:

http://www.geac.com/

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

Symbols
\# 150
\$ 147, 151
\$ prompt 147
\$1 150
\$2 150
*BASE 115
*INTERACT 115
*LOCAL 160
*MACHINE 115
*SHRPOOL1 115
*SHRPOOL2 115
. (dot) 151
.connect applications 1
> 151
@ 142
@ctive Modeler 15, 73
@ctive Processes 15
¢ 147
£ 147

Numerics
100% Pure Java 18
2> 151
2>&1 151
5769-DG1 21
5769-JV1 21
5769-SS1 21
5769-ST1 22
5769-TC1 21

A
active selling 3
Additional number of jobs parameter 111
adminclient 159
administration server 77
Advanced Pricing module 122
Advanced Shipping Notification (ASN) 5
AIF .ini files 92
Apache 39, 41, 79
application settings 132
architectural representation 16
architecture
    concept 8
    goals and constraints 16
    non-functional considerations 16
ASCII 138
ASCII CCSID 142
ASN (Advanced Shipping Notification) 5
ASP for journaling 28
asynchronous messaging 12
authentication services 9
authorization rule 101

automatic performance adjustment 121

B
base memory pool 115
bin 139
blanket order delivery schedule 5
blanket purchase order 5

C
C 18
C++ 18
Cached data 54
cached data 54
call center 2
call management 3
call.connect 12
    Advanced Pricing module 122
    installation 19
    introduction 2
    non-standard installation 56
    roles 2
    typical screen 2
call.connect library list 36
callConnect.bat 67
CallConnect.jnlp 51
    configuring 70
case sensitive 159
cat 149
CCSID 142
CCSID 65535 30
cd command 148
CIW (Compute Intensive Workload) 109
CL 18
clustering 9
commerce.connect 1
commerce.platform 1, 73
Commercial Processing Workload (CPW) 108, 132
Compute Intensive Workload (CIW) 109
Config.xml 25
configuration files 138
configuring HTTP Server 40, 80
configuring the LDAP server 94
connection time out 135
ConnectorManager.xml 92
console 155
constrained performance 132
Content Negotiation 50, 69
controller 91
CPW (Commercial Processing Workload) 108, 132
Create Java Program (CRTJVAPGM) command 127
creating a new instance 77
creating a queue 33
creating a queue manager
    vendor.connect 88

IBM

Redbooks

Geac System21 commerce.connect Implementation on the IBM @server iSeries Server

(0.2"spine)
0.17"<->0.473"
90<->249 pages

Geac System21 commerce.connect

Implementation on the IBM *e*server iSeries Server

IBM®

Redbooks

Gain useful insights through the product overview

Follow the step-by-step guide to install commerce.connect

Learn helpful usage and performance tips and techniques

This IBM Redbook introduces the new Geac commerce platform *.connect* applications – the *call.connect* and *vendor.connect* applications. These applications extend and enhance the capabilities of Geac System21 into the intranet and Internet.

This redbook targets consultants or customers who work with these .connect applications. It explains how to install, maintain, integrate, and manage these applications on the IBM *e*server iSeries server. It also helps you to understand the architecture and middleware used by the applications.

Prior to reading this book, you must be familiar with the basic, traditional use of the iSeries or AS/400 and System21. For example, you should know how to enter simple commands and understand such concepts as the library list. Similarly for System21, you should be familiar with the menus and such tasks as defining a System21 user.

As necessary throughout the book, detail is provided about the newer, less traditional features of the iSeries such as the integrated files system (IFS), Qshell, Java, and WebSphere.