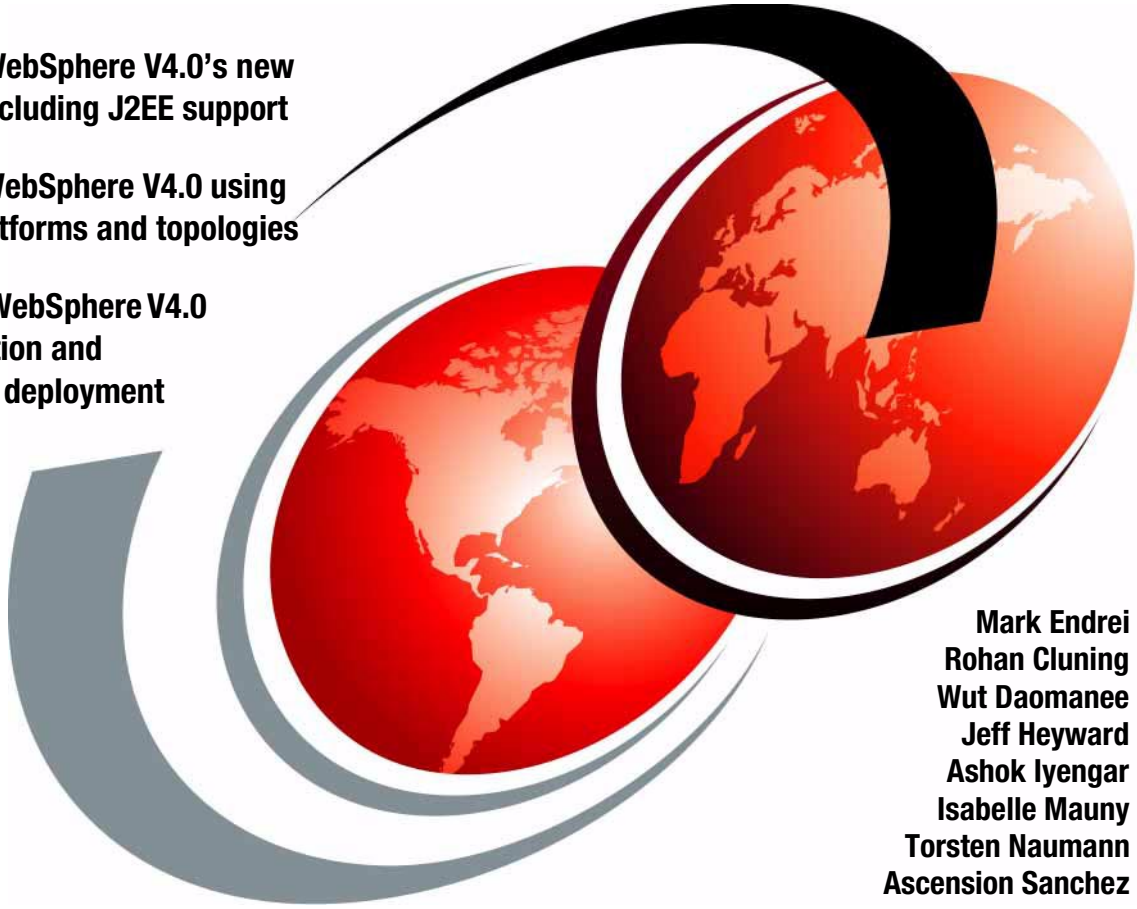
WebSphere software

IBM

IBM WebSphere V4.0 Advanced Edition Handbook

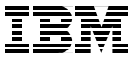
- Exploring WebSphere V4.0's new features, including J2EE support
- Installing WebSphere V4.0 using popular platforms and topologies
- Mastering WebSphere V4.0 administration and application deployment



Mark Endrei
Rohan Cluning
Wut Daomanee
Jeff Heyward
Ashok Iyengar
Isabelle Mauny
Torsten Naumann
Ascension Sanchez

Redbooks

ibm.com/redbooks



International Technical Support Organization

IBM WebSphere V4.0 Advanced Edition Handbook

March 2002

SG24-6176-00

Take Note! Before using this information and the product it supports, be sure to read the general information in “Special notices” on page 1089.

First Edition (March 2002)

This edition applies to IBM WebSphere Application Server Version 4.0.1, Advanced Edition for use with Windows 2000, IBM AIX 4.3.3, Sun Solaris 2.7, and Red Hat Linux 7.1.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Preface

This redbook provides system administrators, developers, and architects with the knowledge needed to implement WebSphere Application Server V4.0, Advanced Edition runtime environment, to package and deploy Web applications, and to perform ongoing management of the WebSphere environment.

Part 1 of the redbook introduces WebSphere Application Server V4.0 and J2EE enterprise applications.

Part 2 looks inside WebSphere V4.0 at architecture and topology alternatives, Web services, and J2EE security.

Part 3 explains how to plan your WebSphere V4.0 installation. It steps you through the installation process on Windows, AIX, Solaris, and Linux platforms for commonly used topologies.

Part 4 guides you through WebSphere administrative console configuration tasks. It also provides examples of how to package and deploy your J2EE enterprise applications.

Part 5 covers other administrative tasks, including using administrative tools, monitoring and tuning your environment, command-line administration, troubleshooting, and migration.

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, Raleigh Center.

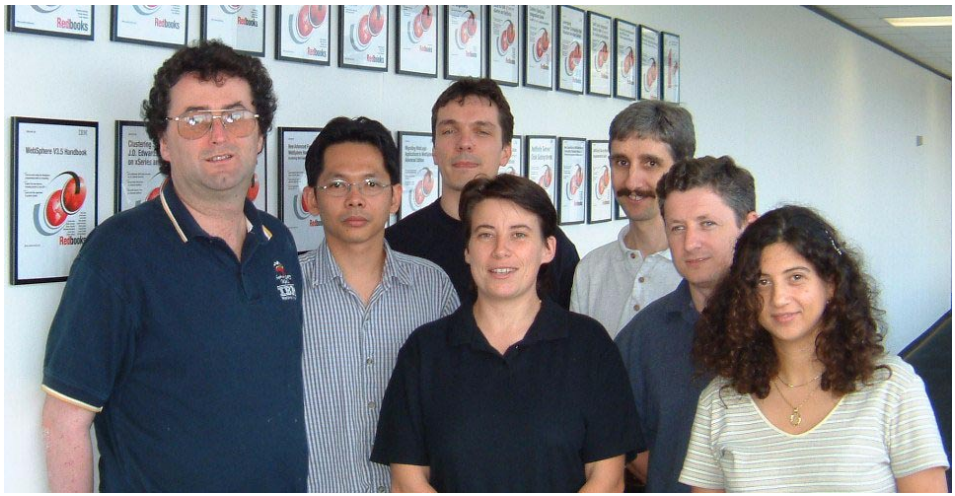


Figure 0-1 The IBM redbook team (Left to right: Rohan Cluning, Wut Daomanee, Torsten Naumann, Isabelle Mauny, Mark Endrei, Jeff Heyward, Ascension Sanchez, Not present: Ashok Iyengar

Mark Endrei is an IT Architect at the International Technical Support Organization, Raleigh Center. He writes on all areas of WebSphere. Before joining the ITSO early in 2001, Mark worked in IBM Global Services Australia as an IT Architect. He holds a bachelor's degree in Computer Systems Engineering from Royal Melbourne Institute of Technology, and an MBA (Technology Management) from Deakin University/APESMA.

Rohan Cluning is an IT Architect in IBM Global Services Australia. He has 15 years of experience in application design and development. He holds a bachelor's degree in Computer Science/Instrumental Science from Swinburne Institute of Technology. His area of expertise is the design and development of large systems using object-oriented technologies. In the past two years he has worked extensively with WebSphere and VisualAge for Java to design and build e-business solutions for a telecommunications company.

Wut Daomanee is a RS/6000 IT Specialist in IBM Thailand. He has five years of experience in RS/6000, AIX and Web technology. He holds a degree in Computer Engineering from the Prince of Songkla University (Thailand). His areas of expertise include WebSphere family products and e-business solutions.

Jeff Heyward is an Advisory IT Specialist in IBM Global Services Australia. He has eight years of experience in the computing field. He holds a degree in Science (Honours) from the University of Melbourne, Australia. His areas of expertise include Java, CORBA, object-oriented design, and WebSphere.

Ashok Iyengar is an Advisory Software Engineer in the USA. He has 18 years of experience in software design/development. He holds a Masters degree in Computer Science from North Dakota State University, Fargo. His current areas of expertise include consulting on the WebSphere platform, particularly on WebSphere Application Server and WebSphere Portal Server. He also co-authored the *WebSphere V3.5 Handbook,* SG24-6161.

Isabelle Mauny is a consultant for the EMEA Technical Sales team in La Gaude (France). She has eight years of experience in application development and object-oriented technologies. She has been working extensively with WebSphere Application Server for the last two years, and VisualAge for Java for the last three years, helping customers to design and implement J2EE applications. She is the co-author of *Effective VisualAge for Java 3.5*, published in early 2001.

Torsten Naumann is a Senior Systems Engineer for IT Services and Solutions (an IBM Global Services company) in Germany. He has eight years of experience in the IT field, specializing in WebSphere products for the last 1.5 years. He holds a degree in Computer Science from Chemnitz University of Technology. His areas of expertise include consulting, coaching, performance tuning, problem determination and support. He has written several articles for professional IT journals/magazines in Germany.

Ascension Sanchez is a Software Support Specialist in the e-business Support Center in Basingstoke, United Kingdom. She has three years IT technical support experience with IBM. She holds a degree in Electronics and Electrical Engineering from the Universidad Politecnica de Madrid (Spain) and a Masters Degree in Engineering from the University of Birmingham (United Kingdom). Her main areas of expertise include AIX, communications software for the RS/6000 and WebSphere family products for UNIX and non-UNIX platforms.

Thanks to the following people for their contributions to this project:

Phil Johnson, IBM Raleigh

Tom Alcott, IBM Costa Mesa
Jeff Baker, IBM Raleigh
Joe Bockhold, IBM Rochester
Gail Christensen, IBM ITSO Raleigh
David Colasurdo, IBM Raleigh
Roger Cundiff, IBM Austin
Tim Deboer, IBM Toronto
Dana Duffield, IBM Rochester
Curtis Ebbs, IBM Raleigh
Lizet Ernand, IBM Austin
Srinivas Hasti, IBM Raleigh
Eric Jenney, IBM Rochester

Ken Klingensmith, IBM Costa Mesa
Peter Kovari, IBM ITSO Raleigh
Gabe Montero, IBM Raleigh
Bill Moore, IBM ITSO Raleigh
Michael Morton, IBM Raleigh
Nataraj Nagaratnam, IBM Raleigh
Russ Newcombe, IBM Austin
Paul Pacholski, IBM Toronto
Ajay Reddy, IBM Austin
Ian Robinson, IBM Tucson
Hany Salem, IBM Austin
Keith Smith, IBM Raleigh
Kevin Sutter, IBM Rochester
Ken Ueno, IBM Raleigh
Kevin Vaughan, IBM Raleigh
Subodh Vinchurkar, IBM Raleigh
Leigh Williamson, IBM Austin
Tom Zhou, IBM Austin

Special notice

This publication is intended to help system administrators, developers, and architects to implement and manage the WebSphere Application Server V4.0, Advanced Edition runtime environment. The information in this publication is not intended as the specification of any programming interfaces that are provided by WebSphere Application Server V4.0, Advanced Edition. See the PUBLICATIONS section of the IBM Programming Announcement for WebSphere Application Server V4.0, Advanced Edition for more information about what publications are considered to be product documentation.

IBM trademarks

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

e (logo)® 
AFP™
AFS®
AIX®
AlphaWorks®
CICS®
Database 2™
DB2®
DB2 Universal Database™
EtherJet™
Everyplace™
Home Director™
Hummingbird®
IBM®
IMS™
Informix™
iSeries™
MQSeries®
OS/390®
OS/400®
PC 300®
Redbooks Logo  ™
Redbooks™
RS/6000®
S/390®
SecureWay®
SP™
TXSeries™
VisualAge®
WebSphere®
z/OS™
zSeries™
Lotus®
Domino™

Comments welcome

Your comments are important to us!

We want our IBM 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 the address on page ii.

Part 1

Getting started

In this part we introduce WebSphere Application Server V4.0 and J2EE enterprise applications.

1

Introduction to WebSphere Application Server V4.0

The release of IBM WebSphere Application Server V4.0 represents a move to a single application server family that scales seamlessly. Businesses can respond to changing markets without migrating to a different technology base or replacing existing technology investments. With WebSphere Application Server V4.0, you can move applications to more capable platforms, or simply add to your existing infrastructure.

WebSphere Application Server V4.0 is leading the way in support for industry open standards. WebSphere Application Server V4.0 provides full Java 2 Platform, Enterprise Edition (J2EE) compliance with a rich set of enterprise Java open standards implementations. It also provides built in support for the key Web services open standards, making it production-ready for the deployment of enterprise Web services solutions.

In this redbook we provide a detailed exploration of the WebSphere Application Server V4.0, Advanced Edition (AE) runtime environment. We will not address the developer (AEd), single server (AEs), or Enterprise Extensions (EE) configurations of WebSphere Application Server V4.0. Readers interested in WebSphere V4.0 application development may also want to look at the *WebSphere Version 4 Application Development Handbook*, SG24-6134.

1.1 IBM WebSphere software platform for e-business

The IBM WebSphere software platform for e-business is a comprehensive set of integrated e-business solutions. It is based on industry standards that make it flexible and pluggable, which can enable you to adapt as markets shift and business goals change. Building on this robust platform, you can integrate diverse IT environments to maximize existing investments. You can deliver core business applications to the Web, and scale these applications to meet changing needs and increasing demand.

For more information about the full line of WebSphere software platform products and solutions, visit:

http://www.ibm.com/websphere

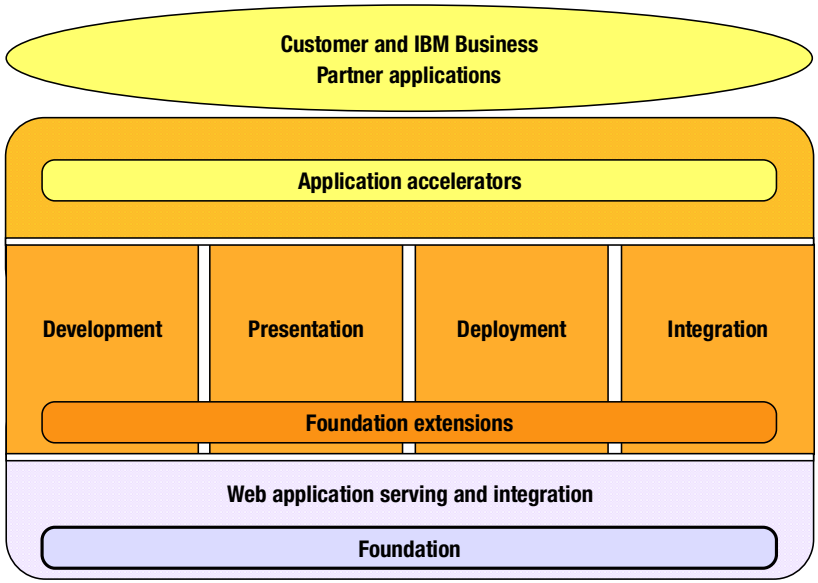


Figure 1-1 IBM WebSphere software platform for e-business

1.1.1 Foundation

The WebSphere software platform for e-business starts with a foundation formed from Web application serving and integration.

WebSphere Application Servers

WebSphere Application Server V4.0 allows you to quickly, reliably and flexibly Web-enable your business. It provides the core software to deploy, integrate and manage your e-business applications.

WebSphere Application Server supports custom-built applications, based on integrated WebSphere software platform products, or on other third-party products. Such applications can range from dynamic Web presentations to sophisticated transaction processing systems.

MQSeries

MQSeries middleware provides reliable and asynchronous messaging for more than 35 platforms. It lets diverse applications communicate securely and reliably between a range of platforms, so you can reuse and expand upon your existing business logic.

1.1.2 Foundation Extensions

WebSphere Foundation Extensions can help you reach new channels and improve access to your Web applications. They also include Web content management tools to help keep all your Internet-based information current.

Presentation

WebSphere Everyplace Suite

- Extends your e-business applications across new communication channels
- Delivers existing content to new devices
- Adds support for new technologies as they are developed

WebSphere Portal Server

- Builds your own custom portal Web site that serves the needs of employees, business partners, and customers
- Your users can sign on to the portal and receive a personalized Web page with access to the information and Web applications they need

WebSphere Personalization Server

- Builds a Web site, intranet, or extranet that delivers Web pages customized for each site visitor

WebSphere Transcoding Publisher

- Bridges data across multiple formats, markup languages, and devices
- Adapts, reformats, and filters content to make it suitable for pervasive computing
- Gives companies better access to customers, business partners, and mobile employees on various devices

WebSphere Voice Server

- Quickly develops and deploys voice-enabled e-business solutions
- Expands the use of Web applications to customer who only have phone access

Deployment

Other Foundation Extensions can dramatically improve your ability to handle high volumes of traffic for your Web site, while providing high availability and fast response times.

WebSphere Site Analyzer provides analysis for:

- Enterprise Web site visitor trends, usage, and content
- WebSphere Commerce Suite reporting

WebSphere Edge Server provides an integrated solution for:

- Local- and wide-area load balancing
- Content-based quality of service routing
- Web content filtering and caching for multi-vendor Web server environment

1.2 WebSphere Application Server V4.0 packaging

The introduction of WebSphere Application Server V4.0 represents a move to a single code base that is supported by virtually all major platforms. The flexible and scalable configurations available with this version allow you to respond to the changing marketplace, without migrating to a different technology base.

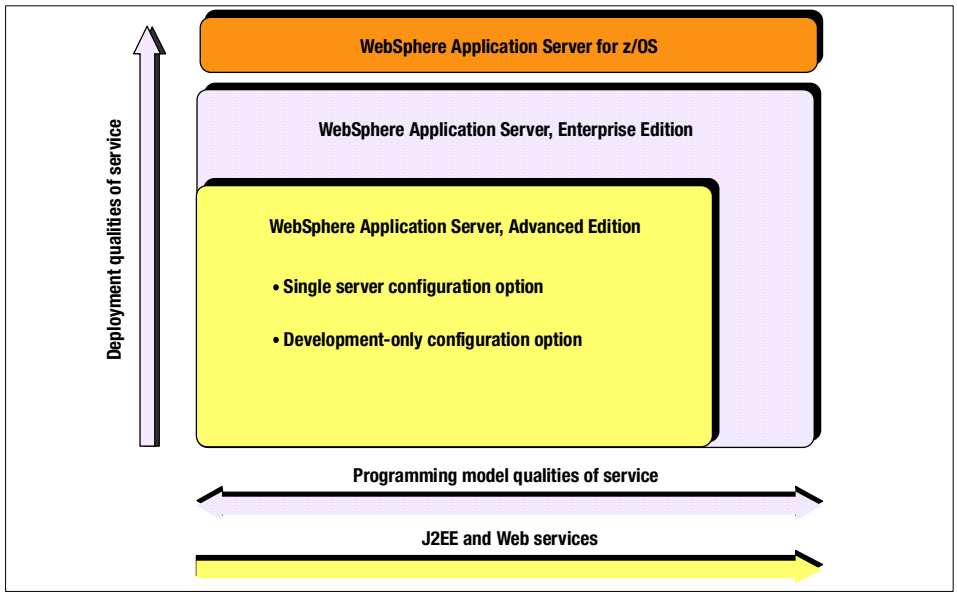


Figure 1-2 IBM WebSphere Application Server V4.0 Editions

With WebSphere Application Server V4.0, Advanced Edition three different configurations are available:

- The **full configuration (AE)** provides application server functionality with strong integration to databases, message-oriented middleware, and legacy systems and applications, along with clustering support. This configuration appeals to businesses that need to build highly transactional, manageable, available, and scalable applications that offer distributed security and remote administration.
- The **Single Server configuration (AEs)** provides application server functionality within a single runtime process. This configuration appeals to businesses that need to build stand-alone or departmental applications that are transaction- or message-oriented, and that don't require failure bypass, workload management, or remote administration.
- The **Developer license (AEd)** provides application server functionality to developers who need an easy-to-use environment for building and testing e-business applications. It appeals to developers who are looking for a friendly and powerful unit testing environment, especially one that is seamlessly integrated with IBM's tooling.

WebSphere Application Server V4.0, Enterprise Extensions (EE) extends WebSphere Application Server V4.0, Advanced Edition. It includes IBM TXSeries technology to meet the most sophisticated needs of rapidly evolving, highly distributed e-business infrastructures. WebSphere Application Server V4.0, Enterprise Extensions extends the Java programming model and provides additional qualities of service.

WebSphere Application Server V4.0 for IBM z/OS and IBM OS/390 fully exploits the IBM zSeries and IBM S/390 architecture to achieve superior levels of scalability, performance, security, and availability. Running WebSphere Application Server V4.0 on the z/OS platform provides the same enterprise services as that offered on distributed platforms with premier qualities of service. WebSphere Application Server V4.0 for z/OS and OS/390 exploits the workload manager that supports LPAR clustering technologies and is fully IBM Parallel Sysplex technology-enabled.

1.3 WebSphere V4.0 Advanced Edition features

The WebSphere Application Server V4.0, Advanced Edition provides the following major functionality:

- Full Java 2 Enterprise Edition (J2EE) platform certification.
- Tools for developing active Web sites through the use of Java servlets and JavaServer Pages (JSP). This functionality is available in all versions of the Advanced Edition.
- Tight integration with tools for developing and deploying enterprise beans written to the EJB specification. Enterprise beans can act as a bridge between your Web site and your non-Web computer systems.
- A set of application programming interfaces (APIs) for generating, validating, parsing, and presenting extensible markup language (XML) documents. This functionality is available in all versions of the Advanced Edition.
- Integrated support for key Web services open standards, which is production-ready for the deployment of enterprise Web service solutions for interoperability and business-to-business applications.
- A graphical user interface (GUI), the WebSphere Administrative Console, for administering the components of the Advanced Edition environment.
- Unparalleled connectivity provided by a preview implementation of Java 2 Connectivity (J2C).

1.3.1 WebSphere V4.0 Advanced Edition environment

Figure 1-3 shows the components that make up the WebSphere Application Server V4.0, Advanced Edition.

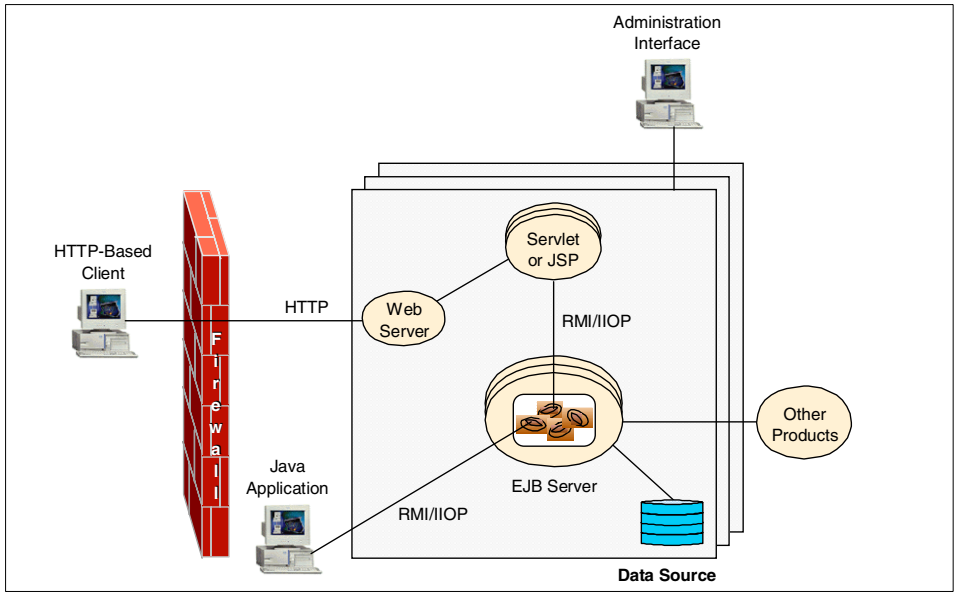


Figure 1-3 Components of the WebSphere Application Server V4.0, Advanced Edition

They can be combined to create a powerful, Java-centered, three-tiered system that puts heavy emphasis on a customer's Web site. Each part of WebSphere V4.0 Advanced Edition is explained as follows.

Administration server

The administration server and the administrative interface enable application servers and processes to be monitored and controlled centrally.

Browser-based clients

Clients of applications that run on the WebSphere V4.0 generally run in Java-enabled browsers. They send and receive information from a Web server by using HTTP. Browser-based clients can include applets and JavaServer Pages (JSP).

Web servers

Except for stand-alone Java applets, which are restricted by built-in Java security, browser-based client applications require that a Web server be installed on at least one machine in your WebSphere Application Server environment. The Web server provides the communications link between browser-based applications and the other components of WebSphere Application Server.

WebSphere Application Server supports many of the most widely used Web servers. The IBM HTTP Server, which is a modified version of the Apache server, comes with the WebSphere Application Server V4.0, Advanced Edition.

Web container

WebSphere contains a Java-based servlet engine that is independent of both your Web server and its underlying operating system. Java servlets extend the Web server's capabilities by creating a framework for providing request and response services over the Web.

Enterprise beans

WebSphere provides full support for enterprise beans. An enterprise bean is a Java component that can be combined with other enterprise beans and other Java components to create a distributed, three-tiered application.

1.3.2 WebSphere application model

WebSphere applications consist of object-oriented business logic that uses relational database systems for data storage. Applications are usually integrated with Web clients (either thick or thin); they can also be integrated with existing procedural applications running in application servers.

An application consists of the following components, each performing a different function:

- HTML and JSP pages provide the user interface and program flow.
- Enterprise beans contain the application's business logic and handle transactional operations and access to databases.
- Servlets coordinate work between the other components of the application. They also can dynamically generate Web page contents.
- JavaBeans components enable the other types of components to work together.
- Relational databases implement persistence and query functions for enterprise beans. Either new or existing databases can be used in an application.

1.4 WebSphere development environment

WebSphere Studio is the WebSphere Application Server application development environment. It can be used to create everything from personal Web pages to Web sites that serve as front ends for e-business applications. WebSphere Studio provides a tool suite for developing HTML content and can be integrated with other content development tools.

VisualAge for Java is an integrated development environment that supports the complete cycle of Java program development. Although it is not formally a part of the WebSphere Application Server V4.0, Advanced Edition, VisualAge for Java is tightly integrated with the WebSphere Application Server environment. This integration enables VisualAge developers to develop, deploy, and test their Java programs without leaving the VisualAge program. It also helps developers to manage the complexity of the enterprise environment and is capable of automating routine steps.

See the *WebSphere Version 4 Application Development Handbook*, SG24-6134 for a complete guide to WebSphere application development.

1.5 Structure of this handbook

Part 1, “Getting started” on page 1

Chapter 2, “What’s new in WebSphere V4.0” on page 15 provides a brief description of the new and improved features of IBM WebSphere Application Server V4.0.

Chapter 3, “The Java 2 platform” on page 29 gives a brief introduction to the various components of a J2EE enterprise application. An understanding of these concepts will help you to work more effectively with WebSphere Application Server V4.0.

Part 2, “Inside WebSphere” on page 49

Chapter 4, “WebSphere architecture overview” on page 51 looks at the major components within IBM WebSphere Application Server V4.0, such as the application server, Web container, and EJB container, and how they map to the J2EE component architecture.

Chapter 5, “Topologies selection” on page 67 describes some of the topologies that IBM WebSphere Application Server V4.0 supports. It covers options regarding the use of clones, domains, scaling, and tiering.

Chapter 6, “Web services” on page 85 introduces Web services. It provides an overview of Web services architecture and how Web services are supported by WebSphere V4.0.

Chapter 7, “Security” on page 103 gives an introduction to the J2EE security model and describes how this security model is supported by WebSphere Application Server V4.0.

Part 3, “Installing WebSphere” on page 125

Chapter 8, “Installation approach” on page 127 provides an explanation of the procedures for installing, configuring and verifying WebSphere Application Server V4.0. It provides a summary of the decisions that must be made when planning an installation to match a particular topology or architecture.

Chapter 9, “Windows 2000 installation steps” on page 181 provides detailed installation procedures for a number of scenarios on the Microsoft Windows 2000 platform.

Chapter 10, “AIX installation steps” on page 263 provides detailed installation procedures for a number of scenarios on the IBM AIX platform.

Chapter 11, “Solaris installation steps” on page 329 provides detailed installation procedures for a number of scenarios on the Sun Solaris platform.

Chapter 12, “Linux installation steps” on page 399 provides detailed installation procedures for a number of scenarios on the Linux platform.

Part 4, “Configuring WebSphere” on page 439

Chapter 13, “WebSphere administration basics” on page 441 introduces the WebSphere Administrative Console and takes you through some of the common WebSphere administrative tasks.

Chapter 14, “Configuring the Web server interface” on page 481 explains how to go about configuring the Web server interface to WebSphere Application Server V4.0.

Chapter 15, “Configuring session management” on page 513 discusses how you configure HTTP session support in WebSphere Application Server V4.0. The HTTP session is used to maintain state information during user’s interaction with a Web application.

Chapter 16, “Configuring WebSphere resources” on page 563 describes how to configure resource providers, including properties for JavaMail, URL, JMS, J2C and JDBC.

Chapter 17, “Server groups and workload management” on page 605 discusses workload management and how it is implemented in WebSphere via server groups and clones.

Chapter 18, “Packaging an application” on page 639 provides the practical steps to packaging a J2EE application. It also covers advanced packaging options, including IBM extensions to the J2EE standard.

Chapter 19, “Deploying an application” on page 687 takes you through setting up an application server environment, deploying an application to the application server, and deploying the client part of an application.

Chapter 20, “Packaging and deploying Web services” on page 721 explains how to package, deploy, and test a simple Web services in WebSphere Application Server V4.0.

Chapter 21, “Configuring security” on page 739 takes you through the steps needed to set up WebSphere security for a Web application and for the WebSphere Administrative Console.

Part 5, “Managing WebSphere” on page 837

Chapter 22, “Monitoring and tuning your runtime environment” on page 839 describes how to use the WebSphere Resource Analyzer and Performance Monitoring servlet to monitor your WebSphere environment. It also explains how to tune your environment using the Performance Tuner tool.

Chapter 23, “Command-line administration and scripting” on page 881 shows you how to create scripts to set up your WebSphere environment using WSCP and XMLConfig.

Chapter 24, “Troubleshooting” on page 949 discusses problem determination methods you can use when errors occur in your WebSphere Application Server V4.0 environment.

Chapter 25, “Migration” on page 1031 discusses migration of existing WebSphere 3.02 or 3.5.x Standard and Advanced Edition environments to WebSphere Application Server V4.0, Advanced Edition.

Part 6, “Appendixes” on page 1055

Appendix A, “Back up and restore your WebSphere environment” on page 1057 describes steps you can use to back up and to restore your WebSphere environment.

Appendix B, “The admin.config file definitions” on page 1063 provides definitions of the properties used in the admin.config file. This file contains many administrative server properties.

Appendix C, “The plugin-cfg.xml file definitions” on page 1071 provides definitions of the properties used in the plugin-cfg.xml file. This file contains Web server plug-in properties.

2

What's new in WebSphere V4.0

This chapter describes in brief the improvements and additions to IBM WebSphere Application Server V4.0 from Release V3.5. Some changes are very obvious, such as the three editions of the application server or the new look and feel of the WebSphere administrative console. Other changes are subtle, but critical, and affect the runtime, such as performance enhancements to connection manager. But the biggest change is architectural in nature, with the J2EE certification of the product.

The list of new/improved features in WebSphere V4.0 include:

1. J2EE compliance
2. WebSphere editions
3. Java 2 Connectors (JCA)
4. Web services
5. Web server plug-in
6. Embedded HTTP server
7. Performance enhancements
8. Administration tools
9. Other tools

10. Expanded platform support
11. Expanded database support
12. Migration
13. Enterprise Extensions

The above feature list is based on WebSphere V4.0.1 and will be discussed in the following sections.

2.1 J2EE compliance

J2EE 1.2 (Java 2 Enterprise Edition platform) certification of WebSphere brings with it all the benefits of the J2EE platform. Details about J2EE can be found in Chapter 3, “The Java 2 platform” on page 29.

WebSphere now maps to the J2EE Container Model, wherein all components operate within containers. WebSphere Application Server provides containers for the runtime environment. JSPs and servlets run in a Web Container, EJBs run in an EJB Container as before, an application client now runs in an Application Container, and even applets run in their own Applet Container.

The application server also provides services such as:

- Transaction integration and management via Java Message Service (JMS) with XA
- Messaging and e-mail interfaces through JavaBeans Activation Framework (JAF), Remote Method Invocation over Internet InterORB Protocol (RMI/IIOP), JavaMail, and JMS with the help of IBM MQ Series
- Connectivity to back-end systems via Java Database Connectivity (JDBC) 2.0 Core and Extension API

2.2 WebSphere editions

A major runtime architectural change with WebSphere Application Server V4.0 is the single code base that is J2EE compatible. The same core server is offered in various flavors known as editions. The full-fledged version is the Advanced Edition (AE). The features of AE and other editions are enumerated here.

2.2.1 Advanced Edition Version 4.0, Single Server (AEs)

- Lightweight, simplified implementation where the application server and the administrative server share a single JVM.

- No relational database requirement for repository. Configuration information is stored in an .xml file.
- Intended for development and unit testing.
- Offers browser-based administration only making it very convenient to administer the application server from a remote location.

Advanced Edition Version 4.0, Development Only (AEd)

- Identical to the Single Server Option, but licensing restricts usage to non-production environments only.
- Available for the Windows NT/2000 platform only.
- Currently available only as a free electronic download from the following Web site:

  http://www.ibm.com/software/info/websphere/r/link-wsdd-all.

2.2.2 Advanced Edition Version 4.0 (AE)

- Offers multiple/distributed server support, including distributed security support.
- Features clustering and cloning.
- Both the application server and the administrative server run in their own JVM.
- Full IIOP-based administration is possible via the administrative console.
- Intended for production and highly scalable deployment environments.
- Repository information is stored in a relational database. IBM's powerful database product, DB2 V7.1, is included with this edition.
- Quality of Service (QoS) enhancements.

2.2.3 Enterprise Extensions (EE)

- High-end functions, beyond the J2EE specifications, focusing on interoperability between distributed applications in large to very large-scale environments.
- Interoperability with CORBA and COM and the work area support, which allows applications to exchange context information.
- Extensions will be packaged as separate products and will require WebSphere Application Server V4.0, Advanced Edition as the base.

Feature differences between WebSphere Single Server Edition (AEs), WebSphere Developer Edition (AEd) and the Advanced Multi-Server Edition (AE) are listed in Table 2-1.

Table 2-1 Feature comparison for WebSphere AEs, AEd, and AE

Feature	AEs	AEd	AE
Full J2EE compliance	Yes	Yes	Yes
Web services support	Yes	Yes	Yes
Connection management and pooling	Yes	Yes	Yes
XML parsing	Yes	Yes	Yes
Expanded DB support	Yes	Yes	Yes
Built-in Web server	Yes	Yes	Yes
Web-based admin console	Yes	Yes	No
Firewall support	Yes	Yes	Yes
Production certified	Yes	No	Yes
Application server	One	One	Multiple
Machine configuration	Single	Single	Multiple
Security	Local OS	Local OS	OS, LDAP, Custom
Integration with Enterprise Edition	No	No	Yes
Directory Services	No	No	Yes
Application-level WLM	No	No	Yes
Clustering and cloning	No	No	Yes
Additional caching	No	No	Yes
Distributed security	No	No	Yes
IIOP-based admin console	No	No	Yes
Multi-node administration	No	No	Yes
IBM DB2 included	No	No	Yes
Merant JDBC drivers included	No	No	Yes
J2C supported	No	No	Yes

2.3 J2EE Connector Architecture

J2EE Connector Architecture or J2C is not part of the J2EE V1.2 specification. J2C is part of J2EE V1.3 but is provided as a techonology preview. This is one of the added value features of WebSphere V4.0 providing unparalleled connectivity.

J2C is similar to the Common Connector Framework (CCF) but is implemented for the Java platform. It provides specialized access to Enterprise Resource Planning (ERP) and mainframe systems such as CICS and IMS from IBM. As part of J2C, WebSphere provides three components:

- Common Client Interface API, which simplifies access to diverse back-end Enterprise Information Systems (EIS).
- Resource Adapter, which lets WebSphere talk to the back-end EIS. One-phase commit resource adapters are available for Host On-Demand, CICS, IMS, SAP, J.D. Edwards, PeopleSoft, Oracle Financial, etc.
- Connection Factory, which connects an application to the Resource Adapter.

2.4 Web services

Web services enable businesses to connect applications to other business applications, to deliver business functions to a broader set of customers and partners, to interact with marketplaces more efficiently, and to create new business models dynamically.

To that extent, WebSphere V4.0 provides four protocols that support Web services:

- Web Services Description Language (WSDL), an XML-based description language that provides a way to catalog and describe services
- Universal Discovery Description and Integration (UDDI), a global, platform-independent, open framework to enable businesses to discover each other, define their interaction, and share information in a global registry
- Simple Object Access Protocol (SOAP), a lightweight protocol for exchange of information in a decentralized, distributed environment
- eXtensible Markup Language (XML), which provides a common language for exchanging information.

2.5 Web server plug-in

Web server or HTTP server plug-in is the component that enables communication between the HTTP server and application server. It has a new transport protocol in WebSphere V4.0. It now uses the industry-standard HTTP transport protocol for non-secure transports and HTTPS for secure transports.

The plug-in can be configured via the administrative console. The plug-in can also be configured in three additional ways: automatically during the WebSphere installation process, set up as a custom service each time the application server is started, or using the GenPluginCfg command.

The older transports used in previous versions of WebSphere, Servlet Redirector and OSE transport, have been removed in WebSphere V4.0.

2.6 Embedded HTTP server

There is now an embedded HTTP server within the WebSphere Application Server. This Web server is very useful for testing purposes but should not be used in production environments.

You still need a file serving servlet to serve up static HTML pages. The change is that you end up having both static and dynamic Web files on the same application server machine.To re-iterate, this embedded Web server does not preclude the use of an external Web server.

2.7 Performance enhancements

Performance enhancements include dynamic caching (multi-tier), dynamic reloading of EJBs, and JNDI caching which are all actually carried over from WebSphere V3.5.3. JNDI Caching improves performance by caching “expensive” lookups. Caching of dynamic content - servlets and JSPs - was also added for improved throughput. Also known as dynamic cache, it is set up per node or application server using XML files and is most effective for non-user specific output such as mutual fund prices.

With all the performance enhancements comes the Performance Tuner wizard, which can be accessed from the WebSphere Administrative Console.

Database connection pooling has been improved to provide automatic connection cleanup. One of the advantages of moving to JDK 1.3 is the fact that garbage collection is now multi-threaded. The result is better performance under heavy loads.

2.8 New and improved administration tools

The WebSphere V4.0 administrative console has been re-designed primarily for J2EE compliance. It is not available in the Single Server Edition. With the redesigned icons and wizards, it is very easy to navigate the administrative console.

WebSphere Control Program (WSCP) and XMLConfig have both improved functionality. These tools allow administration tasks to be invoked from the command line. Their command syntax is similar to that in WebSphere V3.5, but they can handle new objects for WebSphere V4.0. Both tools are available only in the Advanced Edition (AE).

The new objects for WebSphere V4.0 are shown in Table 2-2. These objects represent new areas of functionality that can be handled in scripts.

Table 2-2 New WebSphere V4.0 objects

New object	Functional area
EnterpriseApp	J2EE enterprise applications
J2CConnectionFactory	Java 2 connectors
J2CResourceAdapter	Java 2 connectors
JMSConnectionFactory	Java Message Service
JMSDestination	Java Message Service
JMSProvider	Java Message Service
MailSession	JavaMail sessions
Module	J2EE modules
PmiService	Performance data
SecurityConfig	Global security settings
SecurityRoleAssignment	J2EE security roles
ServerGroup	Server groups and clones
URL	URL
URLProvider	URL

Note: The Web-based administrative console is available only in the Single Server (AEs) and Developer Edition (AEd).

2.8.1 WebSphere Administrative Console

The WebSphere Administrative Console is a graphical, Java-based administrative client to the WebSphere administrative server. This administrative console supports the full range of WebSphere Application Server V4.0, Advanced Edition administrative activities.

The main menu has a new item named Tools, which is used to bring up the Application Assembly Tool (AAT), the Log Analyzer, and/or the Resource Analyzer.

Some of the other new functionality are:

- A built-in Performance Tuner off the Wizards menu
- Options to view/clear the three types of messages that get displayed in the console messages window
- In the View menu, a new Runtime Inspector that gives details of items that are running within the application server

Some other changes that you may notice in the WebSphere V4.0 administrative console include:

- The topology and type views have been streamlined into a consolidated view
- Servlet engines are now known as Web containers, in line with J2EE
- Models are now called server groups

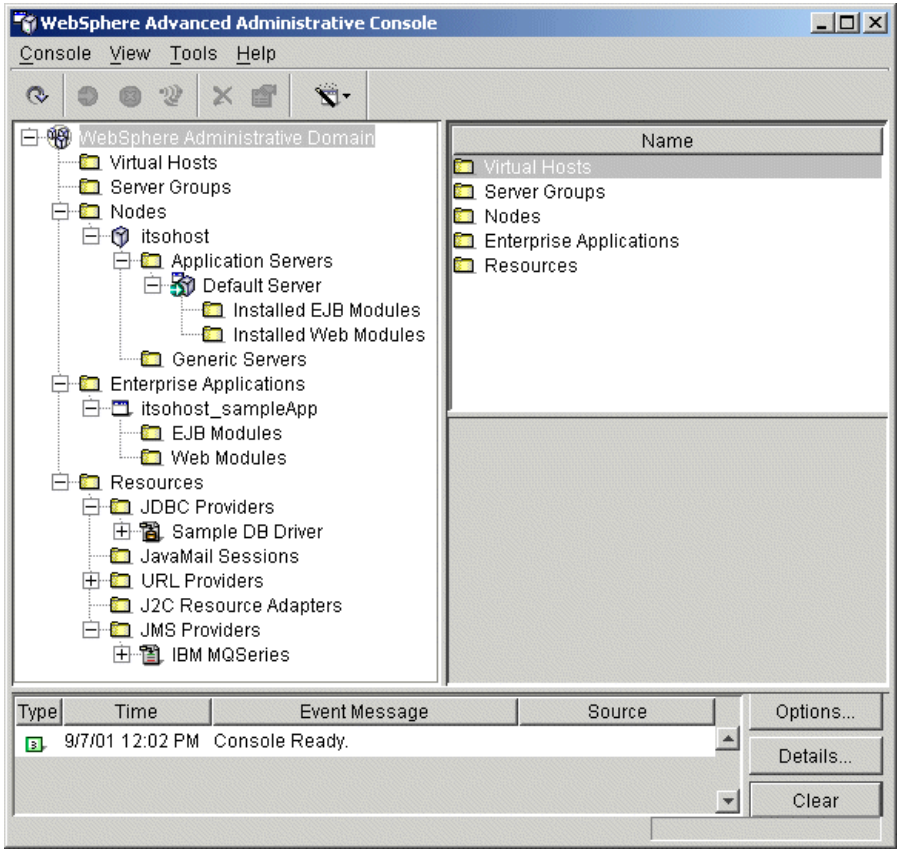


Figure 2-1 WebSphere Advanced Administrative Console window

2.9 Other tools

WebSphere V4.0 comes loaded with many new and improved tools. These tools are discussed below:

- First Steps is a desktop GUI which can start or stop the application server, launch the administrative console, launch the Application Assembly Tool, and monitor the application server. There is also a facility to learn about the application server. Some of these options require a Web browser.

Figure 2-2 WebSphere Application Server - First Steps window

- Application Assembly Tool (AAT) is a brand new tool resulting from J2EE compliance. AAT is used to assemble enterprise applications for deployment into WebSphere. Once the AAT is done with an application, the administrative console is used to install it into WebSphere.

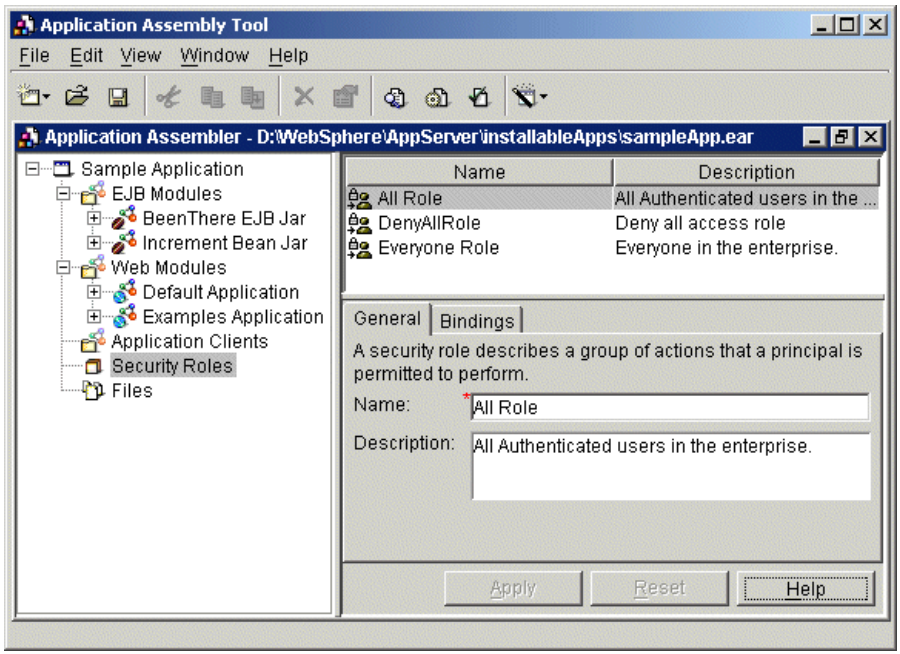


Figure 2-3 Application Assembly Tool (AAT)

- The EJB Deploy Tool is a command-line tool that prepares Enterprise JavaBeans for manual deployment. It is called by the graphical AAT behind the scenes.
- EARExpander reverses what the EJB Deploy Tool does. It is used to expand enterprise application archive (EAR) files into the format desired by the server runtime.
- SEAppInstall is a command-line tool used on the Single Server Edition (AEs) to install and uninstall J2EE applications.
- An improved Log Analyzer not only “formats” the message logs for humans, but also sorts, organizes, separates and analyzes WebSphere logs displaying details such as thread and process IDs of source events.
- An improved Resource Analyzer is a stand-alone program that monitors WebSphere Application Server and records details about request handling and the time taken to respond. It is built on the Performance Monitoring Infrastructure (PMI) Client API that is exposed to third-party developers.
- launchClient is a J2EE application client tool. It is a command line tool that launches a client application given an enterprise archive (EAR) file.

2.10 Expanded platform support

The number of platforms on which WebSphere is available has been further expanded with the release of WebSphere V4.0. WebSphere Application Server now runs on (relevant JDKs are noted in parentheses) the following platforms:

- AIX/6000 (IBM JDK 1.3)
- Windows NT, Windows 2000 (IBM JDK 1.3)
- Sun Solaris (Sun JDK 1.3)
- HP-UX (HP JDK 1.3)
- Linux - Red Hat, SuSE (IBM JDK 1.3)
- Linux/390, Turbolinux (V3.5, IBM JDK 1.2)
- OS/400
- z/OS
- OS/390

2.11 Expanded database support

WebSphere V4.0 adds database support for Informix 9.2.1. In addition, some of the previously supported databases are now supported at later versions.

At the time of writing, the following databases were supported:

- IBM DB2 UDB 7.2w Enterprise Edition
- IBM DB2 UDB 7.2w Workgroup Edition
- Oracle 8i Release 3 (8.1.7)
- Sybase Adaptive Server 12.0
- Informix Dynamic Server 9.2.1
- SQL Server 7.0 SP 2
- SQL Server 2000

You are strongly urged to check the following link for the most current and accurate information on supported databases and JDBC drivers:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

2.12 Migration

Migration tools are available with the Advanced Edition (AE) only. These tools allow the migration of existing configurations from the 3.0.2 and 3.5 levels to WebSphere V4.0 during the installation process. They include both pre-upgrade and post-upgrade activities. They do a much better job of handling security. And EJBs are migrated and then re-deployed.

Edition migration is new in WebSphere V4.0. There is a tool, called migration.sh/bat, that is provided to help with migrating from the Single Server Edition (AEs) to the Advanced Edition (AE).

The migration tools do not update application code.

2.13 Enterprise Extensions

In this section we discuss new features provided with WebSphere Application Server V4.0, Enterprise Extensions.

2.13.1 Business rule beans

Business rule beans enable application developers to separate reusable abstractions within a business domain and base these abstractions on policies, regulations or anything else that tends to change over time. Application developers are able to write rules to derive values, classify situations based on input values, and execute scripts. Business analysts can then manage rules externally (without changing code) through the sophisticated capabilities for rule modification intended to be offered as part of the business rule beans technology.

2.13.2 JMS listener and message bean support

The JMS listener and message bean technology provided with WebSphere Application Server V4.0, Enterprise Extensions enables seamless integration of component and messaging technologies. This technology simplifies the problems of integrating inbound messages and EJB operations by providing automatic message handling and subsequent processing with high integrity.

Note: The JMS listener and message beans provide *Message Driven Bean-like* functionality. However, support for Message Driven Beans as defined in EJB2.0 is not provided.

2.13.3 Globalization

An Internationalization service will allow applications to become “global”. WebSphere Application Server V4.0, Enterprise Extensions will determine the client locale and change all relevant attributes, such as currency, character sets, and so on.

2.14 Conclusion

This latest release of WebSphere Application Server not only meets all J2EE standards but goes beyond it by delivering additional functionality such as Web services and Enterprise Extensions. With all these new features, integrated tooling support, and extended platform support, WebSphere is poised to further expand on its phenomenal growth and success in the application server in the world.

3

The Java 2 platform

This chapter introduces the J2EE platform, the standard for developing, deploying, and executing enterprise applications. We give an overview of the different parts of the J2EE runtime environment: the application components, the containers, the standard services, and the resource managers. We also introduce the concepts of J2EE application packaging and deployment, including deployment descriptors.

Our goal is not to provide a detailed description of all the J2EE components or services, but rather to make sure you understand the “buzzwords” you have to deal with when working with J2EE enterprise applications. It is important that you understand all the concepts introduced in this chapter prior to packaging an application, as described in Chapter 18, “Packaging an application” on page 639.

We conclude this chapter with a list of documents and Web sites that can help you learn more about J2EE.

3.1 What is J2EE?

J2EE stands for Java 2 Platform, Enterprise Edition. It defines a standard that applies to all aspects of architecting, developing, and deploying multi-tier, server-based applications. The standard architecture defined by J2EE is composed of the following elements:

- **Standard application model** for developing multi-tier applications.
- **Standard platform** for hosting applications.
- **Compatibility test suite** for verifying that J2EE platform products comply with the J2EE platform standard.
- **Reference Implementation** providing an operational definition of the J2EE platform.

The J2EE platform specification describes the runtime environment for a J2EE application. This environment includes application components, containers, and resource manager drivers. The elements of this environment communicate with a set of standard services that are also specified. Figure 3-1 shows the J2EE server model and the distribution of J2EE components over multiple tiers.

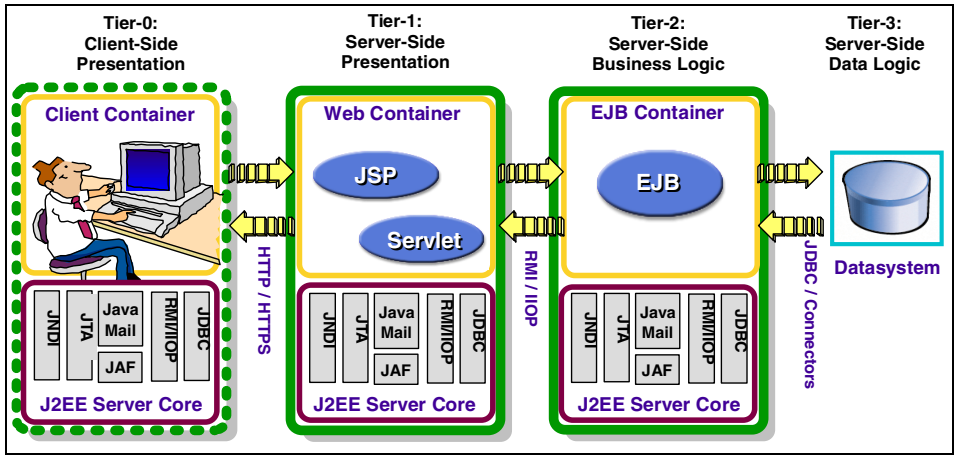


Figure 3-1 J2EE server model

3.1.1 J2EE platform roles

The J2EE platform also defines a number of distinct roles performed during the application development and deployment life cycle:

- **Product provider** designs and makes available for purchase the J2EE platform, APIs, and other features defined in the J2EE specification.

- **Tool provider** provides tools used for the development and packaging of application components.
- **Application component provider** creates Web components, enterprise beans, applets, or application clients for use in J2EE applications.
- **Application assembler** takes a set of components developed by component providers and assembles them in the form of an Enterprise Archive (EAR) file.
- **Deployer** is responsible for deploying an enterprise application into a specific operational environment.
- **System administrator** is responsible for the operational environment in which the application runs.

Product providers and tool providers have a product focus. Application component providers and application assemblers focus on the application. Deployers and system administrators focus on the runtime.

These roles help to identify the tasks that need to be performed and the parties involved. Understanding this separation of roles is important, because it helps to understand the approach that should be taken when developing and deploying J2EE applications.

3.1.2 J2EE benefits

The J2EE standard empowers customers. Customers can compare J2EE offerings from vendors and know that they are comparing apples with apples. Customer benefit as each new release of J2EE is agreed, as comprehensive, independent Compatibility Test Suites ensures vendor compliance with J2EE standards.

Some benefits of deploying to a J2EE-compliant architecture are:

- A simplified architecture based on standard components, services and clients, that takes advantage of Java's write-once, run-anywhere technology.
- Services providing integration with existing systems, including JDBC, JMS, Java IDL, JavaMail, and JTA for reliable business transactions.
- Scalability to meet demand, by distributing containers across multiple system and using database connection pooling, for example.
- A better choice of application development tools, and components from vendors providing off-the-shelf solutions.
- A flexible security model that provides single sign-on support, integration with legacy security schemes, and a unified approach to securing application components.

The J2EE specifications are the result of an industry-wide effort that has involved, and still involves, a large number of contributors. IBM alone has contributed to defining over 80% of the J2EE APIs.

3.1.3 WebSphere Application Server J2EE compliance

WebSphere Application Server V4.0 has completed the full J2EE certification test suite. As listed in Table 3-1, WebSphere V4.0 implements all of the J2EE 1.2 APIs. This table also provides a comparison with the API levels supported by WebSphere V3.5.2+.

Table 3-1 WebSphere and J2EE compliance

	API	WebSphere V3.5.2+	WebSphere V4.0
J2EE Components	Servlet	2.1, 2.2	2.2
	JSP	0.9, 1.0, 1.1	1.1
	EJB	1.0	1.1
J2EE Services	JDBC	1.0, 2.0	2.0
	JTA/JTS	1.0, 1.0.1, 1.1	1.1
	JNDI	1.2	1.2.1
	JAF	N/A	1.0
	XML4J	2.0.15	3.1.1
	XSL	1.0.1	2.0
J2EE Communication	RMI-IIOP		1.0
	JMS		1.0.1
	JavaMail	N/A	1.1

You can check Sun's list of J2EE-compatible configurations at:

`http://java.sun.com/j2ee/1.2_compatibility.html`

WebSphere V4.0 also provides a number of functions that exceed the J2EE 1.2 specifications, such as Web services support, Connector Architecture (technology preview), and JMS/XA interface to IBM MQSeries.

3.2 Application components

The J2EE programming model has four types of application components:

- Application clients
- Applets
- Servlets and JavaServer Pages
- Enterprise JavaBeans

As shown in Figure 3-2, each type of application component executes in a container. All containers are built on the Java 2 Platform, Standard Edition environment. See 3.3, “J2EE containers” on page 39 for our discussion on containers.

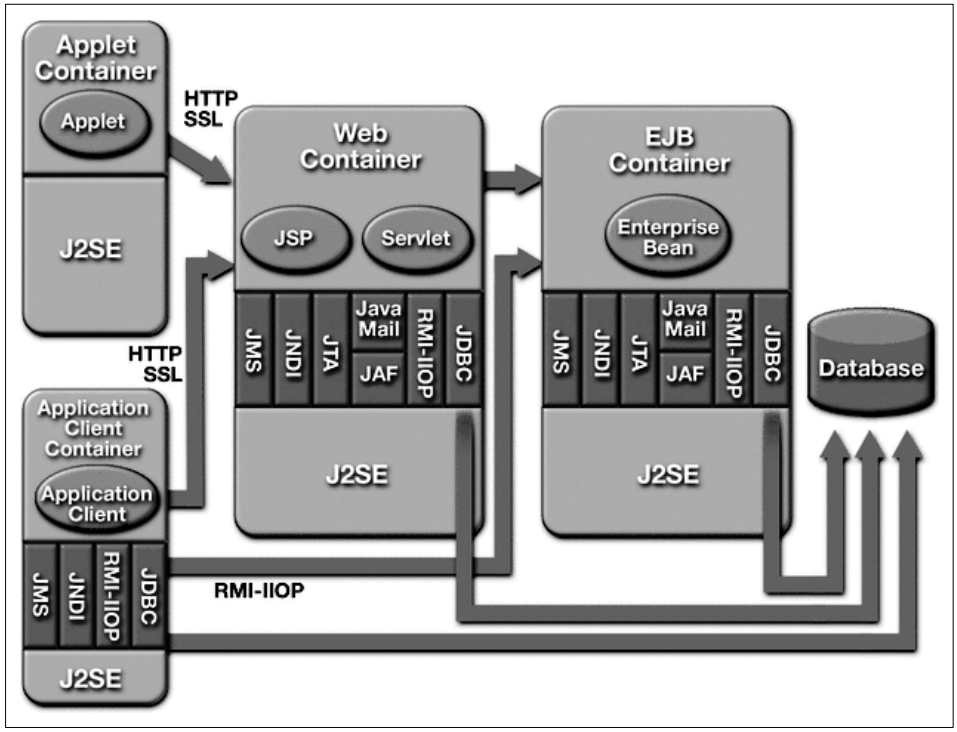


Figure 3-2 J2EE components, containers, and services

3.2.1 Application clients

Application clients are Java programs that typically run on a desktop computer with a graphical user interface (GUI). They have access to the full range of J2EE server-side components and services.

The application client component is often used where the user interface capabilities of a standard Web browser are not considered sufficient.

3.2.2 Applets

An applet is a client Java class that typically executes in a Web browser, but can also run in a variety of other client applications or devices.

Applets are often used in combination with HTML pages to enhance the user experience provided by a Web browser. They can also be used to shift some of the processing workload from the server to the client.

3.2.3 Servlets and JavaServer Pages

Servlets and JavaServer Pages (JSPs) are server-side components used to process requests from HTTP clients, such as Web browsers. They handle presentation and control of the user's interaction with the underlying application data and business logic. They can also generate formatted data, such as XML, for use by other application components.

Servlets and JavaServer Pages are deployed, managed and executed on a J2EE Web container and are often called "Web components".

Introduction to servlets

A servlet is a Java program that works with a Web server to generate dynamic content. Like a CGI (Common Gateway Interface) program, it receives client requests, handles them, and sends a response. If a servlet is called through HTTP, the response is typically an HTML flow. Unlike CGI programs that are loaded in memory each time a client makes a request, a servlet is loaded in memory once by the application server and can serve multiple requests in parallel using threads.

The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. HTTP servlets are a specialized servlet type providing a framework to handle the HTTP protocol, such as GET and POST methods. All HTTP servlets must inherit from the javax.servlet.http.HttpServlet class. A servlet's life cycle is composed of three phases:

- **init() method.** This method is called by the application server when the servlet is first loaded into memory. You can provide initialization parameters for a servlet in the Web application's configuration files.
- **service() method.** This method is called for each client request. For HTTP requests, the service method has been specialized to dispatch the request to the appropriate doGet, doPost, doPut, or doDelete methods, depending on the HTTP request method (DO, POST, PUT, DELETE). If you write HTTP servlets, you should not override the service() method, but rather override the appropriate doXXX method.

- **destroy() method.** This method is called when the application server unloads the servlet from memory. You should free any resources used by the servlet in this method.

Servlets are managed by the application server. They are loaded in memory upon the first client request, or at server startup. Each client request is then served on a different thread.

JavaServer Pages

The servlet and JavaServer Pages programming model is based on the Model-View-Controller (MVC) model. In the MVC model, the data (model), the logic manipulating the data (controller), and the presentation of the data (view) are designed to be independent. If the view needs to change, the business logic and the data are not affected. If the data interface changes, the controller can be updated without affecting the view.

In the MVC model, the servlet receives a request from a client, accesses the data through a set of reusable components (beans or enterprise beans), and invokes a JavaServer Pages (JSP) component to display the results of the request, as shown in Figure 3-3.

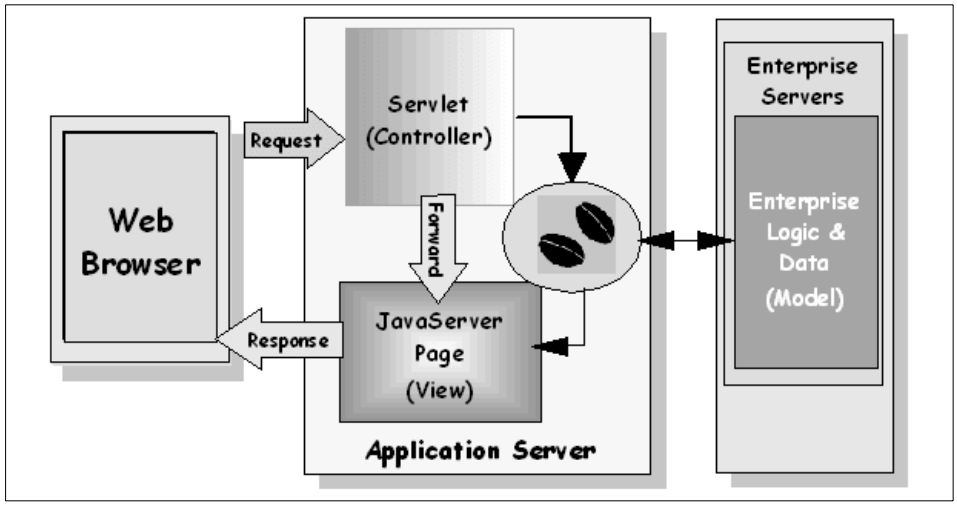


Figure 3-3 The Model-View-Controller design model

A JSP component is like a template for an HTML page, with slots for the dynamic content that varies on each the request. These slots are filled at runtime with dynamic data from the servlet, such as the user name's or the current time. The servlet must query the dynamic data, package it, and pass it to the JSP

component. The servlet stores dynamic data in a bean instance and places the bean instance somewhere the JSP component can access it. The JSP component retrieves the bean instance and inserts the dynamic data (typically bean properties) into the HTML page using special JSP tags.

When you open a JSP file it is compiled by a JSP compiler into a servlet. The servlet is then loaded into memory and run. This compilation process only occurs the first time the JSP is opened, or if you change the source code.

Maintaining JSP files is easier if you keep the Java code included to a minimum. If you include complex Java code fragments in the JSP, you will need to look in the view to find business logic. This is no better than editing a servlet to change the view of your application.

3.2.4 Enterprise JavaBeans

The Enterprise JavaBeans (EJB) specification is a foundation for the Java 2 Platform, Enterprise Edition (J2EE) defined by Sun. Vendors use this specification to implement an infrastructure in which components can be deployed, and use a set of services such as distributed transactions, security, or life-cycle management. As a developer, you just reuse the services detailed in the specification. For example, you do not need to include any code in your components to make them transactional. This lets you concentrate on the business logic of the application. Enterprise beans are designed to be portable from one vendor's execution environment to another, independent of the choices made by the vendor to implement the services described in the specification.

The quality of service required by an enterprise bean is described outside of the component in a deployment descriptor. The deployment descriptor is analyzed at deployment time by a tool provided by the vendor. This feature provides a great level of flexibility for reusing your component. For example, if you wish to change the transactional behavior of an enterprise bean, you need to change only the transaction attribute stored in the deployment descriptor, not the EJB business logic. Changes are taken into account when you re-deploy the enterprise bean in the container.

Enterprise JavaBeans architecture

The Enterprise JavaBeans specification defines the architecture shown in Figure 3-4. It allows server-side components to be reused across applications. In this architecture, an Enterprise Java Server (EJS) manages one or more containers.

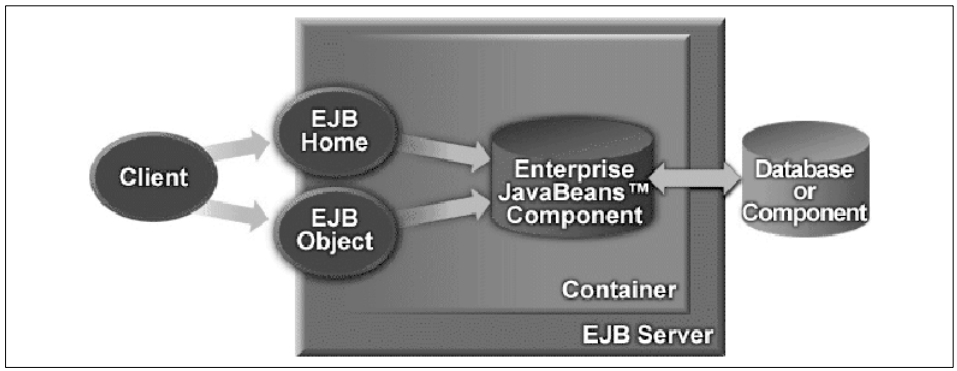


Figure 3-4 Enterprise Java Beans Architecture

EJB containers provide services to enterprise bean instances. Containers create bean instances, manage pools of instances, and destroy them. Containers also provide services such as persistency and security to the beans they manage. Containers are transparent to client programs, but the services of a container are used to invoke an enterprise bean. The container provides the service level specified at deployment time. For example, it will start a transaction before calling a method if you specified in the deployment descriptor that this method had to execute within a transaction context.

The container does not actually provide this service itself. It must communicate with the enterprise server, which implements the services, and obtain access to the service on behalf of the enterprise-bean instance. At a minimum, an EJS must provide JNDI (Java Naming Directory Interface) naming services and transaction services. WebSphere Application Server is an EJS.

Enterprise beans are distributed objects, so a factory service is used to manipulate bean instances. The EJBHome class provides this service. The EJBHome class implements the home interface, which defines the methods that client programs use to create and find bean instances. There is one EJBHome per enterprise bean type. A client must first locate the EJBHome class and then use it to create and find instances.

The EJBObject class implements the remote interface, which defines the business methods of the enterprise bean. The container creates the EJBObject class from this interface definition at deployment time. After a client program has access to the home object, it asks the container to create an enterprise bean instance. The container returns a reference to an EJBObject which the client program uses to access the bean instance.

The enterprise bean class contains the implementation of the methods defined on the remote interface, as well as other mandatory methods defined in the specification, such as ejbCreate(), ejbActivate(), or ejbRemove(). Those methods will be called by the container whenever needed. For example, when you call a create() method on the EJBHome, the container will create an EJBObject, create an execution context, create an instance, and then invoke ejbCreate() on the instance.

Enterprise bean types

There are two types of enterprise beans: session beans and entity beans. A session bean instance belongs to a specific client. Session beans can maintain a state on behalf of a client, but this information will not be saved when the bean instance is destroyed and will be lost if the server crashes. An example of session bean usage is a shopping cart. A session bean instance is created for each client connecting to the online shop. When the client leaves, the session bean is destroyed. The shopping cart contents need to be kept while the client is connected, but do not need to be persistently saved afterwards.

However, an invoice for the online shopper must be persistently saved. We could use an entity bean for that purpose. Entity beans represent data, typically a row in a database table. In this case, creating an entity bean is equivalent to adding a record to the table. Even if the bean instance is removed or if the server crashes, the bean instance can be recreated from its persistent state. Entity beans can be shared by multiple clients, so the invoice could be edited by the shipping department and while being viewed by the sales department.

Session beans

There are two types of session beans: stateless and stateful. A container typically creates a pool of session bean instances to serve multiple clients. With stateless session beans, the container may allocate a different bean instance from the pool for each method call. So, if a client calls methodA() on a session bean, and then calls methodB(), those requests may be served by a different bean instance. A stateless session bean does not maintain any state on behalf of a client. Stateless session beans are efficient because the container can use a small number of instances to serve a large number of clients.

With stateful session beans, the container must allocate the same bean instance for each method call. A new instance is created each time a client invokes create() on the home interface. With stateful beans, you can save data in the bean instance as this data will still be available on the next method call.

Entity beans
Entity beans persistency can be handled either by the developer or by the container. With container managed persistence (CMP), all persistency is delegated to the container. The developer only needs to write the required business logic. For bean managed persistence (BMP), the developer must provide the code that the container uses when it decides to save or restore the state of the enterprise bean.

3.3 J2EE containers

J2EE containers provide the runtime support of the application components we discussed in 3.2, “Application components” on page 32. There must be one container for each application component type in a J2EE application. By having a container between the application components and the set of services, J2EE can provide a federated view of the APIs for the application components.

A container provides the APIs to the application components used for accessing the services. It may also handle security, resource pooling, state management, and naming and transaction issues. You can see the containers, the services they use and their components in Figure 3-2 on page 33.

The various containers and the services they provide are:

- **Application client container** supports application client components. It must provide access to the set of services required by J2EE but is not required to manage transactions. Application clients have access to the Java API and are packaged in a JAR file.
- **Applet container** supports the applet programming model. Typically a J2SE (Java 2 Platform, Standard Edition) 1.2 compatible applet execution environment acts as a container. The Java plug-in may be added to the browser to obtain such a container. Applets communicate over HTTP if they run in a browser, but they can also communicate using serialized objects.
- **Web container** handles requests for servlets, JSP files, and other types of server-side include coding. It creates servlet instances, loads and unloads servlets, creates and manages request and response objects, and performs other tasks for managing servlets effectively.
- **EJB container** provides an interface between deployed enterprise beans and the application server. Together, the container and server provide the bean runtime environment. The container provides many low-level services, including threading and transaction support. It also manages data storage and retrieval for the beans within.

3.4 Standard services

The J2EE platform provides components with a set of standard services that they can use to interact with each other. In this section we provide a short description of each service.

3.4.1 HTTP and HTTPS

The HTTP protocol is used to request and receive files (pages, images, and so on) over the Internet, or from other computer networks.

The client-side API is defined by the java.net package that provides the classes for implementing networking applications. The server-side API is defined by the servlet and JSP interfaces.

The same client and server APIs are required to support HTTP over a Secure Socket Layer (HTTPS).

3.4.2 Java Naming and Directory Interface (JNDI)

The JNDI API allows J2EE components to locate other objects that they may need to access. This API provides standardized access to a variety of naming and directory interfaces. It has an application-level interface that is used by application components to access naming and directory services, and a service provider interface that is used to attach a provider of a naming and directory service to the J2EE platform.

The J2EE platform provides modules with their own local JNDI namespace, java:comp/env. When a module is created, each component must define all the resources that it expects to find in the local JNDI namespace in its deployment descriptor. These resources include enterprise beans, JDBC data sources, and so on.

When the module is deployed, the local references declared in the deployment descriptor are mapped to the global JNDI names of the actual deployed resources that the modules want to locate. At runtime, when a client performs a JNDI lookup in its local JNDI namespace, the container uses the information supplied when the application was installed to map the local name understood by the module to the global name that identifies where the component is actually located.

3.4.3 Java DataBase Connectivity (JDBC)

The JDBC 2.0 Core API provides connectivity with relational database systems. The JDBC 2.0 Extension API is required for connection naming via JNDI, connection pooling, and distributed transaction support.

The JDBC API provides an application-level interface used by the components for accessing databases, and a service-provider interface to attach a database driver to the platform.

3.4.4 Java Message Service (JMS)

The JMS API defines a standard mechanism for using enterprise messaging system, such as IBM MQSeries. JMS supports point-to-point messaging where clients send messages to the message queues of other clients, and publish/subscribe messaging where clients publish to, and subscribe from, well-known topics.

3.4.5 JavaMail and JavaBeans Activation Framework (JAF)

The JavaMail API allows an application to send e-mail messages. It has an application-level interface used by the application to send mail, and a service-provider interface to attach a provider to the platform.

JavaMail includes the JavaBeans Activation Framework API (JAF). It is used by JavaMail to handle the data included in e-mail messages.

3.4.6 Java Transaction API (JTA and JTS)

JTA and JTS provide the J2EE platform with a distributed transaction management service and associated API that is based on the CORBA Object Transaction Service.

The JTA API specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server and the transactional applications.

A Java Transaction Service (JTS) transaction manager component provides transaction services to the parties involved in distributed transactions: the application server, the resource manager, the transactional application, and the resource manager(s).

3.4.7 Remote Method Invocation/Internet Inter-ORB Protocol

Remote Method Invocation (RMI) APIs allow developers to build distributed applications in the Java programming language. They enable an object running in one Java Virtual Machine to access another object running in a different Java Virtual Machine.

The Internet Inter-ORB (Object Request Broker) Protocol (IIOP) is a protocol used for communication between CORBA object request brokers. An object request broker is a library that enables CORBA objects to locate and to communicate with one another.

RMI/IIOP is an implementation of the RMI API over IIOP that allows developers to write remote interfaces in the Java programming language.

3.4.8 Java Interface Definition Language (Java IDL)

Java IDL allows J2EE components to invoke operations on remote CORBA objects (that may not be Java objects) that have been defined using IDL. Java IDL includes an IDL-to-Java compiler and an Object Request Broker (ORB) that supports Internet Inter-ORB Protocol (IIOP).

3.4.9 XML

J2EE specifications define a set of deployment descriptors using XML document type definitions (DTDs). These deployment descriptors are packaged with J2EE modules. We look closer at deployment descriptors in 3.6.2, “Deployment descriptors” on page 45.

3.5 Resource managers

A resource manager provides access to a shared resource, such as a database. Resource manager drivers are software components that provide network connectivity to an external resource manager. A driver can extend the functionality of the J2EE platform by implementing one of the J2EE standard service APIs, such as a JDBC driver, or by defining and implementing a resource manager driver for connecting to an external system. Drivers interface with the J2EE platform using the J2EE Service Provider Interface (J2EE SPI).

3.6 Packaging J2EE applications

The J2EE specification provides instructions for assembling, packaging, and deploying J2EE applications. As shown in Figure 3-5, J2EE components are packaged into modules, modules are then packaged into applications, and applications are then deployed. Each module and application contains a J2EE deployment descriptor.

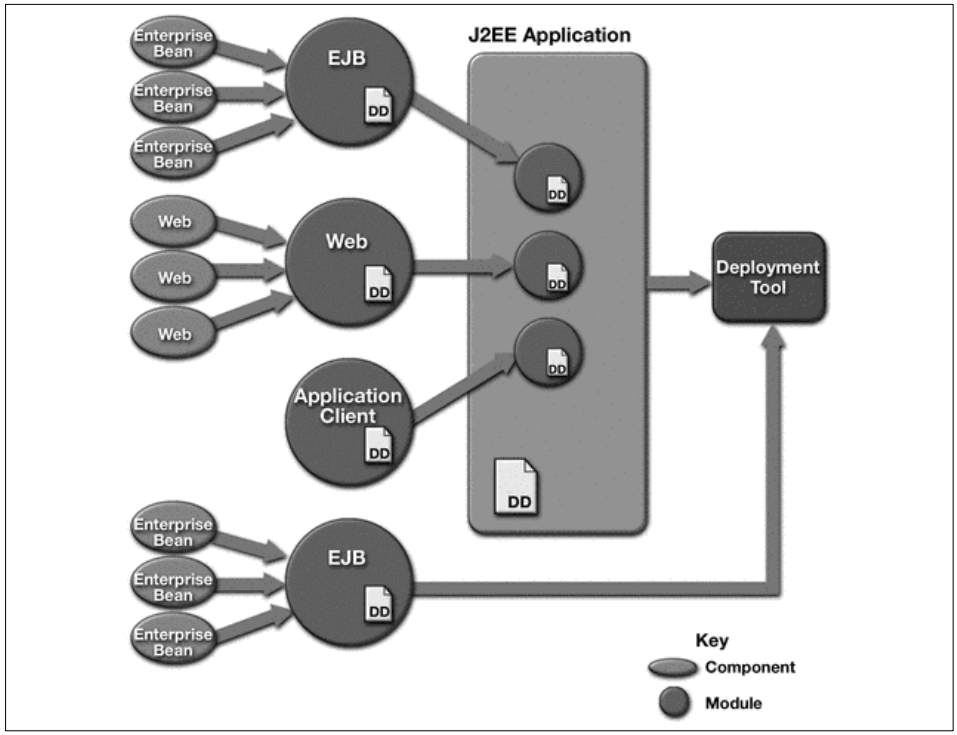


Figure 3-5 J2EE packaging and deployment overview

3.6.1 Assembling modules and applications

J2EE modules are used to assemble application components into deployable units. Assembly is the process of specifying which files make up the module, creating deployment descriptor files for the module, and packaging these files in an archive. Modules are used to represent any of the following:

- One or more Web components. Web components are servlets or JavaServer Pages (JSPs).
- One or more enterprise beans.
- Application clients.

A J2EE application consists of any combination of the above.

The standard JAR file format is used to package modules, regardless of the type of module. However, the archive file is referred to according to its content:

- EJB modules are packaged in JAR files.
- Web modules are packaged in Web archive (WAR) files.
- Application-client modules are packaged in JAR files.
- Enterprise applications are packaged in Enterprise Archive (EAR) files.

Table 3-2 lists the files packaged in an Enterprise Archive (EAR) file, which could make up a sample J2EE enterprise application.

Table 3-2 Sample J2EE enterprise application archive contents

File	Description
/META-INF/application.xml	The application's deployment descriptor
/META-INF/MANIFEST.MF	Standard JAR file manifest
/client_module.jar	An application-client module
/ejb_module.jar	An EJB module containing EJBs used by the application
/helper_classes.jar	A JAR file containing the helper classes used by other modules in the application
/web_module.war	A Web module containing the application's Web components

Note that the enterprise application in Table 3-2 contains the web_module.war Web module. Table 3-3 lists the contents of this sample Web archive (WAR) file.

Table 3-3 Sample J2EE Web module archive contents

File	Description
/web_module.war/META-INF/MANIFEST.MF	Standard JAR file manifest
/web_module.war/WEB-INF/web.xml	The module's deployment descriptor
/web_module.war/WEB-INF/classes/MyServlet.class	A servlet class file
/web_module.war/index.jsp	A JavaServer Page file
/web_module.war/logo.gif	An image file

Note: JARs using helper classes located in a separate JAR should have a reference to the helper class's JAR in their manifest file. In WebSphere, you also need to set the application server "Module visibility" property to "Application".

The J2EE platform supports two deployment units: one or more J2EE modules can be combined into an enterprise application for deployment, or a single module can be deployed as a stand-alone application.

3.6.2 Deployment descriptors

Deployment properties of J2EE modules and applications are defined using deployment descriptors. The deployment descriptor is an XML document that contains application configuration data that the runtime uses.

A J2EE application contains one application-level deployment descriptor file, for the application as a whole. Each module in the application also contains its own deployment descriptor file. The application-level deployment descriptor specifies information such as the application name, the modules it contains, and security information. The module-level deployment descriptor includes the following:

- Information about the content of the module being assembled. For example, for an EJB module, the deployment descriptor lists each enterprise bean's class, home interface class, remote interface class, whether the bean is an entity or session bean, and the bean's attributes (such as persistence management type and primary key class for entity beans).
- References to a module's internal and external dependencies (such as enterprise beans, databases, and resource connection factories needed by the module).
- Runtime-specific information needed by the module, for example the servlet mappings needed for a Web application or the persistence management (BMP or CMP) to be used by an entity bean.
- References to security roles. Security information is used when the module is deployed.

IBM bindings

The J2EE 1.2 specification does not provide a mechanism to map a logical external resource name, specified in a module's deployment descriptor, to its actual name in the global JNDI namespace.

WebSphere V4.0 defines IBM bindings for this purpose. Bindings can be configured in the Application Assembly Tool (AAT), often at the same time you define the deployment descriptor. They can also be configured using WebSphere Studio Application Developer.

They are stored in the EJB, Web, or application archive in a file called ibm-<type>-bnd.xmi, where <type> is ejb-jar, web, or application.

IBM extensions

WebSphere V4.0 supports additional options, that are beyond J2EE specifications, such as transaction scoping attributes, and Web application reloading.

IBM extensions are used to specify properties for these additional options. Extensions can be configured in the Application Assembly Tool (AAT), along with the deployment descriptor and IBM bindings. They can also be configured using WebSphere Studio Application Developer.

They are stored in the EJB, Web, or application archive in a file called ibm-<type>-ext.xmi, where <type> is ejb-jar, web, or application.

EJB JAR additions

WebSphere EJB modules include schema-related files for CMP entity beans. The Application Assembly Tool can be used to generate the files listed in Table 3-4.

Table 3-4 EJB JAR additions

File	Description
META-INF\Table.ddl	Commands to create a database table (used manually)
META-INF\map.mapxmi	Maps the EJB fields to the schema fields
META-INF\Schema\schema.rdbxmi	Specifies the database schema field (column) attributes

3.7 Classloaders

WebSphere Application Server uses several classloaders to enable the J2EE specification. Besides the regular classloader that uses the classpath environment variable to locate and load classes, there are many other classloaders at work.

Figure 3-6 shows a simplified diagram of the WebSphere classloaders. Each box represents a classloader, and the text in brackets describes where that classloader searches for classes.

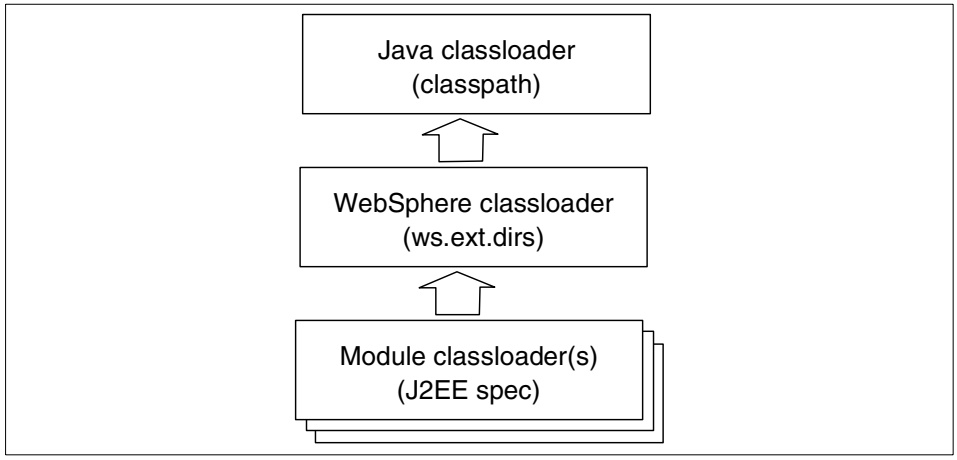


Figure 3-6 WebSphere classloaders

At the top, the regular Java system classloader uses the classpath environment variable to load classes. The second classloader is WebSphere-specific and uses a ws.ext.dirs environment variable to load classes. Besides loading any user code, this classloader also loads all of the WebSphere and J2EE classes that are required at runtime. Finally, the modules running in the server are loaded by one or more module classloaders. The module classloaders follow J2EE class loading rules to load the classes and JAR files from your application.

Web modules have the WEB-INF/classes and WEB-INF/lib folders for Java code. The classes folder may contain "loose" Java classes (classes not inside a JAR file), and it can be used for servlet or utility classes within the scope of the Web application. Often a special classloader is used for this folder, so that if changes are made to the classes, they are automatically reloaded by the application server. The lib folder may contain JAR files (not ZIP files) that are also used by the Web application. You should place third-party JAR files and other utility JAR files in this folder.

An enterprise application may contain JAR files to be used by the contained modules. This allows sharing of code at the application level, and is the best place to put utility JAR files that are used by multiple Web or EJB modules. By placing these JAR files in the enterprise application instead of on the global classpath, they are also within the J2EE specification, and do not require special publishing and setup when moving to a new server.

The most important point to understand about Figure 3-6 is that each classloader is defined as a child of the classloader above it. Whenever a class needs to be loaded, the classloader usually delegates the request to its parent classloader. If none of the parent classloaders can find the class, the original classloader attempts to load the class. Requests can only go up the tree; they cannot go down. If the WebSphere classloader is requested to find a class in a J2EE module, it cannot go down to the module classloader to find that class, and a ClassNotFoundException will occur. Once a class is loaded by a classloader, any new classes that it tries to load will reuse the same classloader, or go up the chain until the class is found.

For more information on WebSphere classloaders, see:

- WebSphere InfoCenter, “Setting classpaths” article.

  http://www.ibm.com/software/webservers/appserv/infocenter.html

- IBM WebSphere Developer Domain, “J2EE Class Loading Demystified” article.

  http://dbsd01.boulder.ibm.com/wsdd/library/techarticles/0112_deboer/deboer.html

3.8 More information

These documents and Web sites are also relevant as further information sources:

- See the Java 2 Platform, Enterprise Edition home page for J2EE specifications, tutorials, BluePrints, and white papers:

  http://java.sun.com/j2ee

- Enterprise JavaBeans Specification, V1.1

  http://java.sun.com/products/ejb

- IBM WebSphere Developer Domain

  http://www.ibm.com/websphere/developer

- IBM developerWorks

  http://www.ibm.com/developer

- WebSphere InfoCenter, *Concepts and terminology* section.

  http://www.ibm.com/software/webservers/appserv/infocenter.html

- Subrahmanyam Allamaraju, et al. *Professional Java Server Programming, Second Edition*, Wrox Press, 2000, ISBN 1861004656

Part 2

Inside WebSphere

In this part we look inside WebSphere V4.0 at architecture and topology alternatives, Web services, and J2EE security.

4

WebSphere architecture overview

In this chapter we take a look at the major components within IBM WebSphere Application Server V4.0, such as the application server, Web container, and EJB container, and how they map to the J2EE component architecture.

We talk about the WebSphere administrative server and the services that it provides. Then there is a discussion about the application server and server groups. Virtual hosts and enterprise applications are briefly touched upon.

When you install and run WebSphere on a single machine you will see certain key processes running. In this section we give a brief introduction to these processes and their purpose. In later chapters we give more details and describe other optional facilities.

Figure 4-1 on page 52 shows the architecture of WebSphere Application Server V4.0, Advanced Edition with all the major components. The following sections describe the components shown in this figure.

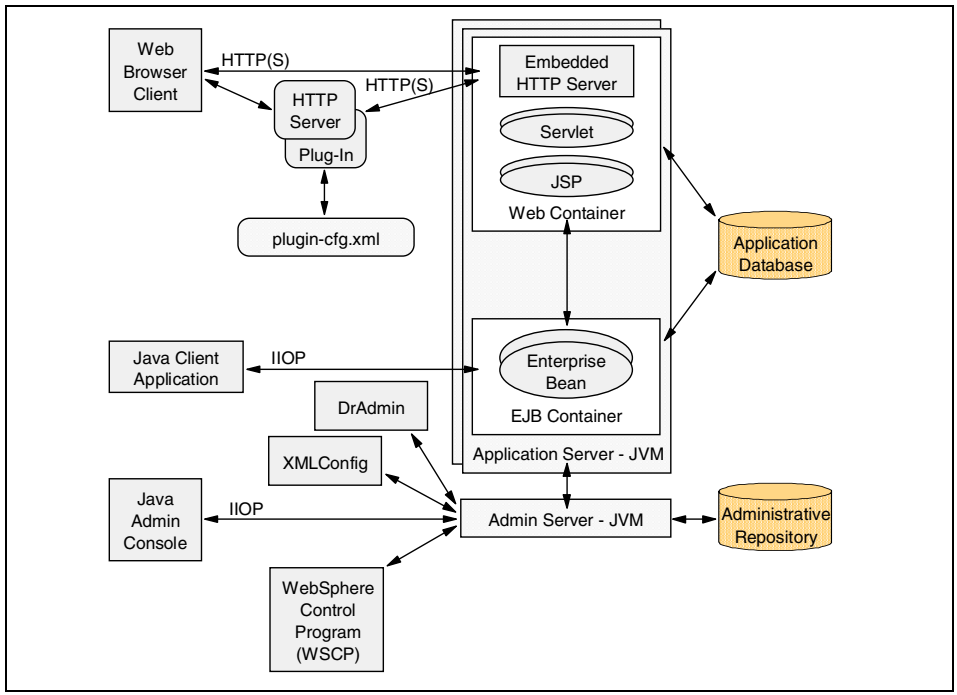


Figure 4-1 Major components in WebSphere V4.0, Advanced Edition

4.1 Application server

The application server collaborates with the Web server to return customized response to a client's request. Application code including servlets, JSPs, EJBs and their supporting classes run in an application server. In keeping with the J2EE component architecture, servlets and JSPs run in a Web container, and EJBs run in an EJB container.

In WebSphere Advanced Edition you can define multiple application servers, each running in its own Java Virtual Machine (JVM). The administrative server runs in its own JVM.

Refer to Chapter 13, "WebSphere administration basics" on page 441 for details on application servers.

Note: In WebSphere V4.0, Web application URLs are now case-sensitive on all operating systems for consistency and security.

4.1.1 Default Server

A default application server, appropriately named “Default Server”, is automatically configured during the default WebSphere Application Server installation. The Default Server, like any other application server, contains a Web container and an EJB container.

4.2 HTTP server and plug-in

The WebSphere Application Server works with an HTTP server, or Web server, to handle requests for dynamic content, such as servlets, from Web applications. (We use the terms HTTP server and Web server interchangeably thoughout the book.) The HTTP server and application server communicate using the WebSphere HTTP plug-in for the HTTP server.

The HTTP plug-in uses an easy-to-read XML configuration file to determine whether a request should be handled by the Web server or the application server. It uses the standard HTTP protocol to communicate with the application server. It can also be configured to use secure HTTPS, if required.

The HTTP plug-in is available for popular Web servers, including IBM HTTP Server, Apache, Microsoft IIS, and Netscape iPlanet.

You can find out more about the WebSphere Web server interface in Chapter 14, “Configuring the Web server interface” on page 481.

4.3 Embedded HTTP server

A nice feature of WebSphere V4.0 is the embedded HTTP server within the application server. This Web server is very useful for testing or development purposes but should not be used in production environments. For performance and security reasons, use a Web server and HTTP plug-in for the Web server in a production environment.

See Chapter 14, “Configuring the Web server interface” on page 481 for further details on using WebSphere without an external Web server.

4.4 Virtual hosts

A virtual host is a configuration enabling a single host machine to resemble multiple host machines. It allows a single physical machine to support several independently configured and administered applications. It is not associated with a particular node (machine). It is a configuration, rather than a "live object", explaining why it can be created, but not started or stopped.

Each virtual host has a logical name and a list of one or more DNS aliases by which it is known. A DNS alias is the TCP/IP host name and port number used to request the servlet, for example yourHostName:80. The default ports and aliases are:

- The default alias is *:80, using an external HTTP port that is not secure
- Aliases of the form *:9080 use the embedded HTTP port that is not secure
- Aliases of the form *:443 use the secure external HTTPS port
- Aliases of the form *:9443 use the secure embedded HTTPS port

When a servlet request is made, the server name and port number entered into the browser are compared to a list of all known aliases in an effort to locate the correct virtual host and serve the servlet. If no match is found, an error (404) is returned to the browser.

WebSphere Application Server provides a default virtual host, aptly named “default_host”, with some common aliases, such as the machine's IP address, short host name, and fully qualified host name. The alias comprises the first part of the path for accessing a resource such as a servlet. For example, it is localhost:80 in the request `http://localhost:80/servlet/snoop`.

Virtual hosts allow the administrator to isolate, and independently manage, multiple sets of resources on the same physical machine.

See Chapter 14, “Configuring the Web server interface” on page 481 for more information on virtual hosts.

4.5 Server groups

A server group is a template for creating additional, nearly identical copies of an application server and its contents. It is a logical representation of the application server.

A server group has the same structure and attributes as the real application server. It may contain Web containers, EJB containers, servlets, EJBs, and other resources. The server group lets you view and modify any property associated with these logical objects. But the server group is not associated with any particular physical node. Nor does a server group correspond to any real server process running on any node.

Once you have created a server group, you can then create clones of that server. Server groups also help in the management of clones. For example, changing a server group will change all the clones and starting a server group will start all the clones.

You can find more details on server groups in Chapter 17, “Server groups and workload management” on page 605.

4.6 Clones

Cloning is the process of creating a server group based upon an existing fully configured server. These copies are called clones.

Clones are identical in every way to the server group from which they were created. Unlike server groups, the clones created from a server group represent real application server processes running on real physical nodes.

Application server clones can be hosted on a single machine whereby you get vertical scaling or be distributed across different machines, which gives you horizontal scaling. Clones can be used for workload management. A request for a server resource can be handled by any of the server clones.

Modifying the server group automatically propagates the changes to all of the clones when the clones are restarted. If a clone is modified directly, the clone no longer is identical to its server group. However, it continues to be part of its server group unless it is disassociated from the server group.

You can learn more about clones in Chapter 17, “Server groups and workload management” on page 605.

4.7 Web container

The WebSphere Web container processes servlets, JSP files and other types of server-side includes. Pre-J2EE, servlets would run in a servlet engine. Each Web container automatically contains a single session manager.

When handling servlets, the Web container creates a request object and a response object, then invokes the servlet service method. The Web container invokes the servlet's destroy method when appropriate and unloads the servlet, after which the JVM performs garbage collection.

The Web container supplies the PageListServlet to call a Java ServerPage (JSP) by name. The PageListServlet uses configuration information to map a JSP name to a Uniform Resource Identifier (URI), and the URI specifies a JSP file in the Web module.

Web container configuration provides information about the application server component that handle servlet requests forwarded by the Web server. The administrator specifies Web container properties including:

1. Application server name on which the Web container runs
2. Number and type of connections between the Web server and the Web container
3. Port on which the Web container listens

The WebSphere Administrative Console or the Web Administrative Console can be used to edit the configurations of Web containers. Each application server runtime has one logical Web container, which you can modify but not create or remove.

Figure 4-2 shows the WebSphere Administrative Console view of the Web modules installed in Default Server's Web container.

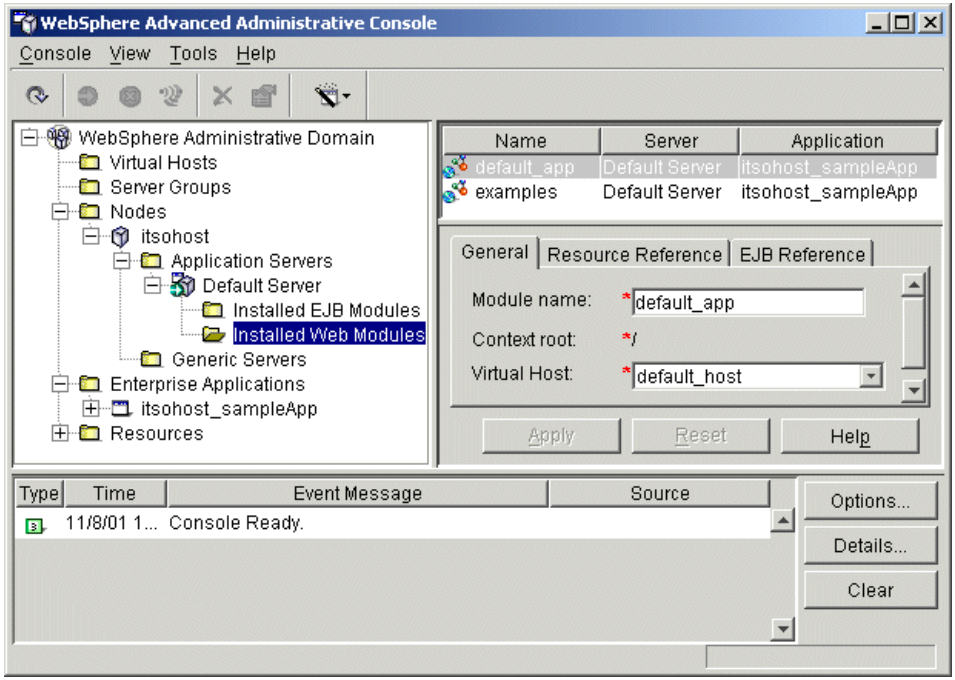


Figure 4-2 Web modules installed in Default Server's Web container

4.7.1 Web module

A Web module represents a Web application. It is used to assemble servlets and JSP files, as well as static content such as HTML pages, into a single deployable unit. Web modules are stored in Web archive files or WAR files (.war), which are standard Java archive files. A Web module contains one or more servlets, JavaServer Pages (JSP) files, and other files. It also contains a deployment descriptor that declares the contents of the module, stored in an XML file named web.xml. The deployment descriptor contains information about the structure and external dependencies of Web components in the module and describes how the components are to be used at runtime.

A Web module can be used as a stand-alone application, or it can be combined with other modules (other Web modules, EJB modules, or both) to create a J2EE application. A Web module is installed and run in a Web container.

You can find out more about Web containers and modules in Chapter 3, "The Java 2 platform" on page 29.

4.8 EJB container

The EJB container provides all the runtime services needed to deploy and manage Enterprise Java Beans (EJBs). It is a server process that handles requests for both session and entity beans.

The enterprise beans (inside EJB modules) installed in an application server do not communicate directly with the server; instead, an EJB container provides an interface between the EJBs and the server. Together, the container and the server provide the bean runtime environment.

The container provides many low-level services, including threading and transaction support. From an administrative viewpoint, the container manages data storage and retrieval for the contained beans. A single container can hold more than one EJB JAR file.

Figure 4-3 shows the WebSphere Administrative Console view of the EJB modules installed in Default Server's EJB container.

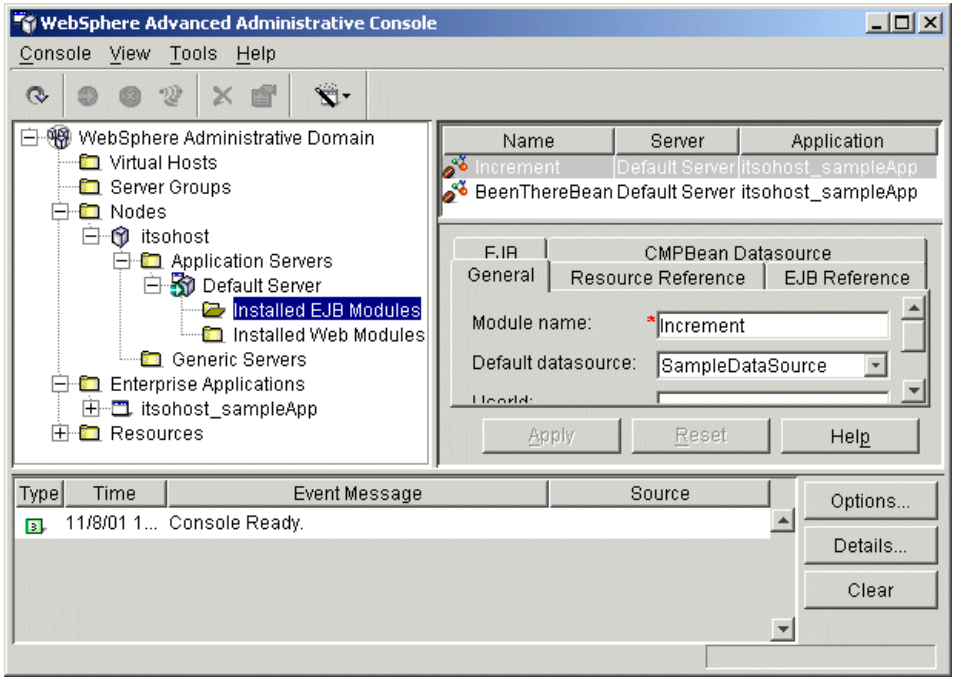


Figure 4-3 EJB modules installed in Default Server's EJB container

4.8.1 EJB module

An EJB module is used to assemble one or more enterprise beans into a single deployable unit. An EJB module is stored in a standard Java archive (JAR) file. An EJB module contains one or more deployable enterprise beans and a deployment descriptor stored in an XML file. The deployment descriptor declares the contents of the module, defines the structure and external dependencies of the beans in the module, and describes how the beans are to be used at runtime.

An EJB module can be used as a stand-alone application, or it can be combined with other EJB modules, or with Web modules, to create a J2EE application. An EJB module is installed and run in an enterprise bean container.

You can find out more about EJB containers and modules in Chapter 3, “The Java 2 platform” on page 29.

4.9 WebSphere administrative model

The WebSphere administrative model is depicted in Figure 4-4.

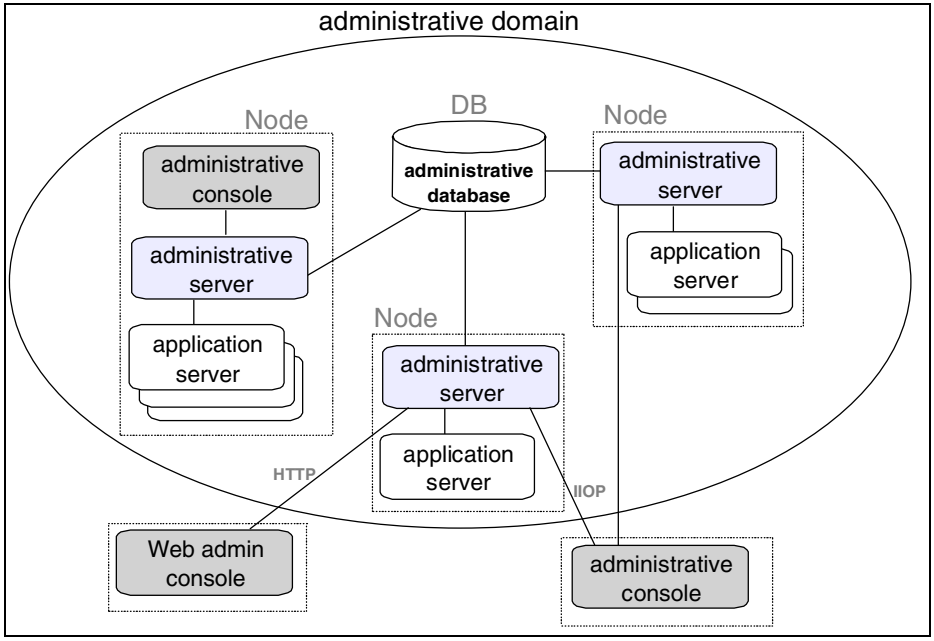


Figure 4-4 WebSphere administrative model

An administrative domain is a set of one or more nodes sharing an administrative repository in the form of a relational database. An administrative domain is the logical space containing the configurations for various objects in your WebSphere environment.

A node is a physical machine running an application server and an administrative server.

Each administrative server in the domain stores its administrative data in a shared repository, shown as the administrative database in Figure 4-4 on page 59.

Administrative user interfaces, outside the administrative domain, communicate with the administrative servers using IIOP or HTTP. WebSphere Advanced Edition uses the Java administrative console and IIOP. WebSphere Advanced Edition Single Server uses the HTTP Web administrative console.

WebSphere resources on a node are represented as administrative resources in the WebSphere administrative domain.

4.10 Administrative server

The administrative server is the systems management runtime component of WebSphere. The administrative server is responsible for runtime management, security, transaction coordination, and workload management. In most cases (exceptions will be outlined later), the administrative server runs on all nodes in a WebSphere administrative domain and controls the interaction between each node and application server process in the domain.

The WebSphere administrative server provides administrators with a single system view of applications and resources, such as JSPs, servlets, and EJBs, that could be deployed across multiple platforms in a distributed environment. Administering resources on a remote machine is just as easy as administering them on the local machine.

4.11 Administrative repository

WebSphere stores all runtime configuration information for a domain in a single persistent repository. That database by default is named *WAS*. All administration takes place through the manipulation of objects in the administrative repository. The respository can be stored in DB2, Oracle, Informix, MS SQL Server, or Sybase. In all cases you need to check the WebSphere release notes for exactly which versions of your chosen database you should use.

In the single server edition this repository is stored in an XML configuration file.

In our diagram we show a single node running all processes, and this is common in small production environments. It is entirely reasonable to configure the database on a remote server, and in production environments we recommend that you do so.

4.12 Administrative interfaces

The WebSphere administrative server provides the services that are used to control resources and perform tasks on the administrative database. Monitoring and configuring of administrative resources as well as stopping and starting of servers are facilitated by four interfaces, as shown in Figure 4-5 on page 61.

The two graphical interfaces and two command-line interfaces nicely complement each other. You can use the graphical interfaces to interactively administer your WebSphere environment. The Java administrative console is used in examples throughout this book. You can use the command-line tools to automate configuration. These tools are discussed in Chapter 23, “Command-line administration and scripting” on page 881.

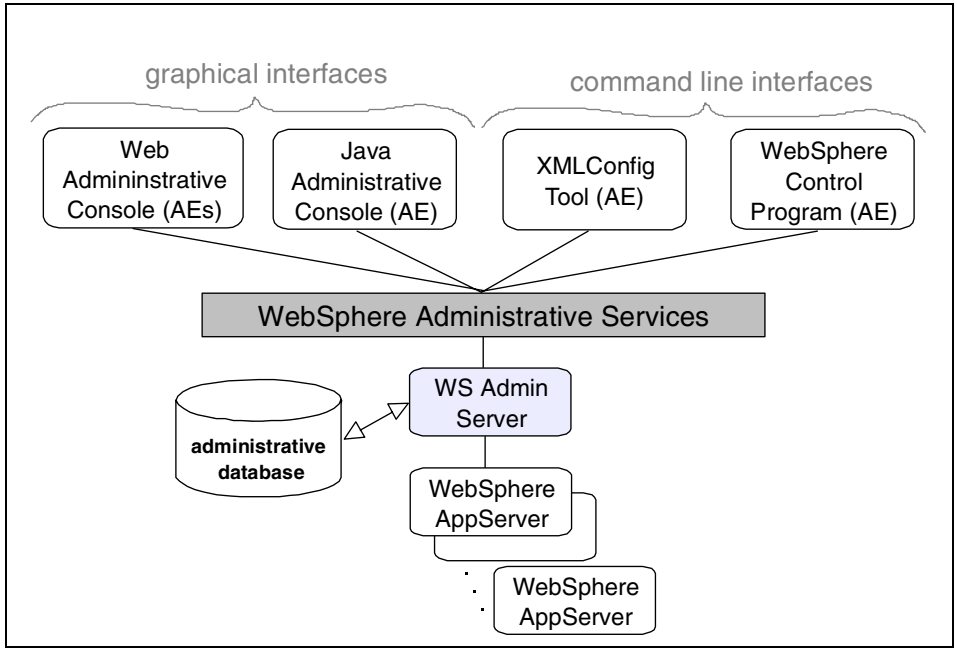


Figure 4-5 WebSphere V4.0 Administrative Services block diagram

4.12.1 Java administrative console

Commonly known as the WebSphere Administrative Console or simply as the admin console, this graphical user interface is primarily used for administration of a WebSphere administrative domain. It provides rich support for the full range of WebSphere Advanced Edition administrative activities. The administrative console can run on one of the nodes that the administrative server is running on, or it can be invoked on a remote node that attaches to a running administrative server.

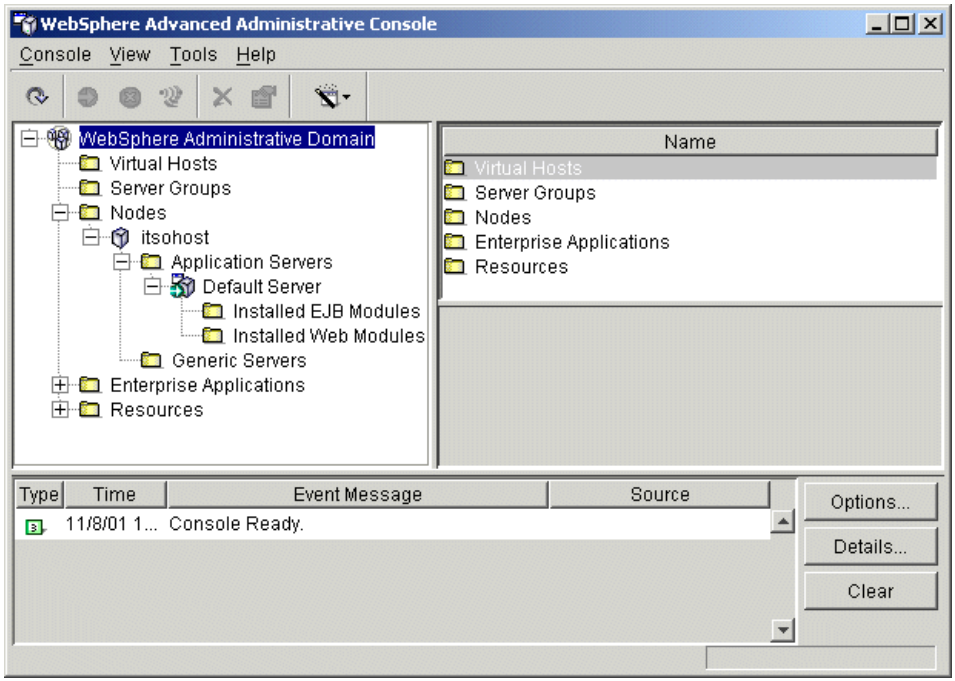


Figure 4-6 WebSphere V4.0 Advanced Edition administrative console

On Windows systems access the administrative console by clicking **Start > Progams > IBM WebSphere > Application Server V4.0 AE > Administrator's Console**. On UNIX systems invoke the adminclient.sh script found in <WAS_HOME>/bin to bring up the administrative console. By default the administrative console connects to the administrative server via port 900.

You can find further details on the WebSphere Administrative Console in Chapter 13, “WebSphere administration basics” on page 441.

4.12.2 Web administrative console

The Web administrative console is like a configuration editor that runs in a Web browser. It provides the opportunity to work with WebSphere Application Server V4.0, Advanced Edition Single Server configuration files encoded in eXtensible Markup Language (XML). This lightweight Web-based GUI is only available in WebSphere AEs.

The Web administrative console, shown in Figure 4-7, can be brought up on the local machine by typing the following URL in a Web browser:

```
http://localhost:9090/admin
```

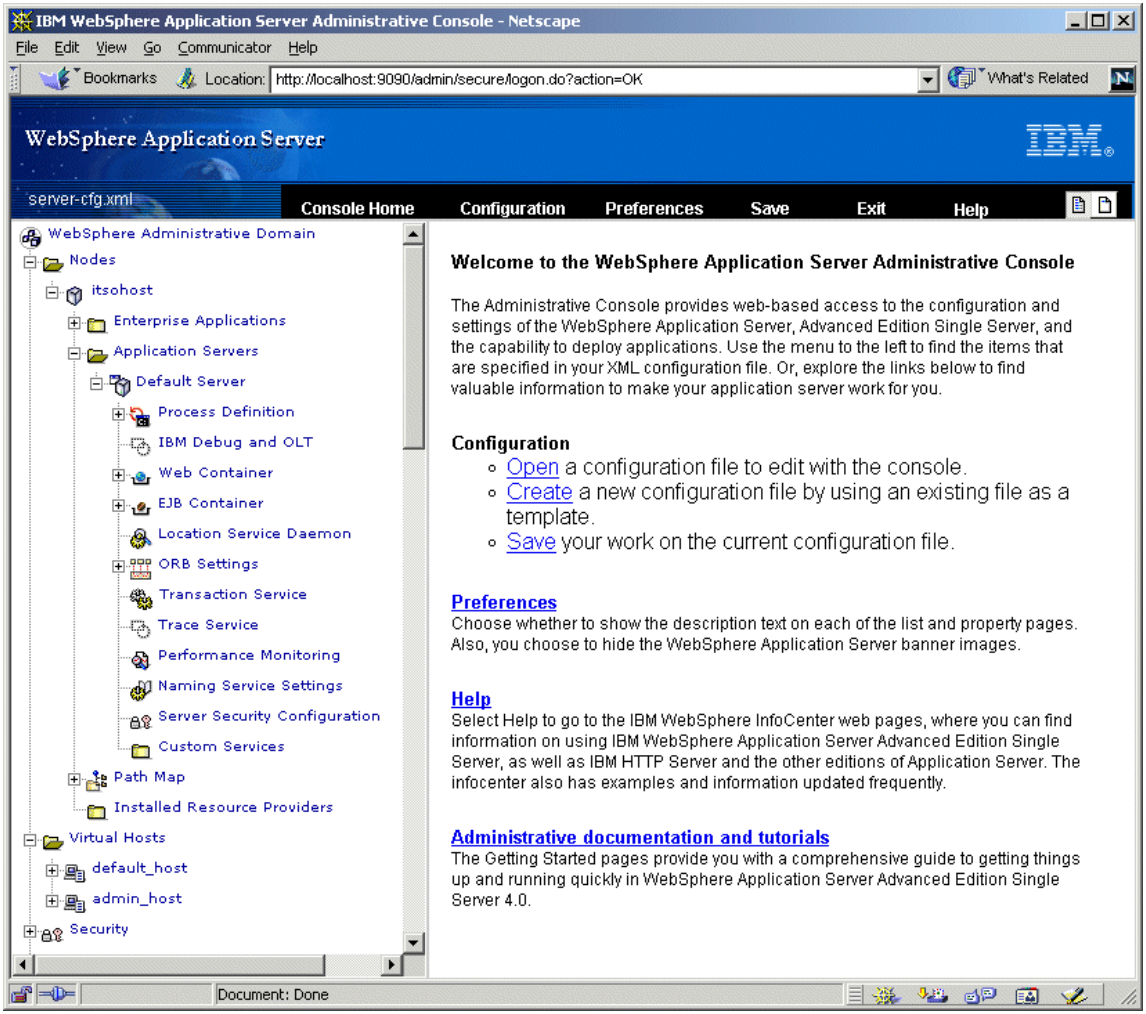


Figure 4-7 WebSphere V4.0 Advanced Edition Single Server Web administrative console

The configuration that gets loaded by default is contained in the server-cfg.xml file, found in the <WAS_HOME>/config directory. A different configuration file may be loaded once the Web administrative console is open on your browser. However, any other chosen configuration file may be passed as a parameter in the browser URL so that it gets loaded on startup. The browser URL should be as follows:

```
http://localhost:9091/admin/edit?configFile=C:/temp/foo.xml
```

This will load foo.xml from the C:/temp directory.

Other than the brief overview here, we don't use the Web administration console in this book. It is not available for WebSphere Application Server V4.0, Advanced Edition, as the Web browser interface does not provide the functional richness needed to manage distributed, scalable, and available enterprise applications.

4.12.3 WebSphere Control Program

The WebSphere Control Program is a command-line administrative tool that is available in the Advanced Edition. It also operates in an interactive mode. You can use the WebSphere Control Program to administer domain resources such as defining, configuring, managing, importing and exporting configurations, and performing diagnostic operations. It is based on Tcl.

Tcl stands for Tool Command Language. It is open source and there is also a Java version of Tcl called Java Control Language (JACL). The Tcl language has a simple and programmable syntax, and can be used stand-alone or in embedded applications. It is extensible and WebSphere Control Program extends Tcl by providing a set of commands for manipulating WebSphere objects.

WebSphere Control Program is started from the command line, by running the wscp.bat batch file on the Windows platform or the wscp.sh shell script on the UNIX platform. These files are found in the <WAS_HOME>/bin directory.

Details about WebSphere Control Program can be found in Chapter 23, "Command-line administration and scripting" on page 881.

4.12.4 XMLConfig

XMLConfig is another command-line administrative tool available with WebSphere Advanced Edition. The XMLConfig tool offers the WebSphere administrator the ability to import and export (full or partial export) the WebSphere configuration data in the administrative repository. You can use this

tool to make multiple changes to the administration repository without having to manually repeat the changes using the administrative console. You can also use it to generate a new plug-in configuration file with the new generatePluginCfg option.

XMLConfig is not an interactive tool and cannot be used to retrieve status information from WebSphere.

XMLConfig is described in detail in Chapter 23, “Command-line administration and scripting” on page 881.

4.12.5 DrAdmin

The DrAdmin command tool is available in all editions of WebSphere Application Server and is primarily used for troubleshooting. Located in the <WAS_HOME>/bin directory, it can be used to diagnose problems when other tools fail.

You can learn more about DrAdmin in Chapter 24, “Troubleshooting” on page 949.

5

Topologies selection

This chapter describes various topologies which IBM WebSphere Application Server V4.0 supports. In addition, options regarding the use of multiple clones, multiple WebSphere domains, vertical and horizontal scaling as well as multiple tiers (separating the Web and EJB containers) are covered. For detailed information including step-by-step configuration, see *IBM WebSphere V4.0 Advanced Edition Scalability*, SG24-6192.

5.1 Topology selection criteria

While a variety of factors come into play when considering the appropriate topology for a WebSphere deployment, the primary factors to plan for typically include:

- Security
- Performance
- Throughput
- Availability
- Maintainability
- Session state

5.1.1 Security

Security concerns usually require physical separation of the HTTP (Web) server from the application server processes, typically across one or more firewalls.

EJB security reminder: Enterprise beans and their clones have separate identities. Therefore, you must explicitly protect each and every bean by configuring resource security for the bean and including it in a secured enterprise application.

5.1.2 Performance

Performance involves minimizing the response time for a given transaction load. While a number of factors relating to the application design can affect this, adding additional resources in the following two ways, or a combination of both, can be used to good effect:

- Vertical scaling, which involves creating additional application server processes on a single physical machine in order to provide multiple thread pools, each corresponding to the JVM associated with each application server process.
- Horizontal scaling, which involves creating additional application server processes across multiple physical machines.

5.1.3 Throughput

Throughput, while related to performance, more precisely involves the creation of some number of application server instances (clones) in order to increase the number of concurrent transactions that can be accommodated. As with performance, the application server instances can be added through vertical and/or horizontal scaling.

5.1.4 Availability

Availability requires that the topology provide some degree of process redundancy in order to eliminate single points of failure. While vertical scalability can provide this by creating multiple processes, the physical machine then becomes a single point of failure. For this reason a high-availability topology typically involves horizontal scaling across multiple machines.

Hardware-based high availability

By providing both vertical and horizontal scalability the WebSphere Application Server runtime architecture eliminates a given application server process as a single point of failure. In fact the only single point of failure in the WebSphere runtime is the database server where the WebSphere administrative repository resides. It is on the database server that any hardware-based high availability (HA) solutions such as HACMP, Sun Cluster, or MC/ServiceGuard should be configured.

In most cases there is very little to be gained from trying to configure WebSphere Advanced Edition to work in conjunction with a hardware-based HA product. The only case where a hardware-based HA solution would provide value is where WebSphere is serving as the coordinator of a distributed (two-phase commit) transaction. If a WebSphere node were to go down, then any in-doubt transaction (after prepare, before commit) could not be resolved automatically until the node was restored to service. So unless one is utilizing WebSphere for distributed transactions, the cluster capabilities inherent in WebSphere should prove sufficient for the WebSphere runtime proper.

At this writing use of a hardware-based HA solution is not supported in WebSphere V4.x, but testing is underway to provide support for HACMP with DB2/Oracle, Sun Clustering with Oracle and DB2, HP Service Guard with DB2 and Oracle, and MS Clustering with SQLServer. Look for support for these products in the near future with WebSphere 4.x.

5.1.5 Maintainability

While maintainability is somewhat related to availability, there are specific issues that need to be considered when deploying a topology that is maintainable. In fact some maintainability factors are at cross purposes to availability. For instance, ease of maintainability would dictate that one minimize the number of application server instances in order to facilitate online software upgrades. Taken to the extreme, this would result in a single application server instance, which of course would not provide a high availability solution. In many cases it is also possible that a single application server instance would not provide the required throughput or performance. In deciding on the degree of vertical and horizontal scaling that one needs to incorporate in a topology, you should also consider the matter of hardware upgrades (for example, adding CPUs, memory, or upgrading to faster CPUs). As we will see below, one alternative topology for maintainability involves creating more than one WebSphere domain.

5.1.6 Session state

Unless you have only a single application server or your application is completely stateless, then maintaining session state between HTTP client requests will also play a factor in determining your topology. In WebSphere V4.0 the recommended method for sharing of sessions between multiple application server processes (clones) is to persist the session to a database.

In WebSphere V4.0 a JMS/Messaging alternative for sharing of session state has been introduced as a technology preview. At present this technology is immature and should not be used for production.

Lastly, the configuration of an HTTP sprayer such as the Network Dispatcher component of WebSphere Edge Server needs to be considered when session state is important.

5.1.7 Topology selection summary

Table 5-1 is a summary of topology selection.

Table 5-1 Topology selection summary

	Security	**Perform-ance**	**Through-put**	**Maintain-ability**	**Availability**	**Session**
Vertical Clones		Limited benefit	Limited to resources on a single machine	Easiest to maintain	Process isolation	Required

	Security	Perform-ance	Through-put	Maintain-ability	Availability	Session
Horizontal Clones		Best in general	Best in general	Code migration to multiple nodes	Process and hardware redundancy	Required
HTTP Separate	Allow for firewalls/ DMZs	Usually better than local	Usually better than local			
Three Tiers	Most options for firewalls	Typically slower than single JVM	Additional clones may improve throughput			
One Domain				Ease of mainten-ance		
Multiple Domains				Harder to maintain than single domain	Process, hardware and software redundancy	

5.2 Vertical scaling with WebSphere workload management

In the simplest case, one can configure many application server clones on a single machine, and this single machine also runs the HTTP server process. This configuration is depicted in Figure 5-1.

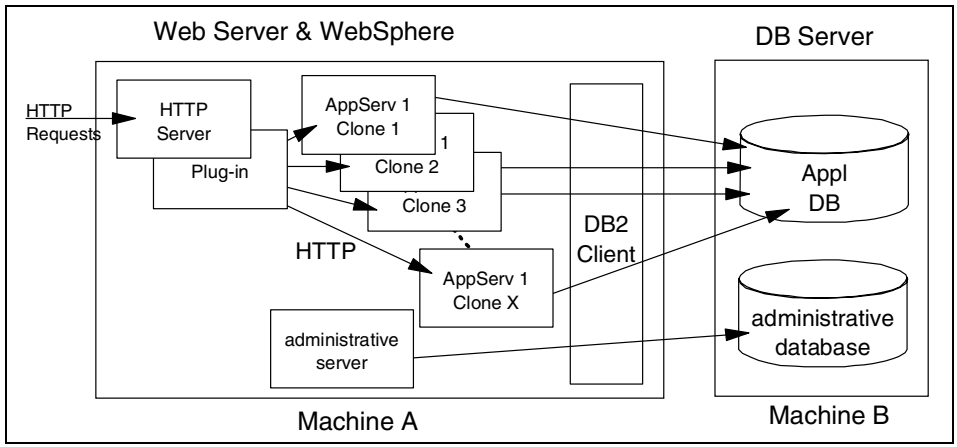


Figure 5-1 Vertical scaling with clones

At first glance this would appear to be the simplest to configure. However, as we'll see later on, separating the HTTP server(s) from the application server processes on separate physical machines is not significantly harder with WebSphere than the simple case depicted here. Though one is limited to the resources available on a single machine as well as the availability risk when using a single machine, this configuration does provide for process isolation. The improved multi-processor support of the WebSphere V4.0 JVM means that the thoughput or performance benefits of vertical scaling with clones is limited.

You will notice that even in the simple "single machine" configuration depicted above, the WebSphere configuration repository resides on a remote database server. There are several reasons why this represents a good practice.

First, most enterprises have already invested in a high availability solution for their database server, and the configuration repository represents a single point of failure in WebSphere, so it pays to make this highly available.

Secondly, the database that houses the configuration repository should be backed up on a regular basis, just as application data is. Housing the repository on the same server as the application data usually simplifies this task, since appropriate DBA procedures such as database backup processes are already defined for this machine.

Additionally, the database server is typically sized and tuned for database performance, which may differ from the optimal configuration for the application server (in fact on many UNIX servers, installing the database involves modification of the OS kernel).

Lastly, if both the database and application server are placed on the same machine, then under high-load you have two processes: the application server and the database server, competing for increasingly scarce resources (CPU and memory), so in general you can expect significantly better performance by separating the application server from the database server.

Note: WebSphere V4.0 provides two types of workload management (WLM):

- Plug-in WLM, which distributes servlet requests across Web containers
- EJS WLM, which distributes EJB requests across EJB containers

Refer to Chapter 17, “Server groups and workload management” on page 605 for further details.

5.3 HTTP server separation from the application server

The WebSphere V4.0 HTTP plug-in allows the HTTP server to be physically separated from the application server.

When compared to a configuration where the application server and the HTTP server are co-located on a single physical server, separation of the application server and the HTTP server can be utilized to provide varying degrees of improvement in:

- Performance
- Process isolation
- Security

Notes:

1. The WebSphere V4.0 Web server plug-in uses the HTTP transport between the HTTP server and application server. The OSE plug-in transport is not supported in WebSphere V4.0.
2. The servlet redirector alternatives for separating the HTTP server from the application server, available in WebSphere V3.x, are no longer supported in WebSphere V4.0. The unsupported servlet redirector alternatives include:
   - Thick Servlet Redirector
   - Thick Servlet Redirector administrative server agent
   - Thin Servlet Redirector
3. The reverse proxy approach for separating the HTTP server and the application server is generally not suggested for use with WebSphere V4.0. The new HTTP plug-in behaves very much like a reverse HTTP proxy, without its disadvantages. Reverse proxy disadvantages include:
   - Requires extra hardware and software
   - Not WebSphere workload management aware

The HTTP plug-in allows for physical separation of the HTTP server and the Web container(s). The HTTP plug-in supports clustering and workload management of application servers. This means that the HTTP server can send requests that require intensive processing to multiple application server machines, freeing up the HTTP server machine to process more requests. The HTTP plug-in provides for both vertical (as depicted in Figure 5-2) and horizontal scaling of the WebSphere environment.

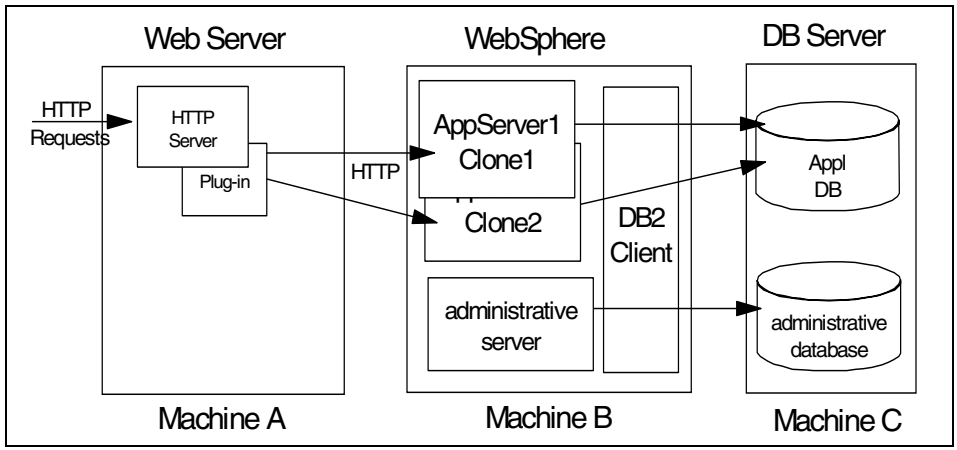


Figure 5-2 HTTP plug-in

The HTTP plug-in also supports data encryption between the HTTP server and the application server, using HTTP over SSL or HTTPS. The HTTP plug-in is suitable for environments that require all network communication be encrypted, even in a DMZ.

The WebSphere V4.0 HTTP plug-in also performs well. In tests it has been shown to be as much as 30 to 60 percent faster than the WebSphere V3.5 OSE plug-in. It performs well on a local or remote Web server. In some configurations it may perform better when remote, by separation of the HTTP and application server processes.

In summary the benefits of the HTTP plug-in are:

- Supports WebSphere workload management.
- Supports WebSphere security.
- Support for NAT firewalls.
- Does not need database access through a firewall.
- Communication can be encrypted.
- Performs better than OSE.
- Administration has been simplified. The HTTP plug-in uses one, easy-to-read XML configuration file, rather than the three files needed for the WebSphere V3.5 OSE plug-in.

While the disadvantages of the HTTP plug-in are:

- Manual configuration and administration of the HTTP server plug-in file when changes are made to the remote application server.

- No protocol shift for inbound and outbound traffic across a firewall. The following approaches can be used to address this disadvantage:
  - Use the HTTPS plug-in to provide a high-security connection between the HTTP server and application server.

    This connection can be configured so that the HTTPS plug-in and application server must mutually authenticate each other using public-key infrastructure (PKI).
  - Use different inbound (browser to HTTP server) and outbound (HTTP plug-in to application server) port numbers

Table 5-2 is a summary of WebSphere HTTP plug-in characteristics.

Table 5-2 WebSphere HTTP plug-in characteristics

Characteristic	Comment
SSL support	Yes
Needs DB access	No
Workload management	Yes
Network Address Translation	Yes
Performance	High
Administration	Manual with remote Web server

5.4 Scaling WebSphere in a three-tier environment

Partitioning your application server processes into servlet application servers and EJB application servers as depicted below can provide some advantages from a security perspective as well as some possible advantages from a performance perspective.

In this topology the EJB layer is closer to the application data, for which an entity EJB provides a representation. When running in an environment where two firewalls are employed, this allows one to provide the same level of security for entity EJBs as is provided for application data.

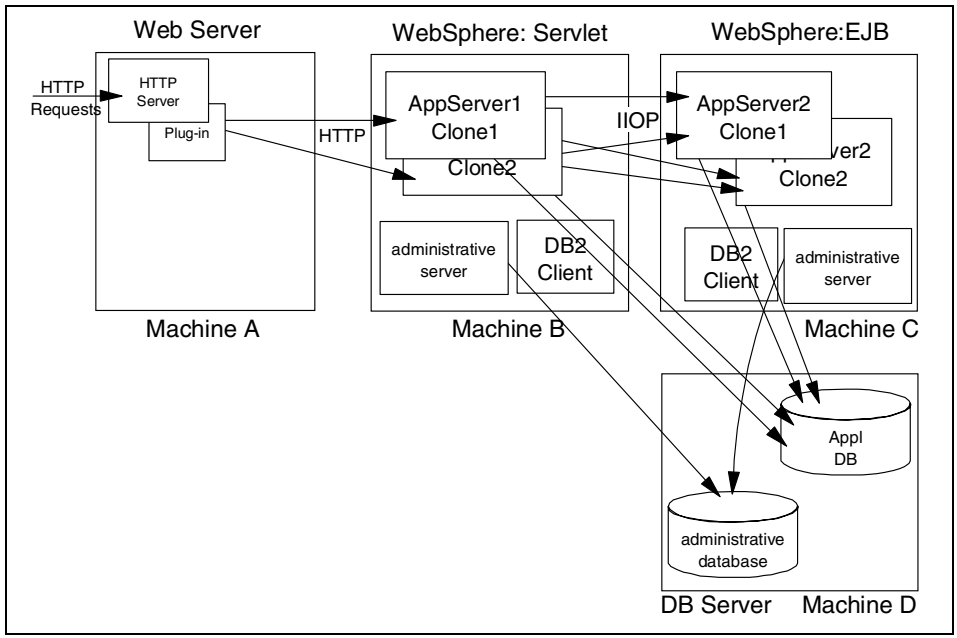


Figure 5-3 3-tier environment

In terms of performance this topology allows one to replicate different numbers of application servers (for example, two Web containers and five EJB servers or vice versa), which may provide better performance. In general, however, elimination of the local JVM optimizations that occur when both the servlet (client) and the EJB (server) are resident in the same application server (JVM) as well as the network latency introduced with the topology will tend to negate any possible performance improvement brought about by the additional processing power afforded by a separate physical server.

While providing more redundancy for application server processes, this topology also introduces more possible points of failure. In addition to more application server processes, this means that you'll have more to manage as well.

5.5 Horizontally scaling Web servers with WebSphere

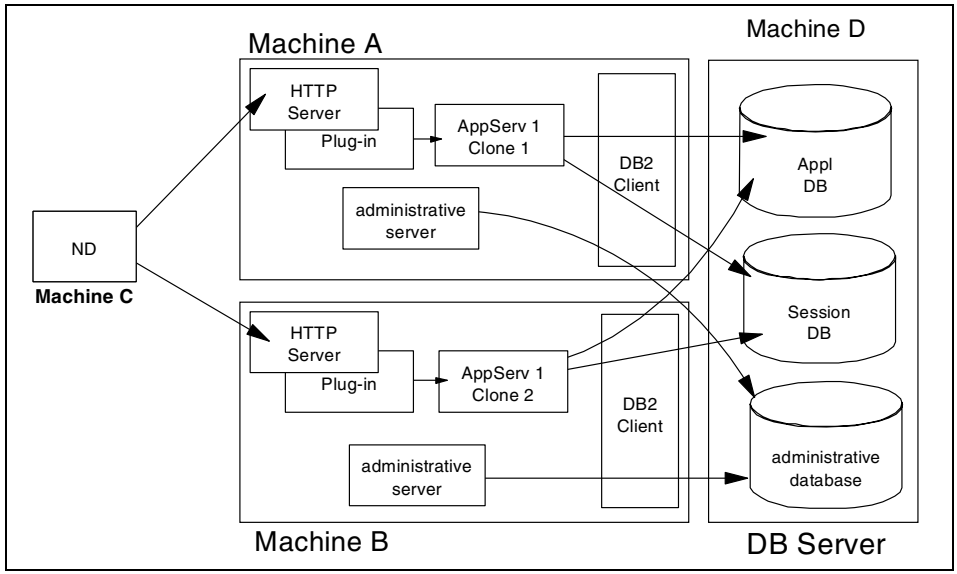


Figure 5-4 WebSphere with Network Dispatcher

Adding a mechanism for distributing HTTP requests, such as the Network Dispatcher component of WebSphere Edge Server as depicted above, provides the following advantages:

- Network Dispatcher allows for an increased number of connected users.
- Network Dispatcher eliminates the HTTP server as a single point of failure and can be used in combination with WebSphere workload management to eliminate the application server as a single point of failure.
- Increased throughput by virtue of adding multiple servers and CPUs to service the workload.

5.6 One WebSphere domain vs. many

While there are no “hard” limits on the number of nodes that can be clustered in a WebSphere domain, one may want to consider creating multiple WebSphere domains for a variety of reasons:

- Two (or more) domains can be employed to provide not only hardware failure isolation, but software isolation as well. This can come into play in a variety of situations:

  - Deployment of a new version of WebSphere. Note that nodes running WebSphere V4.0 and V3.x in the same domain are not supported.
  - Application of an e-fix or patch.
  - Rollout of a new application or revision to an existing application.
- In cases where an unforeseen problem occurs with the new software, multiple domains prevent a catastrophic total outage to an entire site. A roll-back to the previous software version can also be accomplished more quickly. Of course, multiple domains imply the software has to be deployed more than once, as would be the case with a single domain.
- Multiple smaller domains may provide better performance than a single large domain, since there will be less interprocess communication in a smaller domain.

Of course, multiple domains will require more effort for day-to-day operations, since administration must be performed on each domain, although this can be mitigated through the use of scripts employing WebSphere Control Program and XMLConfig. Multiple domains also mean multiple administrative databases, which means multiple backups here as well.

In order to distribute requests across multiple domains as depicted in Figure 5-6, you'll need a mechanism, such as Network Dispatcher, for spraying your HTTP requests across the HTTP servers associated with each domain.

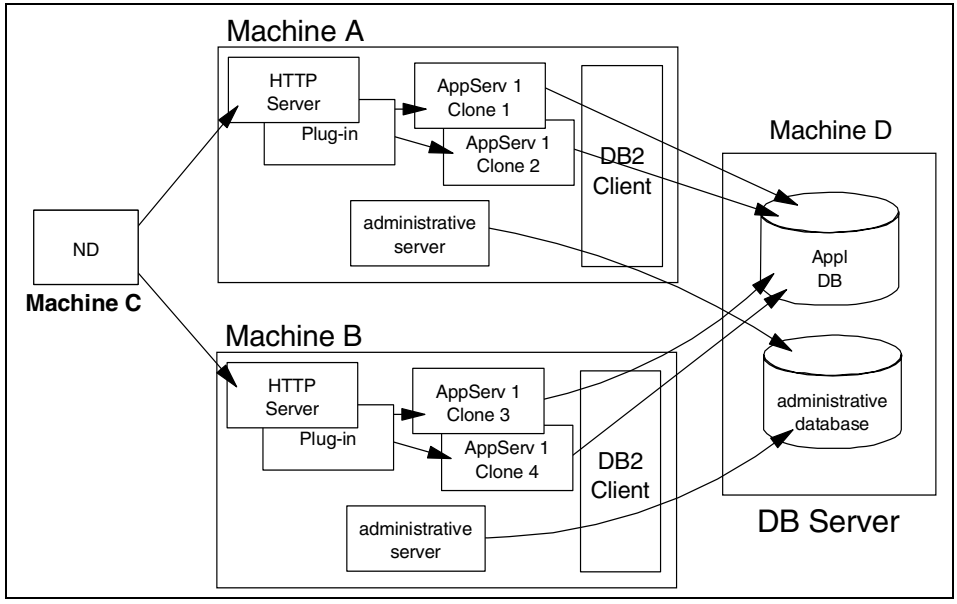


Figure 5-5 One WebSphere domain

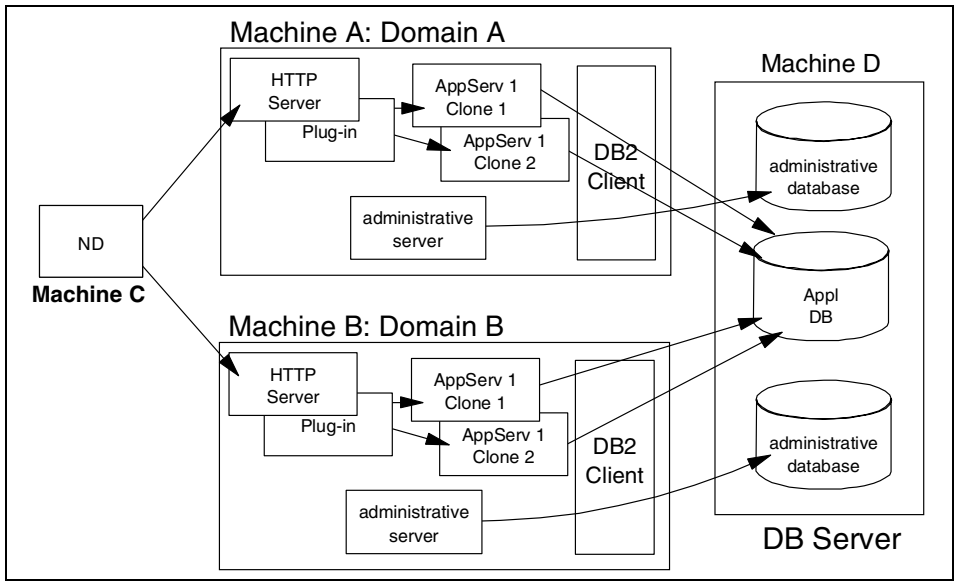


Figure 5-6 Multiple WebSphere domains

5.7 Multiple applications within one node vs. one application per node

When deciding how to deploy your application, one decision point is whether to deploy clones of an application server across all nodes in a cluster as depicted in Figure 5-7, or to place all clones of a given application server on a single node as depicted in Figure 5-8 on page 82.

As discussed previously, horizontal cloning, as depicted in Figure 5-7 provides:

- Process isolation
- Application software failover
- Hardware failover

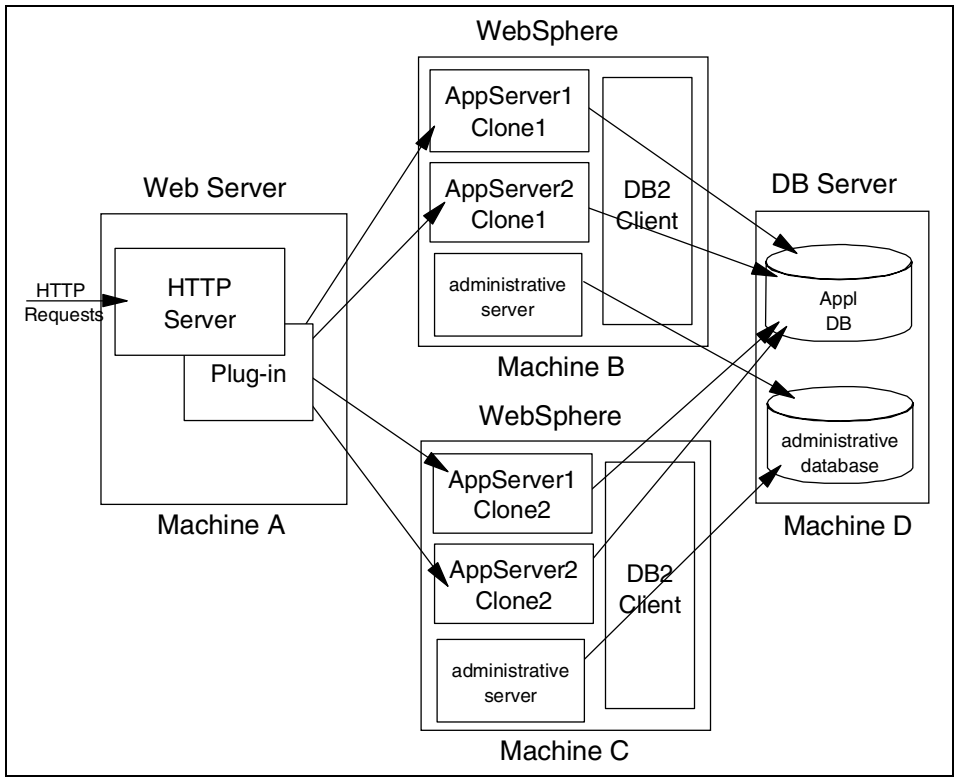


Figure 5-7 Multiple application servers within one server

As with horizontal cloning, vertical cloning as depicted in Figure 5-8 on page 82 provides for process isolation and application software failover, but obviously does not provide for any sort of hardware high availability. The primary advantage to vertical cloning is that the executables for a given application have to be distributed to only a single machine. Horizontal cloning on the other hand requires that your application executables be distributed across multiple machines in a cluster. Use of a file system, such as NFS or AFS, that provides a common file mount point for all nodes can ease the distribution of code across multiple nodes.

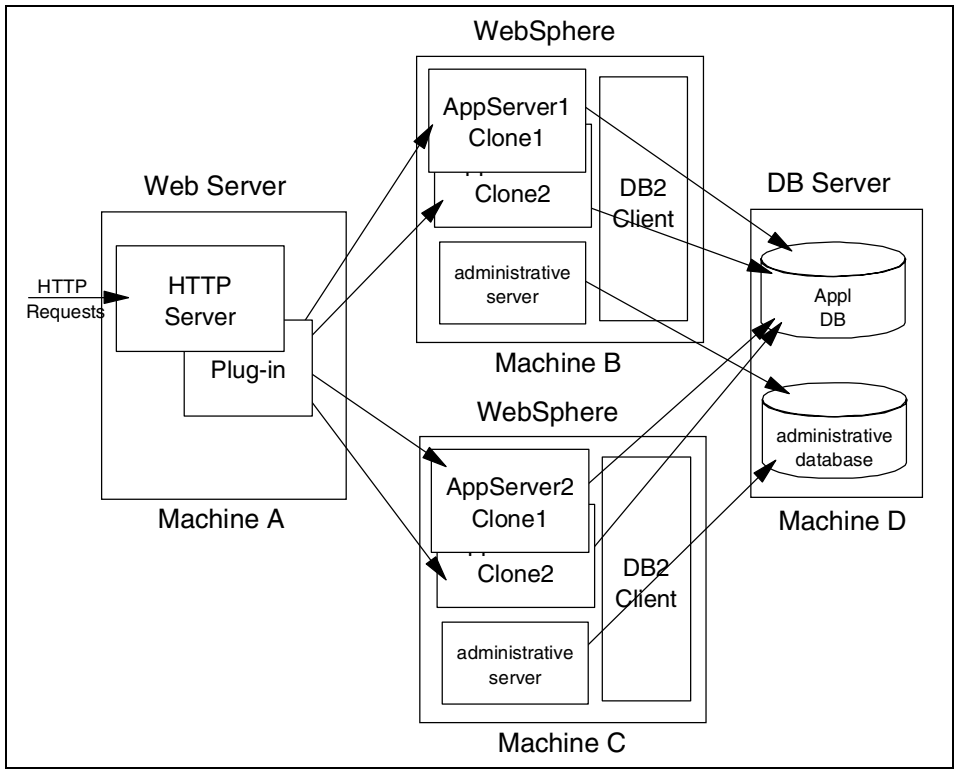


Figure 5-8 One application server per node

5.8 Closing thoughts on topologies

Whatever topology you decide on, a best practice is to partition your production acceptance environment exactly the same as your production environment. This avoids surprises when deploying your application into production.

Another consideration, when practical for your application architecture, is to create a number of smaller application servers, rather than a single large one.

This has at least two advantages:

- The plug-in configuration files can be smaller (less complexity of URIs), which leads to better startup performance and possibly better execution performance.
- At least during the development phase, it takes less time to cycle a smaller application server to pickup various configuration changes.

Of course, creation of multiple application servers in this manner should be carefully balanced against the increased complexity of doing so and the potential increase in response time due to inter-process RMI/IIOP calls and network latency.

6

Web services

WebSphere Application Server V4.0 provides support for a new breed of Web applications called Web services.

In this chapter we discuss:

- What are Web services?
- Web services architecture
- WebSphere V4.0 support for Web services
- Using Web services with WebSphere

6.1 What are Web services?

Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. A sample Web service might provide stock quotes or process credit card transactions. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the service.

Web services are independent of specific programming languages or operating systems. Instead, Web services rely on pre-existing transport technologies (such as HTTP) and standard data encoding techniques (such as XML) for their implementation.

The Web services approach to programming is based on the idea of building applications by discovering and invoking network-available applications to accomplish some task.

6.2 Web services architecture

Web services are deployed on the Web by *service providers.* The functions provided by the Web service are described using the Web Services Description Language (WSDL). Deployed services are published on the Web by service providers.

A *service broker* helps service providers and *service requestors* find each other. A service requestor uses the Universal Discovery Description and Integration (UDDI) API to ask the service broker about the services it needs. When the service broker returns the search results, the service requestor can use those results to bind to a particular service.

As we can see in Figure 6-1 on page 87:

- Web service descriptions can be created and published by service providers.
- Web services can be categorized and searched by specific service brokers.
- Web services can be located and invoked by service requesters.

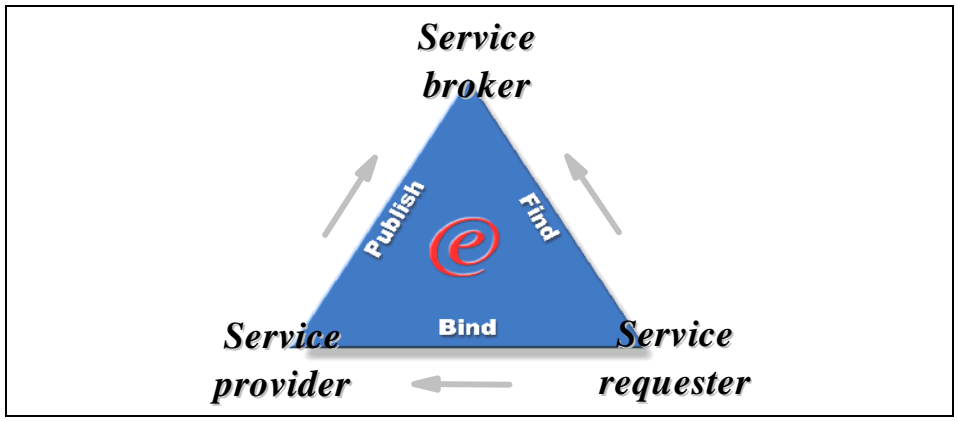


Figure 6-1 Web services components and operations

6.2.1 Web services components

Figure 6-1 shows the three main components used in Web services:

- **Service providers,** who provide services and maintain a registry that makes those services available.
- **Service brokers,** who match service providers with service requestors.
- **Service requesters,** who use service brokers to discover Web services, then invoke those services to create applications.

6.2.2 Web services operations

Web services components use three basic operations:

- **Publish/Unpublish** involves advertising services to a registry (publishing) or removing those entries (unpublishing). The service provider contacts the service broker to publish or unpublish a service.
- **Find** is performed by service requestors and service brokers together. The service requestors describe the kinds of services they're looking for, and the service brokers deliver the results that best match the request.
- **Bind** takes place between the service requestor and the service provider. The two parties negotiate as appropriate so the requestor can access and invoke services of the provider.

6.2.3 Web services implementation

The Web service architecture is implemented using the following open standards:

WSDL	Web Services Description Language
UDDI	Universal Discovery Description and Integration
SOAP	Simple Object Access Protocol
XML	eXtensible Markup Language

These standards allow Web applications to find each other and interact dynamically over the Web.

WSDL - Web Services Description Language

The Web Services Description Language (WSDL) is an XML-based interface definition language that provides a way to catalog and describe Web services. It is used to automate the details involved in applications communication. WSDL defines the:

- Web service interfaces, including:
  - Operation types (one-way, request-response, notification)
  - Messages defining a Web service
  - Data types (XML schema)
- Web service access protocol (SOAP over HTTP, for example)
- Web service contact endpoints (Web service URL, for example)

Compliant server applications must support these interfaces, and client users can learn from the document how a service should be accessed.

UDDI - Universal Discovery Description and Integration

Universal Discovery Description and Integration (UDDI) provides a way to find out about available Web services.

UDDI creates a global, platform-independent, open framework to enable businesses to:

- Discover each other
- Define how they interact over the Web
- Share information in a global registry

Three public UDDI registries exist on the Web, sponsored by IBM, Microsoft and HP. Registration is free and registration entries are replicated to other nodes.

The information provided in a UDDI business registration consists of three components:

- “White pages” including address, contact, and known identifiers
- “Yellow pages” including industrial categorizations based on standard taxonomies
- “Green pages” for the technical information about services that are exposed by the business

Web service providers and requesters use a SOAP API used to communicate with a UDDI registry.

SOAP - Simple Object Access Protocol

The Simple Object Access Protocol (SOAP) is a network-neutral, lightweight protocol for exchange of information between two remote applications.

It is an XML-based protocol that consists of three parts:

1. An envelope that defines a framework for describing what is in a message and how to process it
2. A set of encoding rules for expressing instances of application-defined data types
3. A convention for representing remote procedure calls and responses

Example 6-1 shows a sample SOAP request, and Example 6-2 on page 90 shows a sample SOAP response. These examples show the SOAP request from a client wanting a quote for IBM stock, and the SOAP response from the SOAP server providing stock quotes.

Example 6-1 SOAP request

```
POST /soapsamples/servlet/rpcrouter HTTP/1.0
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 460
SOAPAction: ""

<?xml version='1.0' encoding='UTF-8'?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <SOAP-ENV:Body>
      <ns1:getQuote xmlns:ns1="urn:xmltoday-delayed-quotes"
         SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <symbol xsi:type="xsd:string">IBM</symbol>
      </ns1:getQuote>
```

```
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The SOAP request indicates that getQuote, from the xmltoday-delayed-quotes namespace, should be invoked from `http://localhost/soapsamples/servlet/rpcrouter`. Upon receiving this request, the stock quote application at localhost executes the business logic that corresponds to getQuote.

The SOAP protocol does not specify how to process the request. The provider could run a CGI script, invoke a servlet, or perform any other process that generates the appropriate response.

The response comes in the form of an XML document that contains the results of the processing, in this case the quote for IBM stock.

Example 6-2 SOAP response

```
HTTP/1.1 200 OK
Server: IBM_HTTP_SERVER/1.3.19  Apache/1.3.20 (Win32)
Content-Length: 479
Connection: close
Content-Type: text/xml; charset=utf-8
Content-Language: en

<?xml version='1.0' encoding='UTF-8'?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <SOAP-ENV:Body>
      <ns1:getQuoteResponse xmlns:ns1="urn:xmltoday-delayed-quotes"
         SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <return xsi:type="xsd:float">108.53</return>
      </ns1:getQuoteResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The response does not include a SOAP-specified header. The results are placed in an element whose name matches the method name (getQuote) with the suffix, "Response" as in getQuoteResponse.

Note: You can access a SOAP Web service without using a UDDI registry, as long as you already have the Web service description and host name.

XML - eXtensible Markup Language

XML (eXtensible Markup Language) provides a common language for exchanging information.

6.2.4 Web services benefits

Web services technology will enable businesses to:

- Deliver new IT solutions faster and at lower cost by focusing their code development on core business, and using Web services applications for non-core business programming.
- Protect their investment in IT legacy systems by using Web services to wrap legacy software systems for integration with modern IT systems.
- Integrate their business processes with customers and partners at less cost. Web services make this integration feasible by allowing businesses to share processes without sharing technology. With lower costs, even small business will be able to participate in B2B integration.
- Enter new markets and widen their customer base. Web services listed in UDDI registries can be "discovered" and thus are "visible" to the entire Web community.

Web services promote interoperability

The interaction between a service provider and a service requester is designed to be completely platform and language independent. This interaction requires a WSDL document to define the interface and describe the service, along with a network protocol (usually HTTP). Because the service provider and the service requester have no idea what platforms or languages each other are using, interoperability is a given.

Web services enable just-in-time integration

As service requesters use service brokers to find service providers, the discovery takes place dynamically. Once the requester and provider have found each other, the provider's WSDL document is used to bind the requester and the service together. This means that requesters, providers, and brokers work together to create systems that are self-configuring, adaptive, and robust.

Web services reduce complexity through encapsulation

Service requesters and providers concern themselves with the interfaces necessary to interact with each other. As a result, a service requester has no idea how a service provider implements its service, and a service provider has no idea how a service requester uses its service. Those details are encapsulated inside the requesters and providers. That encapsulation is crucial for reducing complexity.

Web services give new life to legacy applications

It is relatively straightforward to take an application, generate a SOAP wrapper, then generate a WSDL document to cast the application as a Web service. This means that legacy applications can be used in interesting new ways. In addition, the infrastructure associated with legacy applications (security, directory services, transactions, and so on) can be "wrapped" as a set of services.

6.3 WebSphere V4.0 support for Web services

Figure 6-2 on page 93 shows the WebSphere Application Server V4.0 components that provide support for Web services.

Integration of SOAP support in WebSphere V4.0 provides both SOAP server and client application environments. It enables WebSphere applications to send and receive SOAP messages, and leverage WSDL.

Integration of UDDI4J provides a Java interface to UDDI registries. This enables WebSphere applications to communicate with UDDI-compliant registries to publish and find Web services.

Web services deployed on the WebSphere platform can utilize platform strengths such as security, transaction monitoring, and trace/debug functions.

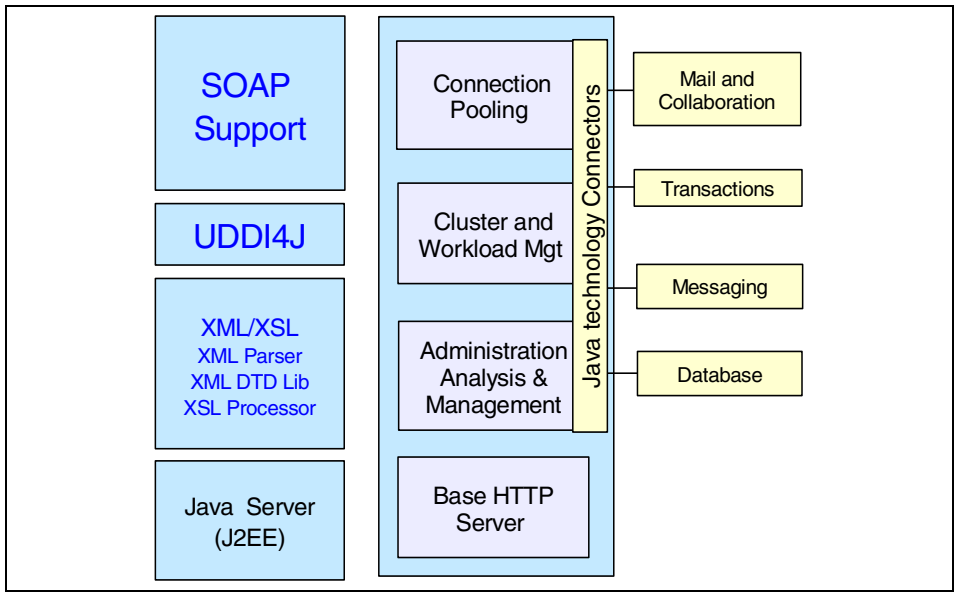


Figure 6-2 WebSphere SOAP components

6.3.1 WebSphere SOAP support

Apache SOAP Version 2.2 is integrated into WebSphere Application Server V4.0. Apache SOAP Version 2.2 is a Java-based implementation of the SOAP 1.1 specification with support for SOAP with attachments. SOAP with attachments allows binary data, such as images, to be passed with a SOAP message.

WebSphere Application Server V4.0 allows you to expose the following artifacts as SOAP services:

- Standard Java classes
- Enterprise JavaBeans
- Bean Scripting Framework (BSF) supported scripts
- DB2 stored procedures

WebSphere provides tools to assist you with packaging and deploying these artifacts as Web services. See Chapter 20, “Packaging and deploying Web services” on page 721 for more information.

When deploying your Web services in WebSphere, you can choose to enable the XML-SOAP Admin tool, which allows you to manage your SOAP-enabled services.

WebSphere Application Server also contains an implementation of the security extensions for SOAP. These security extensions provide secure connections and enable digitally signed messages. See the InfoCenter for more information.

Although Apache SOAP allows for SOAP over SMTP, WebSphere Application Server supports SOAP over HTTP only.

6.3.2 WebSphere UDDI support

UDDI4J is incorporated in WebSphere Application Server V4.0. UDDI4J is an open-source Java class library that provides an API to interact with a UDDI (Universal Description, Discovery and Integration) registry. UDDI4J contains an implementation of the client side of UDDI (everything your application needs to publish, find, and bind a Web service). It also includes the source code, and the complete Javadoc for the APIs.

WebSphere Application Server V4.0 does not provide a private UDDI registry. If you are interested in implementing a private UDDI registry you can download the IBM WebSphere UDDI registry preview from the WebSphere Developer Domain:

`http://www7b.boulder.ibm.com/wsdd/downloads/UDDIregistry.html`

6.3.3 WebSphere XML support

WebSphere Application Server V4.0 ships with Apache XML4J Version 3.1 XML (Xerces Version 1.2.1). This is a JAXP compatible, namespace-aware XML parser, as required for Apache SOAP.

6.3.4 WebSphere SOAP EAR Enabler tool

Enabling an application EAR file to use the SOAP environment in WebSphere takes an additional step.

The SOAPEarEnabler script in the WebSphere <WAS_HOME>\bin directory takes the EAR file and the SOAP deployment descriptor file, and creates a SOAP-enabled EAR file to install in WebSphere.

The SOAP deployment descriptor describes the service provided by the Web services application. The deployment descriptor contains:

- The identifier used by the Web service requesters
- The operations available from the service
- The class that implements the Web service

6.3.5 WebSphere Studio Application Developer

WebSphere Studio Application Developer provides the following tools to assist with Web services development:

- **Discover**. Browse the UDDI registry to locate existing Web services for integration. The Web becomes an extension of WebSphere Studio.
- **Create or Transform**. Create Web services from existing artifacts, such as Java beans, URLs that take and return data, DB2 XML Extender calls, DB2 stored procedures, and SQL queries.
- **Build**. Wrap existing artifacts as SOAP and HTTP GET/POST accessible services and describe them in WSDL. The Web services wizards assist you in generating a SOAP proxy to Web services described in WSDL and in generating Java bean skeletons from WSDL.
- **Deploy**. Deploy Web services into the WebSphere Application Server test environment using Server Tools.
- **Test**. Test Web services running locally or remotely in order to get instant feedback.
- **Develop**. Generate sample applications to assist you in creating your own Web service client application.
- **Publish**. Publish Web services to the UDDI registry, advertising your Web services so that other businesses can access them.

For more information, see the *Web Services Wizardry with WebSphere Studio Application Developer*, SG24-6292 redbook, or the WebSphere Studio Application Developer help view on application development documentation.

Note: WebSphere Application Server does not provide tools for generating WSDL files.You need to use a tool such as WebSphere Studio Application Developer, or you can generate WSDL files manually.

6.4 Using Web services with WebSphere

In this section we give an overview of how a Web service provider can build a Web service for WebSphere, and how a Web service requester can access a Web service using WebSphere Application Server.

6.4.1 Building Web services for WebSphere

Figure 6-3 on page 97 shows the process used by a service provider to build and publish a Java-based Web service for WebSphere. The basic steps are:

1. A Web service provider builds an artifact (a Java bean, for example) that contains the business logic and code to access the required back-end systems to determine the current stock price for a company. In this example, the back-end access is another Web application.
2. The service bean representing the Web service is packaged as a J2EE enterprise application archive (EAR) file for deployment to the WebSphere Application Server.
3. The EAR file is deployed to the application server so the SOAP server servlet can find the class and the methods in response to incoming SOAP messages requesting this service.
4. Once the artifact has been developed that represents the logic of the Web service, the developer can create a Web Services Description Language (WSDL) description of the service. WSDL describes the Uniform Resource Name (URN) where the Web service will be stored, and any methods and parameters the Web service exposes. WebSphere Studio Application Developer wizards produce WSDL from Java beans.
5. The description of the Web service (WSDL and other details) is then published to the UDDI registry so others can find this service. The UDDI registry is a global registry on the Internet that service requesters can query to find available services.
6. Service requesters can then find the Web service in the UDDI registry.
7. Service requesters can bind to and invoke the Web service.

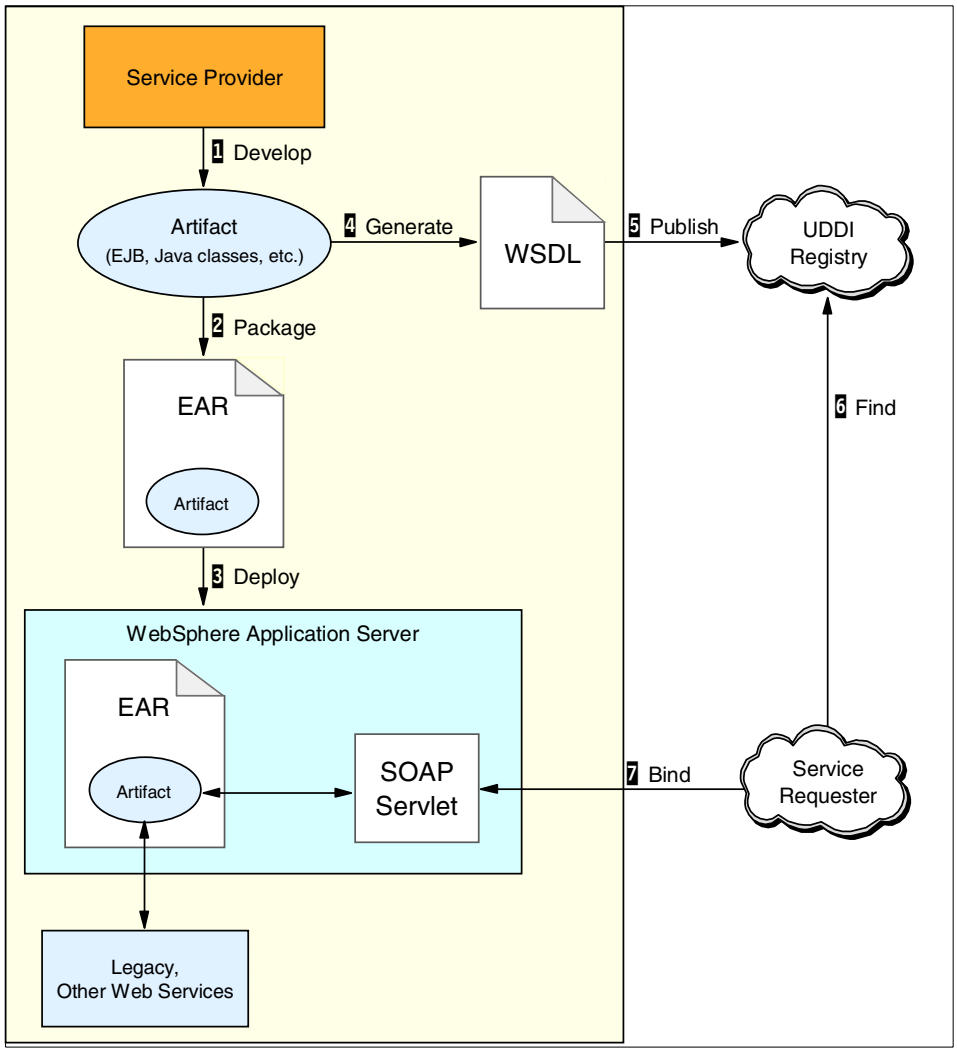


Figure 6-3 Building a WebSphere Web service

6.4.2 Accessing Web services from Web applications

Figure 6-4 on page 98 shows the process followed by a service requester to use a Web service from a WebSphere application. The basic steps are:

1. The Web application developer begins working on a new application.
2. The Web developer realizes that some parts of the main Web application are complex but well-defined entities (such as a stock quote or text translation service). Why develop them from scratch? Perhaps they already exist on the

Web as Web services? The Web developer can search a UDDI registry for an existing Web service that could be used in the Web application.

3. Web developer locates the required Web service and downloads WSDL describing how to bind and interact with the Web service.
4. Web developer generates a Java client proxy from the WSDL using a WebSphere Studio Application Developer wizard. The Java client proxy contains all the methods needed to interact with the Web service. The Java client proxy takes care of all the details, such as formatting the Java method calls into SOAP messages, serializing the method call parameters, connecting with the Web service URN, and receiving responses from the Web service.
5. Web developer codes the calls to the Java client proxy methods in the Web application.
6. Web developer tests the Web application. As the Web application runs, it calls Java client proxy methods.
7. The Java client proxy in turn calls the Web service methods via SOAP over HTTP and returns the results back to the Web application.

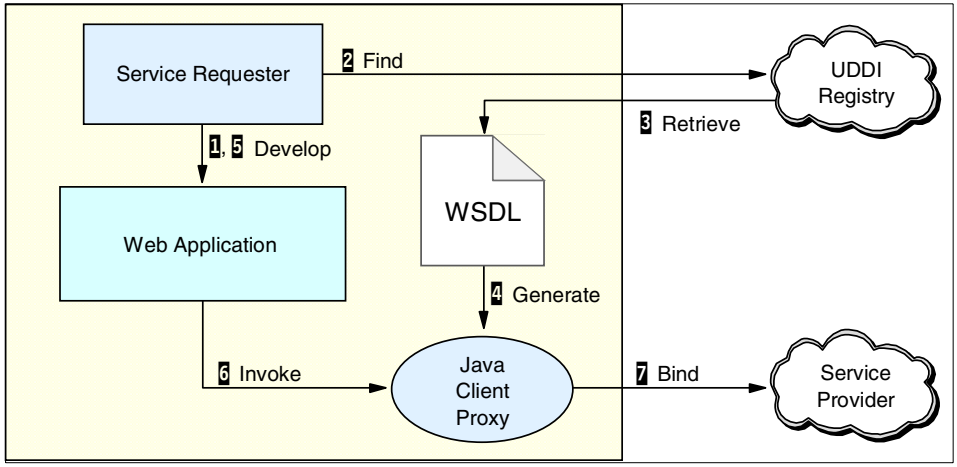


Figure 6-4 Accessing a Web service from WebSphere

6.4.3 WebSphere Web services runtime

We have seen how a service provider can build and publish a Web service, and how a service requester can use a Web service. Now let's look at the Web services runtime environment provided by WebSphere, from a service requester to a service provider, as shown in Figure 6-5 on page 99.

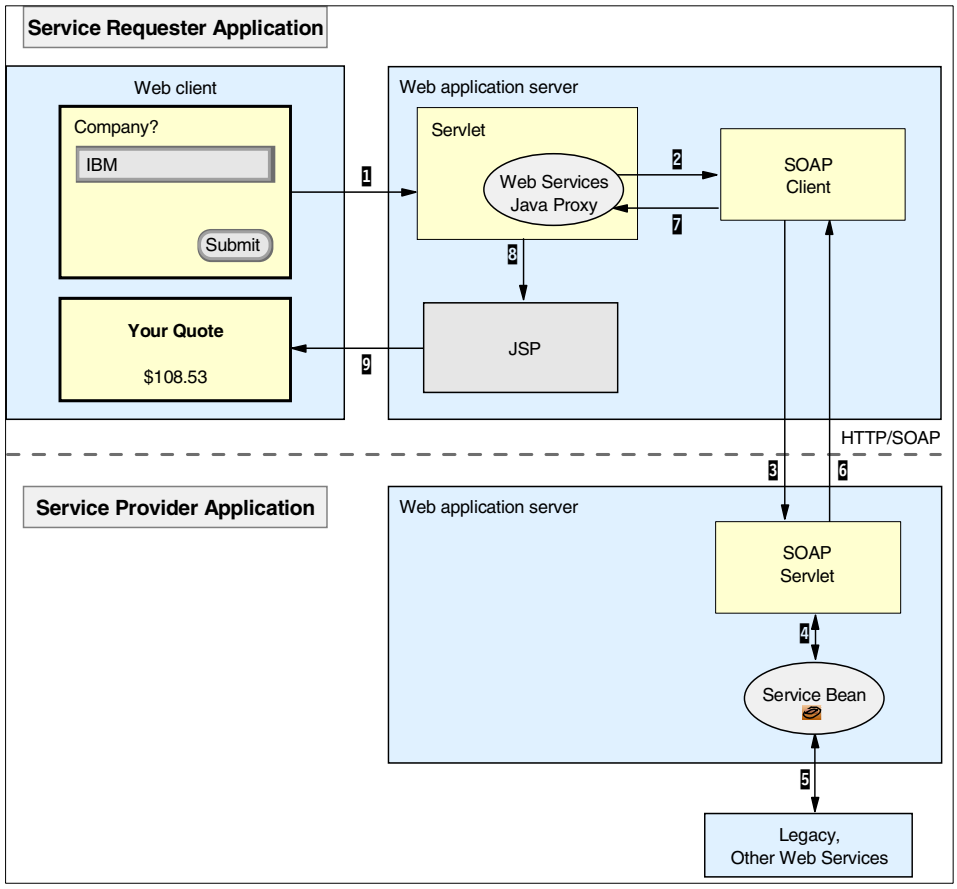


Figure 6-5 WebSphere Web services runtime view

In our example, a service requester has built a servlet/JSP-based Web application that will use the stock quote Web service provided by a service provider.

1. The user enters the company name (such as IBM), and clicks the Submit button in an HTML form. The request is routed to the application server which in turn invokes a Java servlet.
2. The servlet invokes Java proxy stock quote methods locally. The Java proxy calls the SOAP client running in the application server.
3. The SOAP client will serialize the method to be invoked on the remote server (Web service method) and the method's parameters. Then it will post an HTTP request with the SOAP message. When the HTTP request arrives at the application server of the service provider, the application server forwards it to the SOAP servlet.

4. The SOAP servlet de-serializes the method and its parameters, instantiates the service bean, and invokes on it the Web service method that contains the Web service functionality.
5. If necessary, the appropriate back-end access to a legacy system or another Web service occurs. As the usage of Web services increases, you may find that service providers also become service requesters for other Web services. For example, a stock price Web service would be called by a financial news feed Web service.
6. The SOAP servlet returns the results to the service requester SOAP client.
7. The SOAP client returns the results to the Java proxy.
8. The servlet places the results in a data bean and dispatches a JSP that uses this data bean to display the result of the stock quote query.
9. The JSP sends HTML back to the user containing the results of the Web service.

6.5 Summary

Web services is not a revolutionary "start from scratch" technology. It is built from available, standards-based Web technologies, and the entry costs to start using Web services are low. Web services can easily interact with the legacy technologies without having to change them. The value they bring to businesses and to Web application developers is significant and far reaching. It has been said that the potential worth of business transactions conducted using Web services technologies will be worth over $15 billion by 2004.

6.6 More information

These Web sites are also relevant as further information sources:

- *Web Services Wizardry with WebSphere Studio Application Developer*, SG24-6292

  `http://www.redbooks.ibm.com`

- developerWorks - Web Services Zone

  IBM sponsored site that contains articles, tutorials and latest news related to Web services.

  `http://www.ibm.com/developerworks/webservices`

- AlphaWorks - Web Services Toolkit

IBM sponsored site to provide early adopters access to emerging "alpha-code" technology. The Web Services Toolkit exploits new technologies that may be adopted in future releases of IBM Web Services Tools.

http://www.alphaworks.ibm.com/tech/webservicestoolkit

- XMethods

  An organization dedicated to promoting the development, deployment, and use of Web services.

  http://xmethods.com

- UDDI Organization

  UDDI Project and Community Web site contains UDDI specification, white papers, and FAQs. IBM is a member.

  http://www.uddi.org

- uddi4j Project

  UDDI4J is an open-source Java class library that provides an API to interact with a UDDI registry.

  http://oss.software.ibm.com/developerworks/projects/uddi4j/

- SOAP

  The latest version of the SOAP specification hosted by the W3C Organization Web site.

  www.w3.org/TR/SOAP

  Implementation details on Apache SOAP

  http://xml.apache.org/soap/

- WSDL

  The latest version of the WSDL specification hosted by the W3C Organization Web site.

  http://www.w3.org/TR/wsdl

7

Security

This chapter aims to provide an understanding of the WebSphere Application Server V4.0, Advanced Edition security concepts and components used to secure enterprise applications. We focus on the Java 2 Platform, Enterprise Edition (J2EE) security model and describe how it is supported in WebSphere V4.0.

The chapter is organized into the following sections:

- What's new in WebSphere V4.0 Advanced Edition security
- WebSphere security features
- WebSphere security model
- Performance considerations

You can find details on actually setting up WebSphere security for a Web application and for the WebSphere administrative console in Chapter 21, "Configuring security" on page 739.

For further information on WebSphere security see also the redbook, *IBM WebSphere Advanced Edition: Security*, SG24-6520.

7.1 What's new in WebSphere V4.0 security

The following improvements have been made to security in WebSphere V4.0:

- Support for the J2EE security model.

  See 7.3.1, "J2EE security model" on page 107 for details.

- Custom user registry support.

  See "User registry" on page 111 for details.

- HTTP and HTTPS transport support.

  See Chapter 14, "Configuring the Web server interface" on page 481 for further details.

- Encoding of password settings.

  See 7.2.6, "Password encoding in configuration files" on page 106 for details.

- Securing only selected components.

  See 7.4, "Performance considerations" on page 123 for details.

- The "SSOToken" has been deprecated.

  Now, only the "LTPA Token" is used to support single sign-on (SSO).

7.2 WebSphere security features

The IBM WebSphere Application Server V4.0 security system provides a number of features that you can use to secure your applications, including the following:

- Authentication policies and services
- Authorization policies and services
- Delegation policies
- Trust policies
- Single sign-on support
- Password encoding in configuration files

7.2.1 Authentication policies and services

Authentication is the process of verifying that a user (or process) really is who they say they are. This is usually done with some sort of user ID and password lookup scheme, or a certificate. You can indicate how you want WebSphere Application Server to verify the identity of users who try to access your resources. You can choose a supported directory service, the operating system registry, or a custom registry to verify the identity of users and groups.

7.2.2 Authorization policies and services

Authorization is the process of determining if a user has rights to use a secured resource in some way, for example the right to invoke a method on an EJB or access a particular HTML page, servlet or JSP. You can specify policies that give different users differing levels of access to your resources. If you define authorization policies, WebSphere Application Server will enforce them for you.

7.2.3 Delegation policies

Delegation allows an intermediary to do work initiated by a client under an identity based on the associated delegation policy. Therefore, enforcement of delegation policies affect the identity under which the intermediary performs downstream calls made to complete the current request.

When making downstream requests, the intermediary uses the client's credentials by default, but other choices are also possible. The result is that the downstream resources do not know the identity of the intermediary; they see the identity under which the intermediary is operating.

There are three possibilities for the identity under which the intermediary operates when making the downstream requests:

- The client's identity (default)
- Its own identity
- An identity specified by configuration

7.2.4 Trust policies

Decisions on who or what to trust also help make up an application security policy. Ultimately, in a security policy, something must be judged to be trustworthy, be it a user registry that contains user names and passwords or a Certificate Authority that issues certificates.

7.2.5 Single sign-on support

WebSphere Application Server supports third-party authentication, a mechanism for achieving single sign-on across the Internet domain that contains your resources. You can use single sign-on to allow users to log on once per session rather than requiring them to log on to each resource or application separately.

7.2.6 Password encoding in configuration files

Several of the WebSphere configuration files contain user IDs and passwords. These are needed at run time to access external secure resources such as the administrative database:

```
com.ibm.ejs.sm.adminServer.dbuser=db2inst1
com.ibm.ejs.sm.adminServer.dbpassword={xor}Oz1tNjEsK24=
```

Passwords are encoded, not encrypted, to deter casual observation of sensitive information. Password encoding combined with proper operating system file system security is intended to protect the passwords stored in these files.

To modify a password, edit the properties file in clear text and the password will be automatically encoded the next time WebSphere starts.

7.3 WebSphere security model

The WebSphere V4.0 security model has been modified to fit J2EE requirements, as defined in the EJB 1.1 and servlet 2.2 specifications. J2EE introduces the notion of security roles to encapsulate the grouping of method permissions. With J2EE, the EJB 1.1 specification also has a whole section on security (section 15), whereas EJB 1.0 addressed security only in a limited way.

In this section we look at the following topics:

- J2EE security model
- WebSphere security components
- WebSphere security administration model
- WebSphere authentication model
- WebSphere authorization model
- WebSphere delegation model
- WebSphere security and the operating environment

7.3.1 J2EE security model

An important goal of the J2EE security model is to provide policy-driven security that reduces the burden on the application programmer. In this model the security policy can be specified by mapping security roles to method permissions when the application is packaged by the application assembler. The application deployer then grants specific users access to the required roles at deployment time.

Flexibility improves because security policy is not hard coded. Applications are more portable between J2EE containers using different security mechanisms, since applications can be written without knowledge of the target security domain.

The J2EE security model uses two forms of container-based security:

- **Declarative security** expresses an application's security structure, including roles, access control, and authentication requirements in a form external to the application. A deployment descriptor is used to specify the declarative application security.
- **Programmatic security** is used when declarative security alone cannot express the security model of the application. Code needs to be written to specify programmatic application security.

The J2EE security model also uses delegation to allow requests from trusted clients to be delegated to more qualified EJB roles.

Authentication and authorization requirements

The J2EE security specification defines Web client authorization requirements that include:

- Support for login sessions
- Support for Web single sign-on

  J2EE single sign-on means that a user should have to authenticate only when crossing a security policy domain boundary.
- Support for the following login mechanisms:
  - HTTP Basic authentication
  - SSL mutual authentication
  - Form-based login

  We look at WebSphere support for these authentication methods in 7.3.4, "WebSphere authentication model" on page 116.
- Unauthenticated access

Web containers report that a user has not authenticated by returning null from the HttpServletRequest method getUserPrincipal.

For EJB containers, the getCallerPrincipal must never return null. When a call is made by a servlet to a bean, the container must be able to provide a principal. The deployer or system administrator selects which principal to use.

Security management roles

An important feature of the J2EE security model is that it encourages the use of defined security management roles and responsibilities during the application development life cycle. These security management roles are:

- Bean Provider
- Application Assembler
- Deployer

From 15.2, “Bean Provider's Responsibilities” in the Enterprise JavaBeans Specification, V1.1:

> *The Bean Provider is responsible for declaring in the security-role-ref elements of the deployment descriptor all the security role names used in the enterprise bean code.*

From 15.3, “Application Assembler's Responsibilities” in the Enterprise JavaBeans Specification, V1.1:

> *The Application Assembler (which could be the same party as the Bean Provider) may define a security view of the enterprise beans contained in the ejb-jar file...*
>
> *If the Bean Provider has declared any security role references using the security-role-ref elements, the Application Assembler must link all the security role references listed in the security-role-ref elements to the security roles defined in the security-role elements...*
>
> *The Applications Assembler defines method permissions for each security role.*

From 15.4, “Deployer's Responsibilities” in the Enterprise JavaBeans Specification, V1.1:

> *The Deployer is responsible for ensuring that an assembled application is secure after it has been deployed in the target operational environment...*
>
> *The Deployer’s job is to map the security view that was specified by the Application Assembler to the mechanisms and policies used by the security domain in the target operational environment.*

As an example, two different developers may write an *Account* bean and a *Transfer* bean. The Transfer bean module developer may decide to reference "Manager" and "Client" roles in code, and the Account bean module developer may use "Supervisor" and "User".

The Application Assembler then needs to combine the modules in an encompassing application. Security policy dictates a “Banker_Role" to handle “Manager” and “Supervisor” tasks, and a "Customer_Role" to handle “Client” and “User” tasks.

So, the developers provide the module security role references and the assembler defines the application security roles of “Banker_Role” and “Customer_Role”. The assembler then completes the process by linking the security role references to security roles.

7.3.2 WebSphere security components

As shown in Figure 7-1 on page 110, security for WebSphere Application Server is managed as a collaborative effort by a number of components:

- Web and Java clients
- Security server
- Security collaborators
- The user registry
- Security policies
- The Secure Association Service (SAS)
- Secure Sockets Layer (SSL)

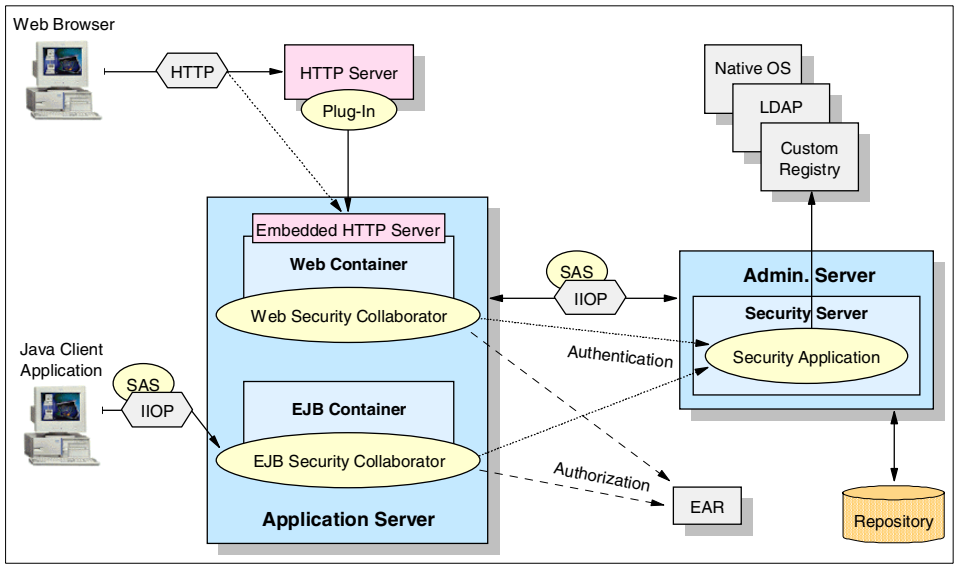


Figure 7-1 WebSphere Advanced Edition security architecture

Web and Java clients

As can be seen in Figure 7-1, WebSphere supports two types of clients, Web (browser) clients and the stand-alone Java application or applet clients. J2EE application clients and stand-alone Java clients can go directly to an EJB under the control of an application server using IIOP. Access by both Web clients and Java clients can be controlled by WebSphere security.

Security server

As can be seen in Figure 7-1, the security server resides within the WebSphere administrative server. It supports both the administrative aspects of WebSphere security, discussed in 7.3.3, “WebSphere security administration model” on page 114, and the runtime aspects.

At runtime, the security server is part of the security application and provides the authentication service. The security server will consult with the user registry (an LDAP server, for example) to authenticate a user and to obtain a credential that can be used in the security context to represent the user identity.

In this respect, the security server is a trusted third party for security policy and control. The WebSphere security collaborators executing in each application server call on the security server to provide authentication services (including Token services when the LTPA authentication mechanism is used).

The security runtime components acquire global security configuration, such as the authentication mechanism and user registry, from the security application. Because the security application is coupled with the WebSphere administrative server, the security configuration information resides in the WebSphere administrative repository (for example, a DB2 database).

The security collaborators

The security collaborators reside in the application server process and are the key runtime components for enforcing the security constraints and attributes specified in the deployment descriptors. As shown in Figure 7-1 on page 110, there is a collaborator for Web resources in the Web container and another collaborator in the EJB container.

The Web collaborator performs authentication and authorization. The EJB collaborator performs authorization, but not authentication, and sets the run-as identity for delegated request. The EJB collaborator relies on the Secure Association Service (SAS) to authenticate Java client requests to enterprise beans. Both collaborators perform an authorization check and log security information when a client request is made for a Web or EJB resource.

If the client is not already authenticated, the Web collaborator can challenge the user to provide a user ID and password before performing the authorization check. The challenge mechanism is specified as the login-configuration element in the Web archive's web.xml deployment descriptor.

The EJB collaborator performs an additional operation after performing the authorization check. It sets the run-as identity, based on the delegation policy. The delegation policy determines the identity to use if the enterprise bean invokes methods on any other enterprise beans. The delegation policy or run-as mode is specified in the ejb-jar.xml deployment descriptor.

User registry

One of the first steps toward enforcing security restrictions is to require users to authenticate, or prove their identities, in order to access applications. A user may submit information such as a password or a certificate to prove their identity. The security system then checks the information against a database of known users. If the submitted information matches the information in the database, the user has successfully authenticated.

The database of known users is a registry. As shown in Figure 7-2, WebSphere Application Server supports the following types of registries:

- **Local registries**, which are limited to environments with a single application server and single node or Windows NT domain controller.

- **Centralized registries**, which use the Lightweight Third Party Authentication (LTPA) protocol to access:
  - **Lightweight Directory Access Protocol (LDAP) registry** using one of the supported LDAP servers.
  - **Custom user registry** using the WebSphere custom registry interface.

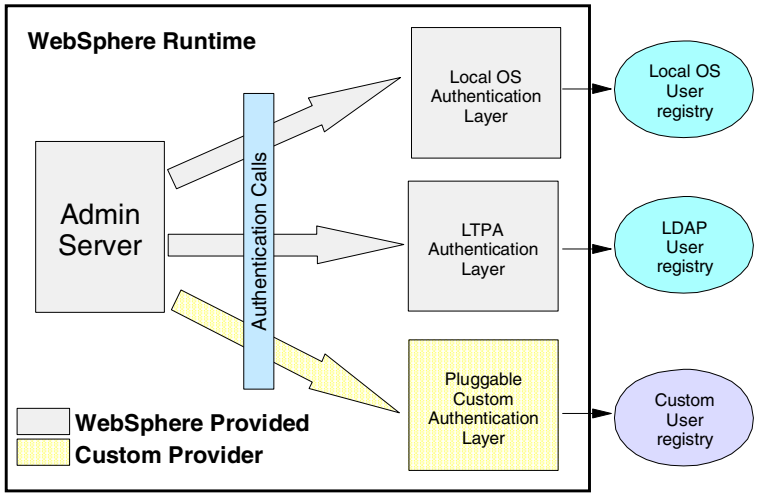


Figure 7-2 WebSphere user registry options

The new pluggable custom user registry option allows WebSphere security to verify users stored in a custom user registry, such as a legacy user database. To allow WebSphere security to interact with any given registry:

1. The application developer implements the methods in the CustomRegistry interface, as defined in the com.ibm.websphere.security package. This layer of code will interact with the custom user registry.
2. The WebSphere administrator selects the Custom User Registry authentication option and specifies the class implementing the CustomRegistry methods.
3. The WebSphere security server calls the CustomRegistry methods to perform authentication for applications.

Security policies

Security attributes for enterprise applications are specified in XML deployment descriptors in the enterprise application archive (EAR), typically using a tool like the Application Assembly Tool (AAT).

Note: Editing the EAR file directly can have some serious side effects, especially in respect to the security policy. In a server group configuration the security information in the deployment descriptors in the EAR file may be on more than one machine. You need to be very careful to keep all of the EAR files in sync.

The security attributes include roles, method permissions, the run-as mode or delegation policy, login-configuration or challenge type, and data-protection (confidentiality and integrity) settings.

When an application is deployed, security roles are mapped to users or groups. This completes the mapping of users and groups to the protected enterprise bean methods and Web methods, forming the authorization table. There is an authorization table for each enterprise application, which is used by the security collaborators during the authorization check.

For more information on security-related attributes for deployment, see:

- Chapter 21, “Configuring security” on page 739.
- InfoCenter article, “Using the Application Assembly Tool interface”.
- The Servlet 2.2 specification, for Web resources.
- The Enterprise JavaBeans 1.1 specification, for EJB resources.

The Secure Association Service (SAS)

All requests from Java clients to EJBs are sent as RMI/IIOP messages to the application server that hosts the EJBs. The Secure Association Service (SAS) is invoked on both the client and the server sides.

On the client side, SAS intercepts requests before they are sent, obtains the client's security credentials, attaches the credentials to the request as part of the security context, and sends the request.

On the server side, SAS intercepts the incoming request, extracts the security context from the message, authenticates the client's credentials, and passes the request to the EJB container, where the request is authorized.

SAS is also used to provide message protection of communication between application server(s) and administrative server(s).

SSL

Secure Sockets Layer (SSL) is a public-key network security protocol that can perform message encryption, client authentication, and server authentication.

When global security is enabled, SSL is used to encrypt all IIOP communications with application servers and administrative servers, by default. SSL can be used to encrypt HTTP communications between Web browser and Web server, and Web server and application server. WebSphere also support LDAPS (LDAP over SSL) between the security server and the LDAP server.

For further details on how to configure SSL with WebSphere Application Server, see Chapter 21, “Configuring security” on page 739.

Security scenario

Now let’s examine a security flow for a request coming from a browser user (Web client). In our scenario, the request is for a JSP. The flow is as follows:

1. The Web client makes an HTTP request to the Web server for a protected Web resource such as a JSP file.
2. The Web server determines that it does not control the JSP, so passes the request on to the application server.
3. The application server determines that it controls the JSP and dispatches the request to the Web security collaborator for the security check.
4. The security collaborator determines that the client must be authenticated and challenges the Web client to provide a user ID and password.
5. The Web client provides a valid user ID and password back through the Web server to the application server.
6. The Web collaborator authenticates the user ID and password supplied by the client against a user registry, for example, an LDAP registry.
7. If the client is successfully authenticated, the security collaborator then consults the permissions in the deployment descriptor in the EAR to determine whether the user is in one of the roles protecting the resource and, if so, permits access.
8. The application server invokes the JSP for the user.

7.3.3 WebSphere security administration model

The different components of WebSphere Application Server use a unified model for security, so after you learn how to set up security for one type of resource, you can apply that knowledge to other resources. You use similar approaches to protect enterprise beans, servlets, JavaServer Pages, and Web pages in a secure enterprise application.

As shown in Table 7-1 on page 115, global security is administered using the WebSphere Administrative Console. Security information that is classified as global applies to all applications running in the environment. Global security information is stored in the administrative repository, with a few items kept in properties files.

Table 7-1 WebSphere V4.0 AE security information administration and storage

Security information	Administration tool	Storage location
Global security, Authentication, SSL settings	Administrative console	Administrative repository
Application security roles	AAT	EAR DD application.xml
Security role binding to users, groups, special subjects	AAT, Administrative Console	EAR DD ibm-application-bnd.xmi, Administrative repository
EJB ► Security role ► Role reference ► Method permission	AAT	EJB DD ejb-jar.xml
EJB ► Security identity - RunAs	AAT	EJB DD ibm-ejb-jar-ext.xmi
EJB ► RunAs mapping	Administrative Console	EAR DD ibm-application-bnd.xmi
Web resources ► Security role ► Role reference ► Security constraints	AAT	WAR DD web.xml

Application security is administered during the assembly phase using the Application Assembly Tool (AAT). It is administered during the deployment phase using the Administrative Console, the WebSphere Control Program tool, or the XMLConfig tool. Application-specific security information is tailored to individual applications.

Application security information is stored in the application deployment descriptors (DDs) in the expanded enterprise application archive (EAR) in the <WAS_HOME>/installedApps directory. Security mappings from roles to users or groups are stored in the administrative repository.

Mappings can be stored by AAT in the ibm-application-bnd.xmi file as preferences. If there is mapping information in the repository, it takes precedence over the information in the .xmi file.

As shown in Table 7-2, when global security is enabled EJB resources are now unprotected by default, as with Web resources. When at least one EJB method has security defined, access to all other methods of any instance of that bean in the domain will be denied. If you only define security on one bean, other beans will remain unprotected.

Table 7-2 WebSphere V4.0 default security settings

Setting	WebSphere V3.5.x	WebSphere V4.0
Security *not* defined for any method in EJB, servlet	EJB: Denied Web resource: Grant	EJB: Grant Web resource: Grant
Default setting for methods with *no* security definition in resources, where at least one method has security defined	EJB: Denied Web resource: Grant	EJB: Denied Web resource: Grant

7.3.4 WebSphere authentication model

Authentication is the process of determining if a user is who the user claims to be. WebSphere security authenticates a user in two steps. First, authentication data in the form of a user ID and password or a certificate are obtained for the user. Second, the authentication data is validated against information contained in a user registry.

Authentication methods

The servlet specification identifies a number of authentication methods for Web resources. WebSphere supports the following four methods:

- **None**. The user is not challenged. If the resource being requested is secure, then it is not served for the user.
- **Basic**. The user is challenged to enter a user ID and password using the standard unencrypted Basic HTTP Authentication.
- **Form**. The user is challenged to enter a user ID and password using a custom HTML form instead of the normal basic challenge.
- **Client certificate** (X.509 certificate). The user is challenged to supply a digital certificate.

The servlet specification also identifies the digest authentication method that transmits the password in encrypted form. However, digest authentication is not supported by WebSphere V4.0.

The authentication method is defined individually for Web applications. Hence each Web application in an enterprise application can use a different authentication method.

See 21.3.2, “Login configuration” on page 763 for details on how to configure the Web application authentication method.

J2EE does not specify authentication methods for enterprise-bean resources. However, WebSphere uses the Secure Association Service (SAS) to authenticate Java clients to enterprise beans.

Authentication mechanisms

WebSphere Application Server authenticates users by using one of several authentication mechanisms. An authentication mechanism validates the authentication information against a user registry. WebSphere supports two mechanisms for authentication:

- **Local operating system**. The underlying operating system is used to authenticate the user. The local operating system supports the basic and form authentication methods.
- **Lightweight Third Party Authentication (LTPA)**. LTPA can use a trusted third-party Lightweight Directory Access Protocol (LDAP) server, or a custom user registry to authenticate the user. LTPA supports the basic, form, and client certificate authentication methods.

Note: If WebSphere security is to be enabled when running the administrative server as a non-root user, then the local operating system cannot be used as the authentication mechanism. You have to use LTPA in connection with LDAP.

See 21.1.2, “Selecting the authentication mechanism” on page 741 for details on how to configure the WebSphere authentication mechanism.

7.3.5 WebSphere authorization model

Authorization information is used to determine if a caller has the necessary privilege to access a resource. WebSphere Application Server uses the Java 2 Enterprise Edition (J2EE) authorization model. In this model, authorization information is defined as follows:

1. Create roles and permissions during assembly of the application, so permission to execute methods is granted to one or more security roles.

2. Assign subjects (such as users or groups) to roles during deployment of the application, so real users are assigned to the security roles.

Figure 7-3 illustrates this mapping between users, groups, roles, and methods.

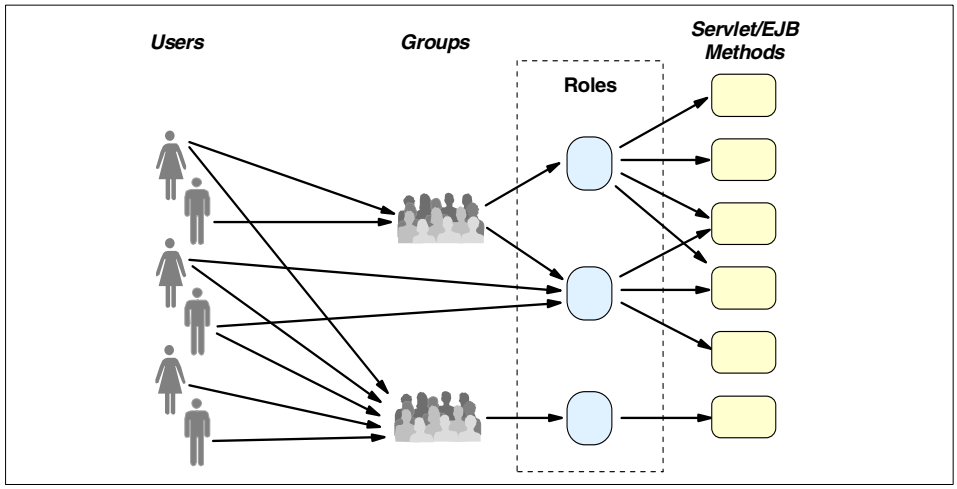


Figure 7-3 User-role-method mappings

At execution time, WebSphere Application Server authorizes incoming requests based on the user's identification information and the mapping of the user or group to roles. If the user belongs to any role that has permission to execute a method, the request is authorized. If the user does not belong to any role that has permission, the request is denied.

Creating roles and permissions

During the assembly of an application, permission to execute methods is granted to one or more security roles. As an example, the roles in a banking application, listed in Table 7-3, include Teller, WebTeller, and Supervisor. The Teller role is granted permissions to run methods related to managing the money in an account, such as withdraw and deposit methods. The Teller role is not granted permission to close accounts, since that permission is given to the Supervisor role. The application assembler defines a list of method permissions for each role, which is stored in the deployment descriptor for the application.

Table 7-3 Role to method permission mapping

Role	AccountBean method			AccountServlet method	
	withdraw	deposit	close-Account	HTTP GET	HTTP POST
Teller	yes	yes	no	no	no

Role	AccountBean method			AccountServlet method	
	withdraw	deposit	close-Account	HTTP GET	HTTP POST
WebTeller	yes	yes	no	yes	yes
Supervisor	no	no	yes	no	no

WebSphere also allows roles to be assigned to special subjects that are not defined by J2EE. A special subject is WebSphere defined entity that is independent of the user registry. It is used to generically represent a class of users or groups in the registry.

- **AllAuthenticatedUsers** is a special subject that permits all authenticated users to access protected methods. As long as the user can authenticate successfully, the user is permitted access to the protected resource.
- **Everyone** is a special subject that permits unrestricted access to a protected resource. Users do not have to authenticate to get access, as this special subject allows access to protected methods as if the resources are unprotected.
- **DenyAllRole** is a special role that is assigned by default to a partially protected resource. For instance, if an enterprise bean has four methods and only three are explicitly protected, the fourth method is associated with the DenyAllRole. This role denies everyone access to the methods it is associated with. The DenyAllRole is never mapped to any users or groups; it is always empty.

Assigning subjects to roles

During the deployment of an application, users or groups of users, called subjects, are assigned to the roles previously defined by the application assembler. The application deployer does not need to understand the individual methods, as the application assembler grants method permissions to the roles.

When a user is assigned to a role, the user gets all the method permissions that are granted to that role. If a user is assigned to more than one role, the permissions granted are the union of the permissions granted to each role. Additionally, if the authentication mechanism supports the grouping of users, these groups can be assigned to roles. Assigning a group to a role has the same effect as assigning each individual user to the role.

As an example, the roles in a banking application are assigned as listed in Table 7-4. Users belonging to the TellerGroup are assigned method permissions of the Teller and WebTeller roles, but not the Supervisor role.

Table 7-4 Subject to role mapping

Subject	Role		
	Teller	**WebTeller**	**Supervisor**
TellerGroup	yes	yes	no
Bob	yes	yes	yes
Supervisor	no	no	yes

A "best practice" during deployment is to assign groups, rather than individual users, to roles for the following reasons:

- It improves performance during the authorization check. There are typically far fewer groups than users.
- Using group membership to control resource access provides better flexibility.
- Users can be added to and deleted from groups in the user repository. This is preferred to adding and removing users for WebSphere roles, since the enterprise application must be stopped and restarted for such changes to take effect, which can be very disruptive in a production environment.

Programmatic authorization

J2EE uses a declarative approach to authorization, but it also recognizes that not all situations can be dealt with this way. Consider a business rule dictating that the Teller role should be authorized to perform transactions under $5000 only. In this case the user role alone is not sufficient to determine if the transaction should be authorized. For these situations, methods are provided for determining user and role information programmatically.

For servlets, the following methods are supported by WebSphere Application Server:

- getUserPrincipal or getRemoteUser: retrieves the user's identification information.
- isUserInRole: checks the user's identification information against a specific role.

Similarly for EJBs, the following methods are supported by WebSphere Application Server:

- getCallerPrincipal: retrieves the user's identification information.

- isCallerInRole: checks the user's identification information against a specific role.

7.3.6 WebSphere delegation model

Delegation allows an intermediary to perform a task initiated by a client under an identity determined by the associated policy. Therefore, enforcement of delegation policies affects the identity under which the intermediary performs downstream invocations. By downstream invocations, we mean invocations made by the intermediary on other objects, in order to complete the current request.

If no delegation policy is set, the intermediary will use the identity of the requesting client while making the downstream calls. Alternatively, the intermediary can perform the downstream invocations under its own identity or under an identity specified by configuration.

When the intermediary operates under an identity other than its own, downstream resources do not know the identity of the intermediary. Therefore, they make their access decisions based on the privileges associated with the identity being used.

With WebSphere Application Server, the application assembler can use AAT to choose among the delegation policy options listed in Table 7-5.

Table 7-5 WebSphere delegation policy options

AAT Run-As mode setting	**Deployment descriptor SecurityIdentity value**
Use identity of caller	UseCallerIdentity (cannot be used for message-driven beans)
Use identity of EJB server	UseSystemIdentity
Use identity assigned to specified role	RunAsSpecifiedIdentity

Use of UseCallerIdentity means that the intermediary will use its client's credentials for downstream invocations. Use of UseSystemIdentity means that the intermediary will use its own credentials for downstream invocations. Use of RunAsSpecifiedIdentity means that credentials determined elsewhere will be used.

The application assembler does not typically know the makeup of the runtime environment, including the specific user identities that are available. Therefore, the run-as identity is designated as a logical role name, which corresponds to one of the security roles defined in the deployment descriptor. At deployment time, a particular user is assigned to that role and becomes the run-as identity by indirection. This allows you to use the specified-identity delegation policy to run beans under the identity of a user who has been associated with the role.

As an example of delegation policy, suppose that a client invokes a session bean that invokes an entity bean. If the delegation policy states that methods are invoked under the client's identity, the session bean makes its invocations under the client's identity. Therefore, it is the client, rather than the session bean, that must have permission to invoke the entity-bean methods. If the delegation policy requires the system identity, the session bean makes its invocation under the identity of the server in which the session bean resides, so it is this server that must have permission on the entity-bean methods. Finally, if the delegation policy requires a specified identity, the session bean invokes the methods under this identity, so the specified identity must have permission on the entity-bean methods.

The WebSphere delegation model is an extension the Enterprise JavaBeans 1.1 specification. Delegation is fully addressed in Enterprise JavaBeans 2.0 specification. Enterprise beans can have delegation policies, but Web resources cannot.

7.3.7 WebSphere security and the operating environment

In this section we discuss how WebSphere security relates to the security provided by your operating system and by Java.

WebSphere Application Server security sits on top of your operating system security and the security features provided by other components, including the Java language. As shown in Figure 7-4 on page 123, the types of security involved include:

- **Operating system security** is used to secure sensitive files in the WebSphere product installation, to authenticate users using the operating system user registry.
- **Java language security** provided through the Java Virtual Machine (JVM) used by WebSphere and the Java security classes.
- **CORBA security** for inter-application calls between secure ORBs invoked over the Secure Association Service (SAS) layer.
- **J2EE security** using the security collaborator to enforce J2EE-based security policies and support J2EE security APIs.

- **WebSphere security** relies on and enhances all of the above. It enforces security policies and services in a unified manner for Web and EJB resources.

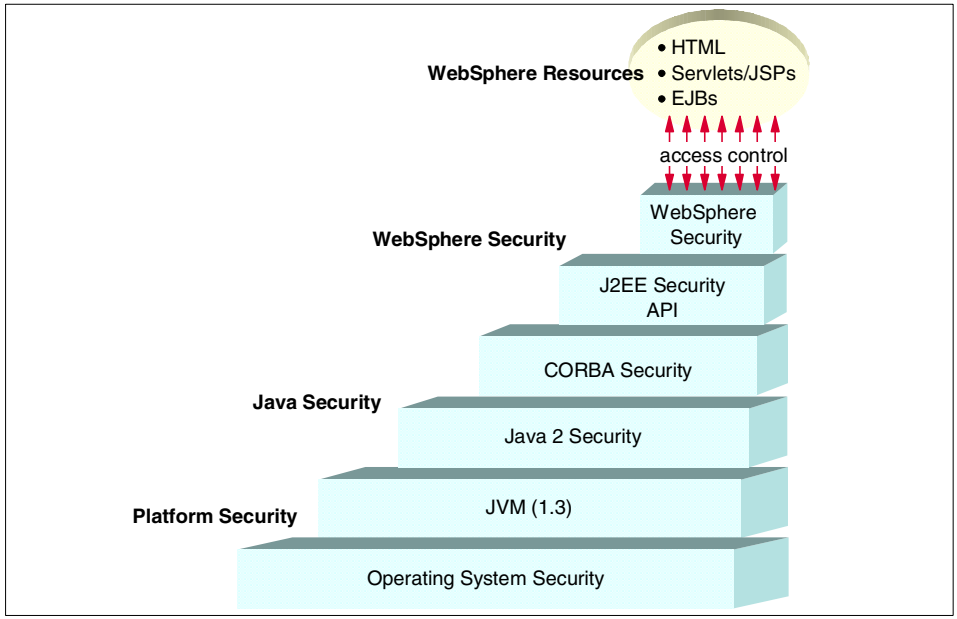


Figure 7-4 WebSphere security layers

7.4 Performance considerations

There is a performance cost associated with enabling WebSphere security. This performance hit can be eliminated by disabling security, of course. However, it may be more appropriate to consider other alternatives, such as:

- Disabling security on selected application servers, as described in 21.1.6, “Securing only the administrative server” on page 750.
- Using an HTTPS transport between the Web server and application server without enabling application server security. See 9.9, “Configure WebSphere HTTP transport for SSL” on page 245 for details.

Refer to the InfoCenter “Tuning Guide” article for further discussion on how various settings related to security affect performance.

7.5 More information

These documents and Web sites are also relevant as further information sources:

- Redbook *IBM WebSphere Advanced Edition: Security*, SG24-6520

  http://www.redbooks.ibm.com/abstracts/sg246520.html

- IBM WebSphere InfoCenter

  http://www.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter

- J2EE specifications

  http://java.sun.com/j2ee/docs.html

  See the “Specifications” section.

Part 3

Installing WebSphere

In this part we explain how to plan your WebSphere V4.0 installation. We step you through the installation process on Windows, AIX, Solaris, and Linux platforms for commonly used topologies.

8

Installation approach

This chapter provides an explanation of the procedures for installing, configuring, and verifying IBM WebSphere Application Server V4.0, Advanced Edition. Since WebSphere Application Server can be installed in a number of different configurations, this chapter provides a summary of the decisions that must be made when planning an installation to match a particular topology or architecture.

This chapter is organized into the following sections:

- Planning your installation
- Perform pre-installation tasks
- Perform WebSphere installation
- Perform post-installation tasks
- Uninstalling WebSphere
- Example scenarios

We provide specific installation examples for Windows 2000/IBM HTTP Server/DB2, AIX/IBM HTTP Server/DB2, Linux/IBM HTTP Server/DB2 and Solaris/iPlanet/Oracle in the chapters that follow.

8.1 Planning your installation

Since WebSphere Application Server V4.0 can be installed in a number of different configurations, this section provides a step-by-step summary of the decisions that need to be made as you plan your installation.

Prior to performing an installation, you should consider each of the following options, as each affects the method and tasks used during the installation:

- Are you migrating an existing installation?
- Should the GUI or a silent installation be used?
- Should a typical or a custom installation be used?
- Will you be installing into an existing administrative domain or creating a new domain?
- Should the embedded HTTP transport be used in place of a stand-alone Web server?
- Will the Web server be located on a separate (remote) machine from the WebSphere Application Server?
- Will the database server be located on a separate (remote) machine from the WebSphere Application Server?
- Is customization of prerequisite checking required?

Directly after the installation of the WebSphere components, the following configuration options should be considered:

- Run on non-standard ports?
- Run with multiple NICs?
- Run as non-root user (UNIX only)?
- Run administrative server in the background (UNIX only)?
- Encrypt communication between plug-in and WebSphere?

A detailed description of each of these options, along with their advantages and disadvantages, is provided in the following sections.

8.1.1 Migration of existing installation?

A description of the decisions and tasks involved in migrating an existing WebSphere Application Server 3.0.2.x or 3.5.x installation to 4.0 is provided in Chapter 25, “Migration” on page 1031.

This chapter concentrates on the decisions and tasks associated with a “clean” installation of WebSphere Application Server.

8.1.2 Use GUI or silent installation?

The WebSphere Application Server installation program provides two modes of operation:

- GUI installation

  Interactive installation that displays a graphical user interface within which the choices are displayed a screen at a time, allowing easy setup and execution of the installation.

- Silent installation

  Non-interactive installation that reads all choices from a script file that must be pre-prepared by the user.

Note: The native installation method provided in previous versions of WebSphere Application Server is no longer available.

The GUI installation option should be used for:

- First time or inexperienced users.
- One-time installs.
- Determination of the settings required to create a script file used for later silent installations.
- Testing of installations involving different combinations of software components.

The silent installation option should be used for:

- Installation of identical configurations of WebSphere Application Server on multiple machines.

  If you are installing WebSphere Application Server on multiple machines, and want them to have identical configurations, you can use the XML export/import utility and the silent installation functionality to do so. Installing WebSphere Application Server in this manner will result in identical but separate installations of WebSphere. In other words, the application servers will not recognize each other.

  However, such an installation does not prepare your system for purposes such as configuring a server group, which requires that all machines in the group be part of the same administrative domain.

- Backup of the installation settings for later reuse.

- Scripted or unattended installations.
- Duplication of development or testing installation in production.

8.1.3 Use typical or custom installation?

The WebSphere Application Server installation program provides two installation options:

- Typical installation

  Automatically installs a WebSphere Application Server configuration consisting of the following components on a single server:

  - WebSphere Application Server V4.0
  - IBM HTTP Server 1.3.19
  - WebSphere HTTP plug-in for IBM HTTP Server
  - IBM JDK 1.3.0
  - Sample applications

  Important: The typical installation option assumes that a DB2 database has been separately installed prior to running the WebSphere Application Server installation.

- Custom installation

  Allows the user to choose among various supported Web servers and database servers and to specifically configure WebSphere Application Server based on those choices.

  Using this option you can:

  - Choose from among a number of supported Web servers already installed on the server, or to install the IBM HTTP Server during the installation.
  - Choose a non-standard JDK already installed on the server, or to install the standard IBM JDK during the installation.
  - Choose from a number of different supported database servers. Any database chosen must be installed and configured on the server prior to running the WebSphere Application Server installation.
  - Choose to install only the HTTP plug-in required by a remote Web server in order to access an existing WebSphere Application Server.
  - Choose to install an instance of WebSphere Application Server that shares a repository database with an existing installation, resulting in a multi-node administrative domain configuration.

The typical installation option should only be used in the following situations:

- First-time installations.
- Where a single server installation composed of WebSphere Application Server, IBM HTTP Server, and DB2 server is required.

We recommend that the custom installation option be used, since it provides the choices and flexibility needs for all but the simplest of WebSphere Application Server configurations.

8.1.4 Use existing or create new administrative domain?

WebSphere Application Server provides the functionality to share a single administrative repository database across multiple WebSphere installations, resulting in a common administrative domain that is accessed through the WebSphere administrative server process running on each administrative node.

Reasons to consider using a common administrative domain include:

- Establishing centralized administration

  Multiple administrative servers can be configured to use the same administrative database, allowing configuration data to be kept in a centralized place for administration of WebSphere Application Server across multiple machines (known as administrative nodes). In a successful multiple node configuration, the administrative console tree view will show each of the administrative nodes and its contents.

- Establishing server groups

  A single enterprise application can be configured to run clones of application servers and/or EJB containers across multiple nodes, providing both horizontal and vertical scalability and application failover potential.

  Additionally, a cloned configuration can support workload management without requiring the use of a “sprayer”, such as Network Dispatcher, in front of the Web server.

- Establishing administrative database high availability

  The single administrative database can be placed on a dedicated high-availability database server that can provide services such as:

  – Clustering
  – Failover
  – Backup and recovery
  – Optimized database performance

8.1.5 Use embedded HTTP transport or stand-alone Web server?

A new feature provided by WebSphere Application Server V4.0 is the embedded HTTP transport provided by each Web container. It is now possible to direct requests for servlets, JSPs or static resources to a HTTP transport configured in the Web container of each application server.

Although provided to service requests from the new WebSphere HTTP Web server plug-in, the transport can also be accessed directly by any HTTP client. It is no longer mandatory to install and configure a separate Web server in order to access resources hosted by a WebSphere Application Server.

Reasons for using the embedded HTTP transport:

- Easy to configure

  All application changes are immediately available to the transport as a result of the direct access to the administrative database. There is no Web server to configure or plug-in configuration file to regenerate whenever a servlet URL is added or changed.

- Stand-alone verification of WebSphere operation

  The WebSphere Application Server can be verified by directly accessing test or application URLs without having to make the request via an intervening Web server. Any problems can be immediately identified as resulting from the WebSphere configuration, not Web server or plug-in problems.

- Stand-alone verification of an application

  By requesting an application's URLs directly through the embedded HTTP transport, an application can be verified prior to configuring a Web server plug-in to support the application.

Reasons to not use the embedded HTTP transport:

- Does not support workload management (server group and clone) functionality.

- Lacks the performance for production applications that is available when using the HTTP plug-in.

  All static content, as well as dynamic, must be served by the WebSphere Application Server. Web servers perform much better than application servers at serving static content.

- Unable to separate Web server and WebSphere interfaces.

  In a firewall environment, the WebSphere Web server would have to be located either outside the firewall (open to attack) or inside the firewall, requiring the firewall to be configured to pass HTTP requests (opening a potential security hole in the firewall).

We recommend that a separate Web server always be installed and configured to provide access to the resources hosted in WebSphere Application Server. However, the embedded HTTP transport should be used to provide a verification functionality separate from the Web server.

8.1.6 Use remote Web server?

The WebSphere HTTP plug-in can be installed into a Web server located on a separate (remote) server from the WebSphere Application Server. The Web server plug-in uses an XML configuration file (plugin-cfg.xml) containing settings that describe how to handle and pass on requests to the WebSphere Application Server(s) made accessible through the plug-in.

Note: The OSE plug-in provided in previous releases of WebSphere Application Server is not supported in WebSphere V4.0. All installations should now use the new HTTP plug-in.

Reasons to use a remote Web server:

- Separate Web server and WebSphere interfaces

  In a firewall environment, the Web server can be located outside the firewall, while the WebSphere Web server is located inside the firewall. The only requirements is that the firewall be configured to allow passage of HTTP requests.

- You can size and configure servers appropriate to each task

  By installing components (Web server and application server) on separate machines, each machine can be sized and configured to optimize the performance of each component. In addition, a component's server can be reconfigured, or even replaced, without affecting the installation of the other component.

- Remove resource contention

  By installing the Web server on a separate machine from the WebSphere Application Server, a high load of static requests will not affect the resources (CPU, memory, disk) available to WebSphere, and therefore will not affect its ability to service dynamic requests.

Reasons to not use a remote Web server:

- Configuration complexity

  The configuration file is generated on the WebSphere Application Server machine and must then be manually copied to the Web server machine.

- Network access may limit performance

  Depending upon the network hardware and remoteness of the Web server, the network response time for communication between WebSphere Application Server and the Web server may limit the performance of the Web server. When co-located on the same server, network response is not an issue.

8.1.7 Use a remote database server?

WebSphere is usually configured to access the database server, for example, DB2 Server Enterprise Edition or Oracle 8i, via TCP/IP. As long as the host name, port number and database ID settings are correct, WebSphere will be able to communicate with a database server located anywhere accessible through TCP/IP.

Reasons to use a remote database server:

- You can size and configure servers appropriate to each task

  By installing components (application server and database server) on separate machines, each machine can be sized and configured to optimize the performance of each component. In addition, a component's server can be reconfigured, or even replaced, without affecting the installation of the other component.

  For example, the installation and execution of Oracle 8i on Solaris requires tuning of the kernel settings to meet Oracle requirements. This database-specific tuning is often detrimental to the performance of application servers located on the same machine.

- Remove resource contention

  By installing the database server on a separate machine from the WebSphere server, they will not have to compete for system resources (CPU, memory, disk).

Reasons to not use a remote database server:

- Network access may limit performance

  Depending upon the network hardware and remoteness of the database server, the network response time for communication between WebSphere Application Server and the database server may limit the performance of WebSphere. When co-located on the same server, network response is not an issue.

- Architectural complexity

  Hosting the database server on a separate machine introduces yet another box that must be administered, maintained and backed up.

- Cost

  The cost of a separate database server may not be justified by the environment in which the WebSphere Application Server will be installed.

8.1.8 Customize prerequisite checking?

To determine your hardware and software requirements, visit the IBM WebSphere Application Server prerequisites site:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

The site contains entry points to prerequisite information for multiple versions of WebSphere Application Server. Use the information to help determine the specific hardware and software required for your WebSphere Application Server environment.

The software components and associated versions supported by a WebSphere Application Server installation package are listed in a separate file (prereq.properties) located at the root level of the installation package. The file contains settings used by the installation program to determine which WebSphere Application Server prerequisites should be checked during the installation.

Customization of prerequisite checking should only be considered in the following situations:

- You plan to use a Web server or database server at a level that exceeds the current version required by the WebSphere Application Server.
- The operating system has been patched with versions of patches that exceed those required by the WebSphere Application Server.

  On some operating systems (for example, Solaris), patches can be superseded by newer versions that may have a different ID or name. The WebSphere Application Server installation program will identify this situation by displaying an error alert indicating “missing patches of the required version”. In this situation, the user will have to:

  – Manually verify the patch levels.
  – If correct, disable prerequisite checking.
  – Restart the installation.

In order to ensure a correctly working installation, we recommend that installations be performed with prerequisite checking enabled and uncustomized. Only in the event of a prerequisite error being reported should customization be performed and the installation repeated.

8.1.9 Run on non-standard ports?

WebSphere Application Server uses a number of IP ports to listen for requests for its various services. Although some of these port numbers are dynamically allocated from those free on the system, other port numbers are specified in WebSphere settings. However, it is possible to reconfigure the WebSphere settings to use other “non-standard” ports for these cases.

Reasons for using non-standard port numbers include:

- The standard ports are in use by other software already installed on the server.
- Limiting administration accessibility

  The WebSphere bootstrap service default port (900) is well known, providing easy access via administration tools (administrative console, XMLConfig and WSCP) to any WebSphere administrative domain with security disabled. Although running on different ports does not secure the WebSphere administrative domain, it may make it harder to access without effort.
- WebSphere is run under a non-root account (UNIX only).

  Any WebSphere process run as non-root and using a port number lower than 1024 must be reconfigured to use a port number greater than 1024.

  Tip: On UNIX systems, only the root account has access to IP ports lower than 1024.

- If two instances of WebSphere are installed on the one server.

  If two instances (for example, different versions) of WebSphere Application Server are installed on the same server, only one instance can use the standard ports. The other instance must be reconfigured to use different port numbers that do not conflict.

We recommend that unless restrictions of the environment do not allow it, WebSphere Application Server processes should be run on the standard ports.

8.1.10 Run with multiple NICs?

WebSphere tolerates the presence of more than one network interface card (NIC) on a single platform as long as certain conditions are met. If your environment requires multiple NICs on a machine hosting WebSphere Application Server then you need to consider this installation option.

8.1.11 Run as non-root user (UNIX only)?

Any application server or administrative server can be run using a non-root ID. However, by default WebSphere processes use a root ID.

Reason to use non-root ID:

- Limit access to root account.

  It is normal administration practice on UNIX systems to limit those processes that run as root to essential system processes only. This measure limits the extent to which a malicious program or user can gain complete access to the system.

  Each application server, database server or Web server should, if possible, be run under a non-root account created specifically for its use.

Reasons to not use non-root ID:

- Administration complexity

  If run as non-root, the permissions of WebSphere directories and files must be adjusted to provide the necessary access rights for the non-root account. In addition, the user and group under which each WebSphere administrative server is run must be configured in the WebSphere administrative domain settings.

- If run as non-root you must use an LDAP directory as the authentication mechanism for WebSphere security. You can no longer use the local operating system.

Note: The Java administrative console can be accessed from a non-root ID, provided security permissions are configured appropriately.

We recommend that for simplicity reasons, first-time, development, and testing, WebSphere Application Server installations should be run as root. In a production environment, the reasons given along with the restrictions of the environment should be used as input to the decision making process.

8.1.12 Run administrative server in the background (UNIX only)?

Processes in the UNIX preemptive multitasking environment can be set to run in the background, that is, run at a low priority so other processes (with higher priority) will get more processing cycles.

We recommend that this configuration not be used in production installations of WebSphere Application Server.

8.1.13 Encrypt communication between plug-in and WebSphere?

The new Web container HTTP transport provided in WebSphere Application Server V4.0 provides functionality to enable SSL encryption of communication between the transport and the Web server HTTP plug-in (either on the same or a remote server). The handshaking that occurs at the start of each SSL request involves an exchange and trust determination of digital certificates between the client (Web server HTTP plug-in) and server (HTTP transport).

Reasons to encrypt the communication between the Web server plug-in and HTTP transport:

- Secure communication with remote Web server.

  SSL encryption of the communication makes sense if the Web server is located remotely, for example, on the Internet side of a firewall, but would make little sense if on the same machine.

- Enforce authentication between the Web server and HTTP transport.

  By choosing appropriate Certificate Authorities (CA) and certificates, the SSL mechanism can be configured so that the plug-in and HTTP transport must mutually authenticate each other. The result is that only known/allowed Web servers can successfully access a particular WebSphere Application Server, and vice versa.

Note: In WebSphere Application Server 3.5.x, to control which Web servers could successfully pass requests through to a given WebSphere Application Server, editing of the virtual host is required to contain the following Host Aliases entries for known/allowed Web servers only:

```
<hostname:<port#>
<hostname>.<domain.com>:<port#>
<IP address>:<port#>
```

This method is still available in WebSphere Application Server V4.0, but DNS spoofing makes it much less secure than the digital certificate mechanism provided by the SSL functionality of the HTTP transport.

8.2 Perform pre-installation tasks

Before WebSphere Application Server can be installed, the following high-level installation and configuration tasks need to be performed as required on each server comprising your solution. Table 8-1 shows a breakdown of the tasks for each of the component combinations that may result from your installation planning stage.

Table 8-1 Pre-installation tasks by component architecture

Software Component	Server 1	Server 2	Server 3
Web server and database server located on the same server as WebSphere	Application, database and Web server 1. Install the operating system 2. Install and configure Web server 3. Install and configure database server 4. Create administrative database 5. Customize prerequisite checking (optional)	-	-
Remote Web server. Database server located on the same server as WebSphere.	Application and database server 1. Install the operating system 2. Install and configure database server 3. Create administrative database 4. Customize prerequisite checking (optional)	Web server 1. Install the operating system 2. Install and configure Web server	-

Software Component	Server 1	Server 2	Server 3
Remote Web server. Remote database server.	Database server 1. Install the operating system 2. Install and configure database server or client 3. Create administrative database	Web server 1. Install the operating system 2. Install and configure Web server	Application server 1. Install the operating system 2. Install database client 3. Customize prerequisite checking (optional)

For examples of the installation of each of these components on various platforms, and their configuration to meet WebSphere Application Server V4.0 requirements, refer to the chapters listed in Table 8-2.

Table 8-2 Component installation examples

Platform	Examples provided in ...
Windows 2000	Chapter 9, “Windows 2000 installation steps” on page 181
AIX 4.3.3	Chapter 10, “AIX installation steps” on page 263
Solaris 2.7	Chapter 11, “Solaris installation steps” on page 329
Red Hat Linux 7.1	Chapter 12, “Linux installation steps” on page 399

8.2.1 Install the operating system

The operating system required for each server, as well as the required patch level and installed utilities, can be determined by:

- Visiting the IBM WebSphere Application Server prerequisites site:

  `http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html`

- Reviewing the prereq.properties file located in the root directory of the WebSphere Application Server installation archive package.

Install the required operating system onto each server comprising the WebSphere Application Server solution. For further details, see the examples provided in the installation steps chapters (listed in Table 8-2 on page 140), as well as the operating system product documentation.

The following is a list of the high-level installation and configuration tasks common to all operating systems used in a WebSphere Application Server solution:

1. Installation of operating system software.
2. Configuration to meet component requirements:
   a. Install any required patches.
   b. Install any required packages/filesets.
   c. Install any required third-party tools used to install and/or execute each of the components on that server.

8.2.2 Install and configure Web server

The Web servers supported can be determined by visiting the IBM WebSphere Application Server prerequisites site:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

Install the Web server onto the designated Web server machine of your WebSphere Application Server solution. For further details, see the examples provided in the installation steps chapters (listed in Table 8-2 on page 140), as well as the Web server product documentation.

The following is a list of the high-level installation and configuration tasks common to all Web servers used in a WebSphere Application Server solution:

1. Pre-installation tasks:
   a. Create operating system account under which the Web server will run.
   b. Check that IP ports required by the Web server are not already in use.
2. Installation of Web server software.
3. Verification:
   a. Use a Web browser to issue a request for the Web server home page.

8.2.3 Install and configure the database server and client

The database servers and clients supported can be determined by visiting the IBM WebSphere Application Server prerequisites site:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

The database server software needs to be installed onto the machine that is designated as the database server. It will be used to host the WebSphere administrative repository database.

The database client software only needs to be installed onto each machine hosting WebSphere Application Server that will be configured to access a remote database server. Installation of the database client is not required if the WebSphere Application Server is only installed on the same server as the database server.

Database server

Install the database server onto the designated database server machine of your WebSphere Application Server solution. For further details, see the examples provided in the installation steps chapters (listed in Table 8-2 on page 140), as well as the database server product documentation.

The following is a list of the high-level installation and configuration tasks common to all database servers used in a WebSphere Application Server solution:

1. Pre-installation tasks:
   a. Check that the file system(s) used to store the database server and databases has sufficient free space.
   b. Check that IP ports required by the database server instance or listener are not already in use.
2. Installation of the database server.
3. Configuration:
   a. Configure TCP/IP connectivity of the database server.
   b. Configure database server JDBC 2.0 access (required by WebSphere Application Server V4.0).
   c. Configure the database server to be restarted automatically on system restart.
4. Verification:
   a. Check TCP/IP connectivity of the database server.
   b. Check JDBC 2.0 access to the database server.

Database client

Install the database client onto each machine in your solution that will host a WebSphere Application Server instance and access a remote database server. For further details, see the examples provided in the installation steps chapters (listed in Table 8-2 on page 140), as well as the database client product documentation.

The following is a list of the high-level installation and configuration tasks common to all database clients used in a WebSphere Application Server solution:

1. Pre-installation tasks.
2. Installation of database client.
3. Configuration:
   a. Configure TCP/IP connectivity of the remote database server.
   b. Configure JDBC 2.0 access to the remote database server (required by WebSphere Application Server V4.0).
4. Verification:
   a. Check TCP/IP connectivity of the remote database server.
   b. Check JDBC 2.0 access to the remote database server.

8.2.4 Create administrative repository database

The following is a list of the high-level steps required to create a database and configure it to meet WebSphere requirements:

1. Create WebSphere administrative database.
2. Tune the database settings to meet WebSphere requirements. The specifics of the tuning required varies depending upon the operating system and the database server being used.
3. Create the database account that WebSphere will use to access the database.
4. Check TCP/IP connectivity to the database using the access account.

For further details, see the examples provided in the installation steps chapters (listed in Table 8-2 on page 140), as well as the WebSphere Application Server online InfoCenter documentation.

8.2.5 Customize prerequisite checking

If you need to customize prerequisite checking, then perform one of the following procedures:

- Download the most current prereq.properties file from the Web site:

  `http://www.ibm.com/software/webservers/appserv/tools.html`

  to a temporary directory on the machine onto which WebSphere Application Server is to be installed.

Note: If the most current prereq.properties does not support the newer component level you wish to install, you must proceed with the next option to disable prerequisite checking.

- Edit the prereq.properties file to change one or more of the WebSphere prerequisite checker settings before installing WebSphere Application Server:
  a. Copy the prereq.properties file from the installation root directory to a temporary directory.
  b. Edit the file by changing the value of the `prereq_checker` parameter from 1 to 0.

In either case, pass the prereq.properties path to the installation program by using its `-prereqfile` argument, for example, on Windows 2000:

```
setup.exe -prereqfile <tmp>\prereq.properties
```

8.3 Perform WebSphere installation

The following table shows a breakdown of where each WebSphere component (server or HTTP plug-in) needs to be installed for possible architectures resulting from your installation planning stage.

Table 8-3 WebSphere component installation by architecture

Software Component	Server 1	Server 2	Server 3
Web server and database server located on the same server as WebSphere	Application, database and Web server 1. Install both WebSphere and the WebSphere HTTP plug-in	-	-
Remote Web server. Database server located on the same server as WebSphere.	Application and database server Install WebSphere	Web server Install the WebSphere HTTP plug-in	-
Remote Web server. Remote database server.	Database server -	Web server Install the WebSphere HTTP plug-in	Application server Install WebSphere

8.3.1 Install WebSphere

Install the WebSphere Application Server software using the following high-level steps.

Pre-installation tasks

1. Check that IP ports to be used by WebSphere Application Server are not already in use.

> **Note:** If the standard ports are already in use, or you choose to run on non-standard ports, you will need to reconfigure WebSphere during the post-installation stage to use non-standard ports.

2. If the Web server is located on the same machine as the WebSphere Application Server, then the Web server processes must be stopped prior to installing WebSphere. The installation will update the Web server configuration to support the WebSphere HTTP plug-in.

Installation of WebSphere Application Server software

Three different methods can be used to install the WebSphere Application Server software.

Install WebSphere using interactive (GUI) mode

For further details, see the examples provided in the installation steps chapters (listed in Table 8-2 on page 140), as well as the WebSphere Application Server online InfoCenter documentation.

Install WebSphere using non-interactive (silent) mode

For further details, see the examples provided in the installation steps chapters (listed in Table 8-2 on page 140), as well as the WebSphere Application Server online InfoCenter documentation.

Install identical configuration on multiple machines

This is an extension of the silent installation method.

1. Perform a silent installation on the first machine by customizing the default response file and executing the installation program at the command line, supplying the response file as a command-line parameter. See the installation steps chapters (listed in Table 8-2 on page 140) for specific instructions on performing a silent installation on a variety of platforms. The customized response file will be used to install WebSphere Application Server from the command line on the other machines you want to configure.
2. Start the WebSphere Application Server, open the administrative console, and configure your system.

3. Export an XML file that details the WebSphere Application Server configuration existing on the first machine. You can export the XML file by either:
   - Selecting **Console > Export to XML...** from the administrative console main menu.
   - Using the XMLConfig tool. See 23.4, “Introducing XMLConfig” on page 923 for more information about using the XMLConfig tool.

   The XML file will be imported and used to configure WebSphere Application Server on the other machines you want to configure.
4. Copy your custom response file to the root directory on the machine you want to configure next.
5. Copy the exported XML file to the machine you want to configure next.
6. Edit the exported XML file to include variable substitutes. This enables each server to specify a particular variable unique to that machine, such as the node name. See 23.4, “Introducing XMLConfig” on page 923 for more information.
7. Install WebSphere Application Server by performing a silent installation.
8. Import the configuration stored in the XML file to the new application server.

Verification of installation

1. Check the installation log for any errors.
2. Check that the WebSphere administrative server configuration file (admin.config) contains the following details:
   - Correct database connection settings.
   - Database JDBC classes have been added to the required path(s).
3. Start up the WebSphere administrative server to confirm that the WebSphere environment and administrative repository database connectivity are correct.
4. Start up the test application server (Default Server) installed by default. This application server provides servlets that enable you to check whether all the components of a WebSphere Application Server environment (application servers, data sources, servlets, EJBs, and so on) are working correctly.
5. Regenerate the WebSphere HTTP plug-in settings file (plugin-cfg.xml) to include the test application server settings.
6. Copy the WebSphere HTTP plug-in settings file to the Web server machine (if a separate machine), restart the Web server and then issue requests for servlets contained within the test application server.

8.3.2 Install WebSphere plug-in

Install the WebSphere HTTP plug-in software using the following high-level steps:

Pre-installation tasks

1. The Web server processes must be stopped prior to installing the WebSphere HTTP plug-in. The installation will update the Web server configuration to support the plug-in.

Installation of WebSphere plug-in

Use the same methods described in 8.3.1, “Install WebSphere” on page 145 to install the WebSphere HTTP plug-in.

Verification of installation

1. Check the installation log for any errors.
2. Check that the Web server configuration file has been updated to support the plug-in library.
3. Restart the Web server to ensure that it executes and loads the plug-in library successfully.

8.4 Perform post-installation tasks

Some or all of the following tasks may need to performed, based upon the requirements of your WebSphere solution:

- Configure database access
- Configure for common administrative database
- Configure for non-standard ports
- Run administrative server in the background (UNIX only)
- Configure for non-root user (UNIX only)
- Configure for multiple NICs
- Encrypt communication between the plug-in and WebSphere

8.4.1 Configure database access

In most cases (see below for special cases), the administrative database tables are created and then populated with a default configuration, when you install WebSphere Application Server. During product installation, you must provide information regarding the administrative database you wish the installation's administrative server to use.

Settings in the administrative configuration file (admin.config) record the database connection details, and determine whether the database tables are created and/or populated with the default configuration.

In some cases, the database tables are not created automatically during product installation, or the tables are not populated with the default configuration. If you need to, you can trigger the actions manually by setting properties in the administrative configuration file and then restarting the administrative server.

Oracle database on Solaris or AIX

If you are using WebSphere Application Server V4.0, Advanced Edition on Solaris and AIX operating systems with an Oracle database as the administrative database, note the following. In those cases, the Default Server is not created when you start the administrative server for the first time after installing the product. To create Default Server and other default resources set:

```
install.initial.config=true
```

in the administrative configuration file (admin.config) and then restart the administrative server.

Oracle database on all platforms

If you use an Oracle database user ID other than EJSADMIN (default value assigned during the WebSphere Application Server installation), you will need to edit the administrative configuration file (admin.config) file:

1. Ensure that the following user ID and schema settings reflect your actual database user ID and password:

   ```
   com.ibm.ejs.sm.adminServer.dbuser=your_user_name
   com.ibm.ejs.sm.adminServer.dbSchema=your_user_name
   ```

2. Restart the administrative server.

Using a remote DB2 administrative database

If the administrative server is going to store its administrative data in a remote DB2 database, you need to catalog the remote database. See your database product documentation for instructions.

Note: Setting the `dbserverName` and `dbportNumber` settings for the data source (flags in the administrative configuration file) does not work currently for the case of a remote DB2 database, for example, if you have the application server and the DB2 client installed on one machine, and are trying to connect the client to the DB2 server on another machine.

This problem may be overcome in a subsequent WebSphere Application Server fixpak. Be sure to consult the release notes for each fixpak to see whether the problem is fixed.

8.4.2 Configure for a common administrative database

If installing into an existing administrative domain we must edit admin.config so that the install.initial.config setting is set to false. Why? If true, when the administrative server is started on the new node, it will regenerate the settings of the initial configuration in the administrative database, overwriting existing settings such as virtual hosts.

For each new node sharing an existing administrative database, ensure the database tables are not created and the default resources are not installed a second time:

1. Edit the admin.config file and set the values of the following settings as shown:

```
install.initial.config=false
com.ibm.ejs.adminServer.createTables=false
```

Configuring the second through Nth machine in a cluster

Do not create the database tables or populate them with the default configuration. When you are setting up a cluster of administrative nodes, only the WebSphere Application Server product installation needs to install the default configuration:

1. On the first machine, ensure that the database tables have been created and populated with the configuration for the default resources. In the administrative configuration file, this means setting the following values:

```
install.initial.config=true
com.ibm.ejs.adminServer.createTables=true
```

2. On all subsequent machines sharing the same administrative database, ensure the database tables are not created and the default resources are not installed a second time:

```
install.initial.config=false
com.ibm.ejs.adminServer.createTables=false
```

3. If you have already installed the second (or later) machine, with install.initial.config set to true, then the problem should work itself out after all machines in the cluster have been shut down and started again. Until then, you might notice the following problems:
   - Absence of the WebSphere administrative domain resource in the tree view of the administrative console, or other peculiarities in the administrative topology.
   - Administrative server on the second or later machine will throw exceptions when you first start it.

8.4.3 Configure for non-standard ports

Although WebSphere Application Server uses default port numbers, these can be modified to allow different ports to be used.

Use the following guidelines when deciding upon non-standard port numbers:

1. Ports can range from 1024 to 64000. Choose a port that does not conflict with existing ports in use.
2. To check ports in use:
   - Use the **`netstat`** command (**`netstat -a`**)
   - View the /etc/services file on UNIX
   - View the %windir%\system32\drivers\etc\services file on Windows 2000
3. Ports must be unique in the scope of each physical host. That is, two servers on the same machine cannot have the same port values.
4. The same port numbers can be used for servers running on different physical hosts. That is, the administrative server on Machine A can use bootstrap port 900, while administrative servers on other machines use this same number.
5. For administrative servers, pick port numbers above 1024 if not using the default port and running as non-root.

Bootstrap port

Changing the bootstrap port from its default affects the administrative client. Use the following steps to reconfigure WebSphere to use a non-standard bootstrap port:

1. Edit the admin.config file to add or amend the following setting:

   ```
   com.ibm.ejs.sm.adminServer.bootstrapPort=<port>
   ```

   where `<port>` is the port number to use for the bootstrap port.
2. Save the changes.

3. Restart the administrative server.

Note: When starting the Java administrative console, you must specify the new port number:

- On UNIX-based operating systems:

```
# ./adminclient.sh hostname port
```

- On Windows-based operating systems:

```
C:\> adminclient hostname port
```

Remember to configure firewalls to allow traffic to pass for each assigned port. For security, try to minimize ports.

Location Service Daemon (LSD) port

Use the following steps to reconfigure WebSphere to use a non-standard LSD port:

1. Edit the admin.config file to add or amend the following setting:

   ```
   com.ibm.ejs.sm.adminServer.lsdPort=<port>
   ```

   where <port> is the port number to use for the LSD port.
2. Save the changes.
3. Restart the administrative server.

8.4.4 Run administrative server in the background (UNIX only)

To run the server as a background process:

1. Add the following parameter to the admin.config file:

   ```
   com.ibm.ejs.sm.adminServer.processPriority=<number>
   ```

   where <number> is 28 for AIX and 24 for Solaris. It could also be the default priority of the non-root user ID under which the administrative server is running.
2. Save the changes.
3. Edit the properties of each application server associated with this administrative server. You will need to modify the process priority of each server to be the same process priority (number) you assigned to the administrative server.
4. Restart the administrative server.

8.4.5 Configure for non-root user (UNIX only)

Although WebSphere Application Server processes runs under the root ID by default, it can be reconfigured to use another (non-root) ID. However, the new operating system user and group must exist on the machine where the server is to run before the server is started. This user and group must also be assigned the necessary operating system privileges for performing operations such as creating output files on the local file system.

Running administrative servers as non-root

To configure the WebSphere administrative server to run under a non-root ID, perform the following steps:

1. Change the permissions of the product installation directories so that the non-root ID has access, using one of the following methods:
   a. Change the ownership of all files and directories under <WAS_HOME> to the user and group that you want the administrative server to "run as".
   b. Change the ownership of only the following files and directories to the user and group that you want the administrative server to run under:
      - <WAS_HOME>/bin/*
      - <WAS_HOME>/config/*
      - <WAS_HOME>/etc/*
      - <WAS_HOME>/installedApps/*
      - <WAS_HOME>/logs/*
      - <WAS_HOME>/properties/*
      - <WAS_HOME>/tranlog/*
      - <WAS_HOME>/temp/*

      Make sure that the user has write and execute privileges.
2. Change the bootstrap port of the administrative server to a value greater than or equal to 1024:
   a. Edit the administrative configuration (admin.config) file.
   b. Add the following setting:

      ```
      com.ibm.ejs.sm.adminServer.bootstrapPort=<port>
      ```

      where `<port>` is the port number chosen for the bootstrap port.

Note: Changing the bootstrap port affects the administrative clients that connect to the server.

3. If running the DB2 database server, update the new user's environment file to include setup of the DB2 environment:
   a. Edit the user's environment file.
   b. Add the following lines:

```
# Setup DB2 environment for root user.
if [ -f /home/<db2_instance_owner>/sqllib/db2profile ] ; then
. /home/<db2_instance_owner>/sqllib/db2profile
fi

# Force DB2 to use JDBC 2.0.
if [ -f /home/<db2_instance_owner>/sqllib/java12/usejdbc2 ] ; then
. /home/<db2_instance_owner>/sqllib/java12/usejdbc2
fi
```

4. Restart the administrative server.

Important: On Solaris, the **ndd** commands in the administrative server startupServer.sh script need to be commented out unless you are running as root. If an **ndd** command is executed by a non-root user, the following error message will be issued to stdout or stderr:

```
Operation failed, Not owner.
```

The **ndd** command is for dynamically adjusting certain IP stack parameters. It attempts to operate on operating system level kernel device settings, which can only be performed by root. This causes the error message.

The workaround is to either run the administrative server as root or edit the startupServer.sh script, commenting out the **ndd** command. It is still strongly recommended that the changes to the TCP parameters that the **ndd** command be made by root on all machines running the application server and Web server (in case they are not the same box).

Problems caused by incorrect non-root configuration

The following problems indicate the need to review the above instructions to ensure that your configuration is correct for running as non-root.

- The following error message is displayed when the administrative server is started as non-root on Solaris:

  ```
  $ ./startupServer.sh operation failed,Not owner
  ```

  The likely cause is that the **ndd** command is being executed as non-root, as explained above.

- The following error message is displayed when starting the server as non-root:

```
NMSV0011E: Unable to start Bootstrap Server
```

The likely causes are:

a. The bootstrap port is already in use.

   Ensure that no servers or other processes are already using the bootstrap server port.

b. The bootstrap port value is less than 1024.

   Only the root account can use port numbers lower than 1024.

Running Java administrative consoles as non-root

To configure the WebSphere Java administrative console to run under a non-root ID, you need to change the ownership of the following directories and files to the user and group that you would like the console under:

- <WAS_HOME>/bin
- <WAS_HOME>/properties/sas.client.props

Running application servers as non-root

To configure a WebSphere Application Server to run under a non-root ID, perform the following steps:

1. Start the WebSphere administrative server under the root ID.
2. Start the Java administrative console.
3. In the tree view, locate and click the application server to display its properties.
4. In the Advanced properties, modify the user ID and group ID to be the user and group under which the application server is to run.
5. In the General properties, modify the standard output and standard error log paths to refer to directories for which the new ID has access.
6. Start the application server, using the new ID.

8.4.6 Configure for multiple NICs

WebSphere uses the value returned by the UNIX/Windows **hostname** command as its node name. This will generally be the primary NIC (network interface card) on the machine, which is a problem if you need it to bind to the secondary NIC. This can be worked around by forcing WebSphere to use a specific node name by including the following properties in the admin.config file before WebSphere is first started:

1. com.ibm.ejs.sm.adminServer.nodeName=<required node name>

2. You may also need to force the IP address to which the AdminServer binds and includes in its generated IOR. This setting should be set to the real IP address of the NIC you want to access the AdminServer through. Either specify as -D argument to the com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs property in admin.config, or try as a stand-alone property itself, that is, com.ibm.CORBA.LocalHost=<NIC IP address>
3. You may need to force the short host name to which the administrative server binds and includes in its generated IOR. This setting should be set to the real short host name of the NIC you want to access the administrative server through. Either specify as -D argument to the com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs property in admin.config, or try as a stand-alone property itself, that is, com.ibm.CORBA.LSDHostName=<NIC short hostname>

Note: Try the node name setting first. Settings 2 and 3 should only be used if the problem still exists.

If WebSphere has been started previously, then you will need to dump the WebSphere administrative database and allow it to be rebuilt so that it contains the new settings specified in the admin.config file.

8.4.7 Encrypt communication between plug-in and WebSphere

Each transport of a WebSphere V4.0 Web container can be configured to support SSL encryption of the HTTP communication (HTTPS). The SSL functionality can be run in one of two modes:

- Client authentication disabled

  Only the WebSphere Application Server must send a certificate to the client (WebSphere HTTP plug-in) in order to authenticate itself during SSL handshaking. This configuration requires the following tasks:

  a. Creation of a JKS format keyfile (key store) for use by the WebSphere transport.
  b. Creation of a self-signed certificate stored in the JKS keyfile.
  c. Configuration of a new (or reconfiguration of an existing) transport to use SSL encryption and the above keyfile for the required keyfile settings.
  d. Creation of a CMS format keyfile (key store) for use by the plug-in.
  e. Import of the server's self-signed certificate as a trusted CA (certificate authority) within the plug-in's keyfile.

- Client authentication enabled

This is an extension of the client authentication disabled case. In this case, the client (plug-in) is also required to send a certificate to the server in order to authenticate itself during SSL handshaking. This configuration requires the following additional tasks to the client authentication disabled case:

a. Creation of self-signed certificate in the plug-in's CMS keyfile.

b. Import of the client's self-signed certificate as a trusted CA (certificate authority) into the server's keyfile.

c. Reconfiguration of the transport's SSL settings to specify the JKS keyfile as the trustfile, as well as enabling of the client authentication option.

In either case, after performing the above tasks, the Web server plug-in configuration must then be updated by performing these remaining tasks:

1. Regenerate the Web server plug-in configuration file.
2. Restart the Web server.

8.5 Uninstalling WebSphere

To uninstall WebSphere Application Server, stop all WebSphere processes, then execute the uninstall script located in the root directory of WebSphere installation:

- On Windows 2000 systems:

```
C:\> cd <WAS_HOME>
C:\> uninstWAS40.exe
```

- On UNIX and Linux systems

```
# cd <WAS_HOME>
# ./uninstall.sh
```

Note: Uninstalling WebSphere Application Server does not uninstall any of the other components (Web server, database server and database client). These will need to be uninstalled separately. See the product documentation for details.

8.6 Example scenarios

In this section we describe a number of example scenarios that illustrate how a number of common WebSphere Application Server topologies may be installed.

8.6.1 Topologies

This section describes the tasks necessary to install and configure a number of example configurations of WebSphere. Many of the same tasks are used in the different configurations.

Table 8-4 Example installation scenarios

Scenario	Description
A	WebSphere and the database server are installed on one server (server #1). No Web server is installed, since the embedded HTTP transport of the WebSphere Web container is used.
B	WebSphere, Web server and database server are all installed on one server (server #1). Web server to WebSphere communication are set up to use HTTP.
C	This scenario is the same as scenario B, except that the HTTP communication between the HTTP plug-in and WebSphere is set up to use SSL encryption with client authentication enabled.
D	Web server and WebSphere HTTP plug-in are installed on one server (server #2), with WebSphere and the database server installed on another server (server #1). Web server to WebSphere communications are set up to use HTTP.
E	This scenario is the same as scenario D, except that the HTTP communication between the HTTP plug-in and WebSphere is set up to use SSL encryption with client authentication disabled.
F	This scenario is the same as scenario E, except that the HTTP communication between the HTTP plug-in and WebSphere is set up to use SSL encryption with client authentication enabled.
G	This scenario is the same as scenario F, except that the Web server is configured to use SSL encryption for all Web requests.
H	Web server and WebSphere are both installed on two servers (server #1 and server #2), with both configured to share the one administrative repository database. This is an example of a shared administrative repository database configuration.

8.6.2 Scenario A - single server with embedded Web server

This example describes the most basic configuration of WebSphere. WebSphere and the database server are installed on the same server and the Web container's embedded HTTP server is used in place of a stand-alone HTTP server.

- WebSphere V4.0 and the database server are installed on Server 1.
- 1 x HTTP transport is configured for the default Web container in WebSphere.
- Web applications are accessed directly through the HTTP transport of the Web container.

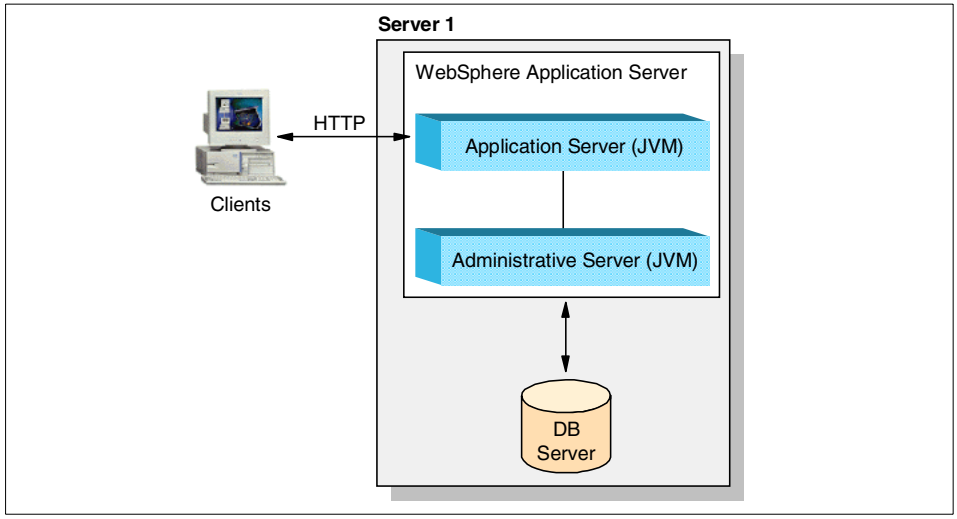


Figure 8-1 Scenario A components

Installation and configuration tasks

This section summarizes the high-level steps that should be performed in order to install and configure Scenario A.

Table 8-5 High-level installation steps: example scenario A

Steps	Server 1 (DB server, WebSphere)
Step 1	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites
Step 2	Database server 1. Install the database server 2. Configure TCP/IP connectivity to the database server 3. Configure JDBC 2.0 access to the database server 4. Create the WebSphere repository database 5. Tune the database to WebSphere requirements 6. Verify the configuration

Steps	Server 1 (DB server, WebSphere)
Step 3	WebSphere Application Server 1. Customize prerequisite checking (optional) 2. Install WebSphere Application Server 3. Check the installation log 4. Check the admin.config configuration file 5. Start the WS AdminServer process 6. Start WebSphere Default Server 7. Verify the configuration

For detailed instructions on the high-level steps outlined in this section, refer to the following sources:

- Installation steps chapters (listed in Table 8-2 on page 140). These chapters include detailed instructions on a number of platforms (Windows 2000, AIX, Linux, Solaris) for each of the high-level tasks used for this example.
- Product installation guides for the appropriate platform.

8.6.3 Scenario B - single server with stand-alone Web server

This example describes a common basic configuration of WebSphere. WebSphere, database server and Web server are installed on the same server. Communication between the Web server and WebSphere is provided by the WebSphere HTTP plug-in.

This scenario is the same as scenario A, but with the following additional elements:

- The Web server is installed on the same server as WebSphere (Server 1).
- The Web server is configured for HTTP access on port 80.
- The WebSphere HTTP plug-in to HTTP transport communication is configured to use HTTP.
- Web applications are accessed through the stand-alone Web server.

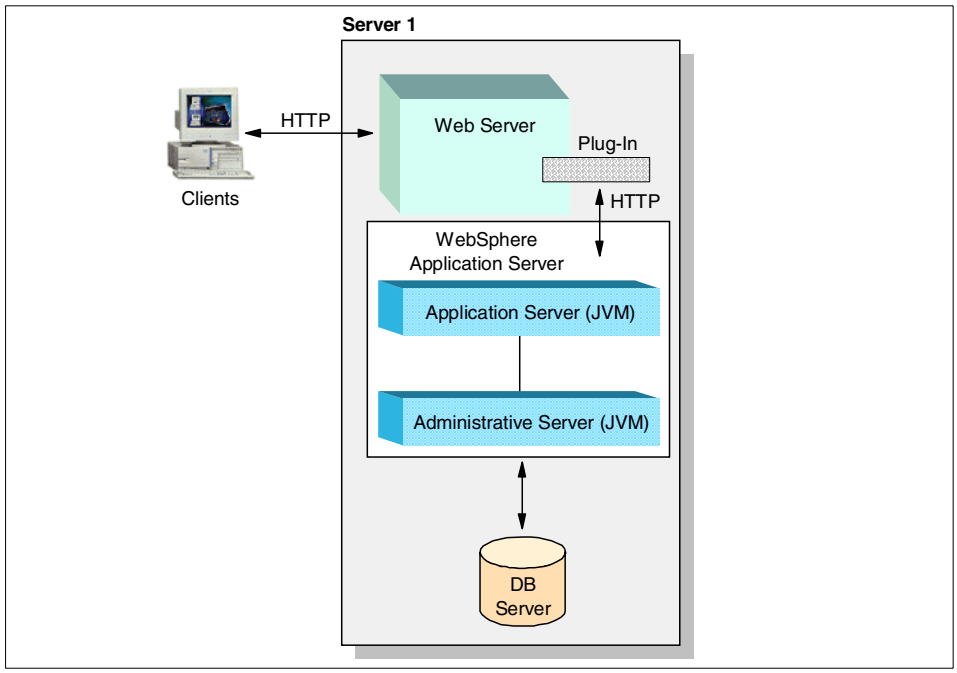


Figure 8-2 Scenario B components

Installation and configuration tasks

This section summarizes the high-level steps that should be performed in order to install and configure Scenario B.

Table 8-6 High-level installation steps: example scenario B

Steps	Server 1 (Web server, DB server, WebSphere)
Step 1	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites
Step 2	Web server 1. Create an account to run Web server under 2. Install the Web server 3. Verify the configuration

Steps	Server 1 (Web server, DB server, WebSphere)
Step 3	Database server 1. Install the database server 2. Configure TCP/IP connectivity to the database server 3. Configure JDBC 2.0 access to the database server 4. Create the WebSphere repository database 5. Tune the database to WebSphere requirements 6. Verify the configuration
Step 4	WebSphere Application Server 1. Customize prerequisite checking (optional) 2. Install WebSphere Application Server 3. Check the installation log 4. Check the admin.config configuration file 5. Start the WS AdminServer process 6. Start WebSphere Default Server 7. Regenerate the plug-in configuration settings file 8. Verify the configuration

For detailed instructions on the high-level steps outlined in this section, refer to the following sources:

- Installation steps chapters (listed in Table 8-2 on page 140). These chapters include detailed instructions on a number of platforms (Windows 2000, AIX, Linux, Solaris) for each of the high-level tasks used for this example.
- Product installation guides for the appropriate platform.

8.6.4 Scenario C - single server with stand-alone Web server and SSL encryption of all communication

This example describes an extension to the basic configuration described in scenario B. In this example, SSL is used to encrypt the communication between the HTTP plug-in and the HTTP transport, with both the plug-in (client) and WebSphere (server) having to authenticate themselves during SSL handshaking. In addition, the Web server is configured to require SSL encryption for all Web requests.

This scenario is the same as scenario B, but with the following additional elements:

- HTTPS transport is configured on the WebSphere Web container on port 9081.
- The WebSphere HTTP plug-in is configured to support HTTPS transport for communication with Web container.
- The Web server is configured to use SSL encryption for all Web requests.

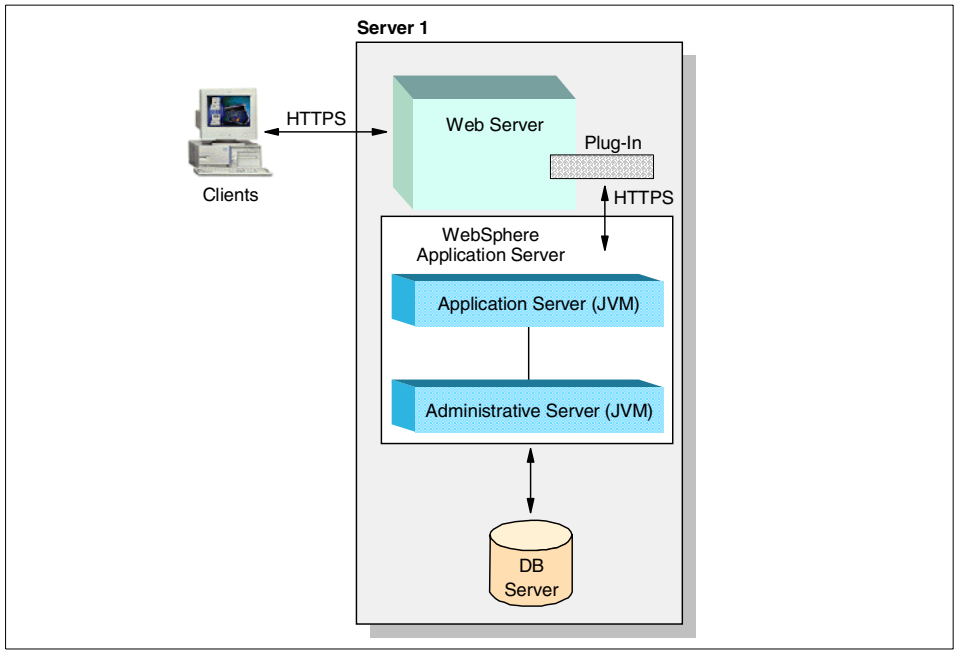


Figure 8-3 Scenario C components

Installation and configuration tasks

This section summarizes the high-level steps that should be performed in order to install and configure Scenario C.

Table 8-7 High-level installation steps: example scenario C

Steps	Server 1 (Web server, DB server, WebSphere)
Step 1	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites

Steps	Server 1 (Web server, DB server, WebSphere)
Step 2	Web server 1. Create an account to run Web server under 2. Install the Web server 3. Verify the configuration
Step 3	Database server 1. Install the database server 2. Configure TCP/IP connectivity to the database server 3. Configure JDBC 2.0 access to the database server 4. Create the WebSphere repository database 5. Tune the database to WebSphere requirements 6. Verify the configuration
Step 4	WebSphere Application Server 1. Customize prerequisite checking (optional) 2. Install WebSphere Application Server 3. Check the installation log 4. Check the admin.config configuration file 5. Start the WS AdminServer process 6. Start WebSphere Default Server 7. Create the certificate keystore 8. Create or import certificates used for SSL handshaking 9. Create new HTTP transport for WebSphere Default Server 10. Configure transport to use SSL encryption 11. Regenerate the Web server plug-in configuration file 12. Verify the configuration
Step 5	WebSphere HTTP plug-in 1. Create the certificate keystore 2. Create or import certificates used for SSL handshaking
Step 6	WebSphere Application Server 1. Start WebSphere Default Server 2. Verify the configuration

For detailed instructions on the high-level steps outlined in this section, refer to the following sources:

- Installation steps chapters (listed in Table 8-2 on page 140). These chapters include detailed instructions on a number of platforms (Windows 2000, AIX, Linux, Solaris) for each of the high-level tasks used for this example.
- Product installation guides for the appropriate platform.

8.6.5 Scenario D - two servers with remote Web server

This example describes a basic networked configuration. WebSphere and the database server are installed on the same server (Server 1), but the stand-alone Web server and HTTP plug-in are installed on a second “remote” server (Server 2). Communication between the plug-in and the HTTP transport is configured to use HTTP.

This scenario is the same as scenario B, but with the following additional elements:

- The Web server and the WebSphere HTTP plug-in are installed on a separate server (Server 2).

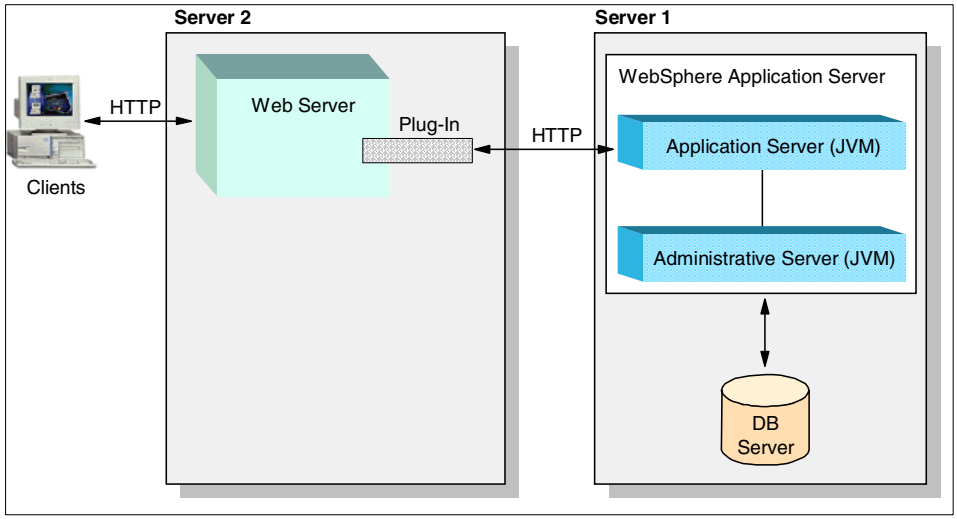


Figure 8-4 Scenario D components

Installation and configuration tasks

This section summarizes the high-level steps that should be performed in order to install and configure Scenario D.

Table 8-8 High-level installation steps: example scenario D

Steps	Server 1 (DB server, WebSphere)	Server 2 (Web server)
Step 1	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites
Step 2		Web server 1. Create an account to run Web server under 2. Install the Web server 3. Verify the configuration
Step 3	Database server 1. Install the database server 2. Configure TCP/IP connectivity to the database server 3. Configure JDBC 2.0 access to the database server 4. Create the WebSphere repository database 5. Tune the database to WebSphere requirements 6. Verify the configuration	

Steps	Server 1 (DB server, WebSphere)	Server 2 (Web server)
Step 4	WebSphere Application Server 1. Customize prerequisite checking (optional) 2. Install WebSphere Application Server 3. Check the installation log 4. Check the admin.config configuration file 5. Start the WS AdminServer process 6. Start WebSphere Default Server 7. Regenerate the Web server plug-in configuration file 8. Verify the configuration	
Step 5		WebSphere HTTP plug-in 1. Install the WebSphere HTTP plug-in 2. Replace the default plugin-cfg.xml with the working copy generated on Server 1
Step 6		Web server 1. Restart the Web server 2. Verify the configuration

For detailed instructions on the high-level steps outlined in this section, refer to the following sources:

- Installation steps chapters (listed in Table 8-2 on page 140). These chapters include detailed instructions on a number of platforms (Windows 2000, AIX, Linux, Solaris) for each of the high-level tasks used for this example.
- Product installation guides for the appropriate platform.

8.6.6 Scenario E - two servers with remote Web server and SSL encrypted (client authentication disabled) HTTP transport

This example describes an extension to the basic networked configuration described in scenario D suitable for firewall environments. In this example, SSL is used to encrypt the communication between the HTTP plug-in and the HTTP transport of the Web container. SSL is configured to require the HTTP transport (server) to authenticate itself to the plug-in (client).

This scenario is the same as scenario D, but with the following additional elements:

- Configure a second transport on Server 1 Web container, with SSL encryption enabled (with client authentication option disabled).
- The plug-in is configured to support the remote HTTPS transport.

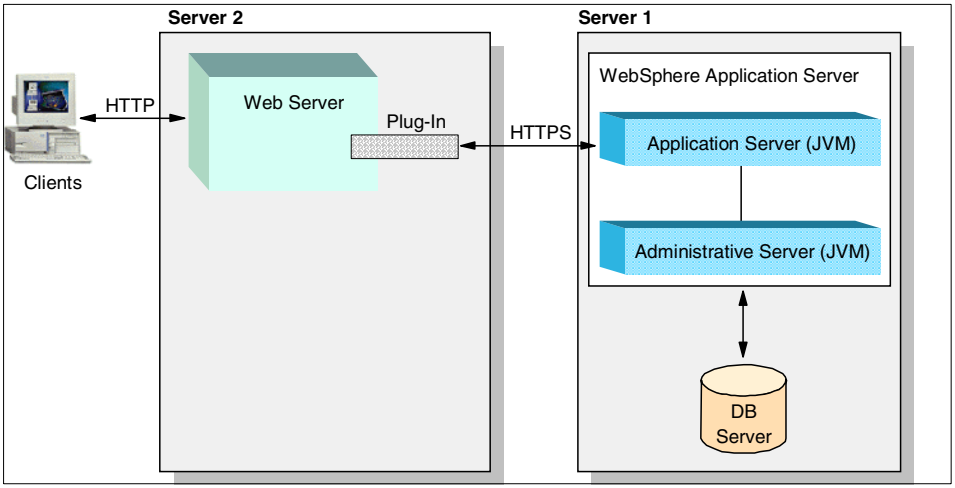


Figure 8-5 Scenario E components

Installation and configuration tasks

This section summarizes the high-level steps that should be performed in order to install and configure Scenario E.

Table 8-9 High-level installation steps: example scenario E

Steps	Server 1 (DB server, WebSphere)	Server 2 (Web server)
Step 1	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites
Step 2		Web server 1. Create an account to run Web server under 2. Install the Web server 3. Verify the configuration
Step 3	Database server 1. Install the database server 2. Configure TCP/IP connectivity to the database server 3. Configure JDBC 2.0 access to the database server 4. Create the WebSphere repository database 5. Tune the database to WebSphere requirements 6. Verify the configuration	

Steps	Server 1 (DB server, WebSphere)	Server 2 (Web server)
Step 4	WebSphere Application Server 1. Customize prerequisite checking (optional) 2. Install WebSphere Application Server 3. Check the installation log 4. Check the admin.config configuration file 5. Start the WS AdminServer process 6. Start WebSphere Default Server 7. Create the certificate keystore 8. Create or import certificates used for SSL handshaking 9. Create new HTTP transport for WebSphere Default Server 10. Configure transport to use SSL encryption 11. Regenerate the Web server plug-in configuration file 12. Verify the configuration	
Step 5		WebSphere HTTP plug-in 1. Install the WebSphere HTTP plug-in 2. Create the certificate keystore 3. Create or import certificates used for SSL handshaking 4. Replace the default plugin-cfg.xml with working copy generated on Server 1 5. Edit the plugin-cfg.xml to support remote HTTPS transport
Step 6		Web server 1. Restart the Web server 2. Verify the configuration

For detailed instructions on the high-level steps outlined in this section, refer to the following sources:

- Installation steps chapters (listed in Table 8-2 on page 140). These chapters include detailed instructions on a number of platforms (Windows 2000, AIX, Linux, Solaris) for each of the high-level tasks used for this example.
- Product installation guides for the appropriate platform.

8.6.7 Scenario F - two servers with remote Web server and SSL encrypted (client authentication enabled) HTTP transport

This example describes an extension to the secured networked configuration described in scenario E. In this example SSL is used to encrypt the communication between the HTTP plug-in and WebSphere, with both the plug-in (client) and WebSphere (server) having to authenticate themselves to the other.

This scenario is the same as scenario E, but with the following additional elements:

- Configure HTTPS transport on Server 1 to use client authentication.

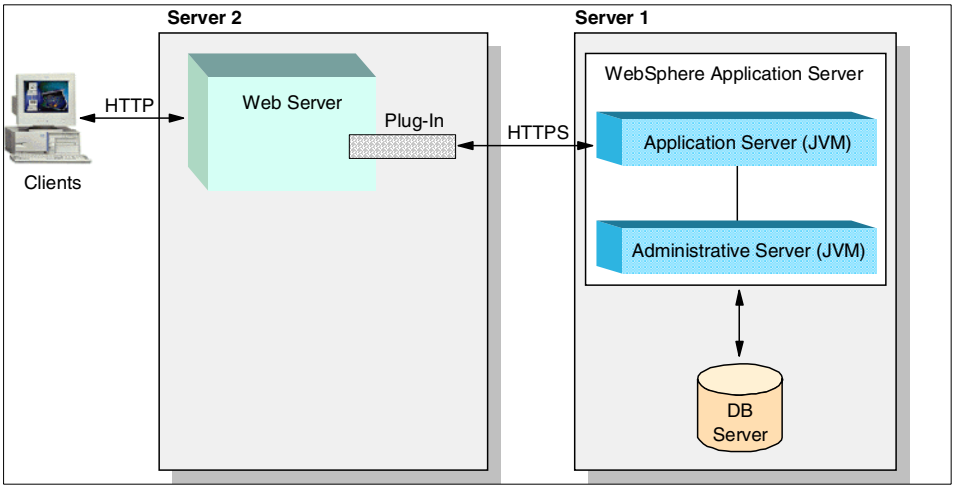


Figure 8-6 Scenario F components

Installation and configuration tasks

This section summarizes the high-level steps that should be performed in order to install and configure Scenario F.

Table 8-10 High-level installation steps: example scenario F

Steps	Server 1 (DB server, WebSphere)	Server 2 (Web server)
Step 1	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites
Step 2		Web server 1. Create an account to run Web server under 2. Install the Web server 3. Verify the configuration
Step 3	Database server 1. Install the database server 2. Configure TCP/IP connectivity to the database server 3. Configure JDBC 2.0 access to the database server 4. Create the WebSphere repository database 5. Tune the database to WebSphere requirements 6. Verify the configuration	

Steps	Server 1 (DB server, WebSphere)	Server 2 (Web server)
Step 4	WebSphere Application Server 1. Customize prerequisite checking (optional) 2. Install WebSphere Application Server 3. Check the installation log 4. Check the admin.config configuration file 5. Start the WS AdminServer process 6. Start WebSphere Default Server 7. Create the certificate keystore 8. Create or import certificates used for SSL handshaking 9. Create new HTTP transport for WebSphere Default Server 10. Configure transport to use SSL encryption 11. Regenerate the Web server plug-in configuration file 12. Verify the configuration	
Step 5		WebSphere HTTP plug-in 1. Install the WebSphere HTTP plug-in 2. Create the certificate keystore 3. Create or import certificates used for SSL handshaking 4. Replace the default plugin-cfg.xml with working copy generated on Server 1 5. Edit the plugin-cfg.xml to support remote HTTPS transport.
Step 6		Web server 1. Restart the Web server 2. Verify the configuration

For detailed instructions on the high-level steps outlined in this section, refer to the following sources:

- Installation steps chapters (listed in Table 8-2 on page 140). These chapters include detailed instructions on a number of platforms (Windows 2000, AIX, Linux, Solaris) for each of the high-level tasks used for this example.
- Product installation guides for the appropriate platform.

8.6.8 Scenario G - two servers with remote Web server and SSL encryption of all communication

This example describes an extension to the secured networked configuration described in scenario F. In this example, the Web server is configured to use SSL encryption for all Web requests.

This scenario is the same as scenario F, but with the following additional elements:

- The Web server is configured to use SSL encryption.
- The WebSphere virtual host is updated to support requests using the standard HTTP SSL (HTTPS) port 443 instead of the standard HTTP port 80.

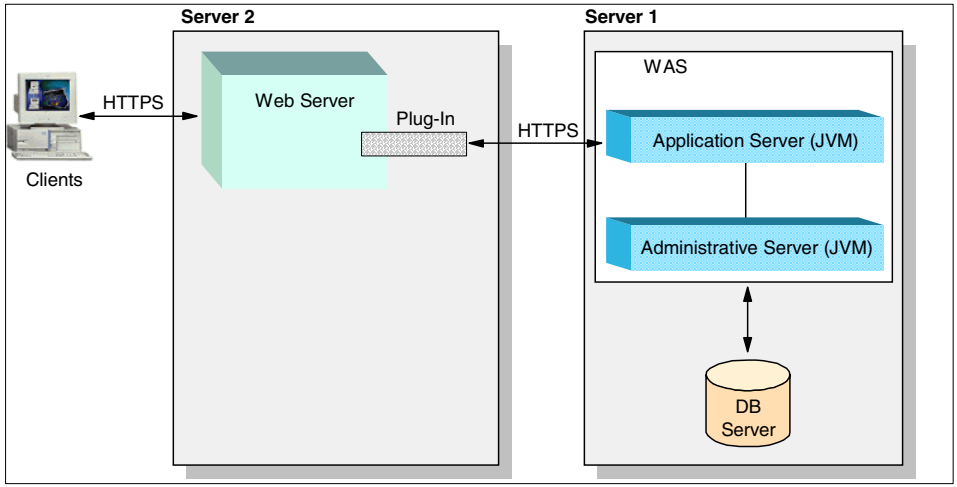


Figure 8-7 Scenario G components

Installation and configuration tasks

This section summarizes the high-level steps that should be performed in order to install and configure Scenario G.

Table 8-11 High-level installation steps: example scenario G

Steps	Server 1 (DB server, WebSphere)	Server 2 (Web server)
Step 1	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites
Step 2		Web server 1. Create an account to run Web server under 2. Install the Web server 3. Create the certificate keystore 4. Create or request certificate used for SSL handshaking of Web requests 5. Edit Web server configuration to enable SSL encryption for Web requests 6. Verify the configuration
Step 3	Database server 1. Install the database server 2. Configure TCP/IP connectivity to the database server 3. Configure JDBC 2.0 access to the database server 4. Create the WebSphere repository database 5. Tune the database to WebSphere requirements 6. Verify the configuration	

Steps	Server 1 (DB server, WebSphere)	Server 2 (Web server)
Step 4	WebSphere Application Server 1. Customize prerequisite checking (optional) 2. Install WebSphere Application Server 3. Check the installation log 4. Check the admin.config configuration file 5. Start the WS AdminServer process 6. Start WebSphere Default Server 7. Create the certificate keystore 8. Create or import certificates used for SSL handshaking 9. Create new HTTP transport for WebSphere Default Server 10. Configure transport to use SSL encryption 11. Regenerate the Web server plug-in configuration file 12. Verify the configuration	
Step 5		WebSphere HTTP plug-in 1. Install the WebSphere HTTP plug-in 2. Create the certificate keystore 3. Create or import certificates used for SSL handshaking 4. Replace the default plugin-cfg.xml with working copy generated on Server 1 5. Edit the plugin-cfg.xml to support remote HTTPS transport
Step 6		Web server 1. Restart the Web server 2. Verify the configuration

For detailed instructions on the high-level steps outlined in this section, refer to the following sources:

- Installation steps chapters (listed in Table 8-2 on page 140). These chapters include detailed instructions on a number of platforms (Windows 2000, AIX, Linux, Solaris) for each of the high-level tasks used for this example.
- Product installation guides for the appropriate platform.

8.6.9 Scenario H - two server administrative domain

This example describes the most basic WebSphere shared administrative repository configuration. Instances of WebSphere installed on two separate servers are configured to share the one administrative repository database.

- WebSphere, the database server and Web server are installed on Server 1.
- The WebSphere administrative repository database is created on Server 1.
- The database client is installed on Server 2 and configured to access the remote database server.
- WebSphere and the Web server are installed on Server 2.
- The WebSphere instance on Server 2 is configured to access the remote administrative database through the database client.
- 1 x HTTP transport is configured on each WebSphere instance.
- Web applications deployed to each WebSphere node are accessed through the Web server on that node.

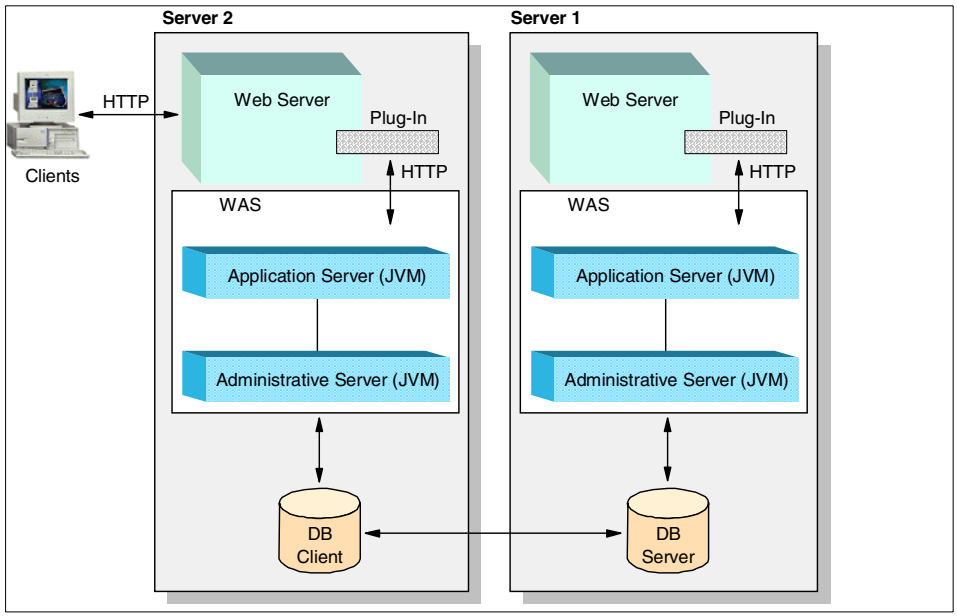


Figure 8-8 Scenario H components

Installation and configuration tasks

This section summarizes the high-level steps that should be performed in order to install and configure Scenario H.

Table 8-12 High-level installation steps: example scenario H

Steps	Server 1 (Web server, WebSphere, DB server)	Server 2 (Web server, WebSphere, DB client)
Step 1	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites	Operating system 1. Install the operating system 2. Install any required patches and packages as detailed in the WebSphere platform-specific prerequisites
Step 2	Web server 1. Create an account to run Web server under 2. Install the Web server 3. Verify the configuration	Web server 1. Create an account to run Web server under 2. Install the Web server 3. Verify the configuration

Steps	Server 1 (Web server, WebSphere, DB server)	Server 2 (Web server, WebSphere, DB client)
Step 3	Database server 1. Install the database server 2. Configure TCP/IP connectivity to the database server 3. Configure JDBC 2.0 access to the database server 4. Create the WebSphere repository database 5. Tune the database to WebSphere requirements 6. Verify the configuration	
Step 4	WebSphere Application Server 1. Customize prerequisite checking (optional) 2. Install WebSphere Application Server 3. Check the installation log 4. Check the admin.config configuration file 5. Start the WS AdminServer process 6. Start WebSphere Default Server 7. Regenerate the Web server plug-in configuration file 8. Verify the configuration	
Step 5	Web server 1. Restart the Web server 2. Verify the configuration	
Step 6		Database Client 1. Install database client 2. Configure TCP/IP access to the remote database server 3. Configure JDBC 2.0 access to the remote database server 4. Verify the configuration

Steps	Server 1 (Web server, WebSphere, DB server)	Server 2 (Web server, WebSphere, DB client)
Step 7		WebSphere Application Server 1. Customize prerequisite checking (optional) 2. Install WebSphere Application Server 3. Check the installation log 4. Check the admin.config configuration file 5. Start the WS AdminServer process 6. Start WebSphere Default Server 7. Regenerate the Web server plug-in configuration file 8. Verify the configuration
Step 8		Web server 1. Restart the Web server 2. Verify the configuration

For detailed instructions on the high-level steps outlined in this section, refer to the following sources:

- Installation steps chapters (listed in Table 8-2 on page 140). These chapters include detailed instructions on a number of platforms (Windows 2000, AIX, Linux, Solaris) for each of the high-level tasks used for this example.
- Product installation guides for the appropriate platform.

9

Windows 2000 installation steps

This chapter provides detailed procedures for installing, configuring, and verifying a number of the scenarios described in Chapter 8, “Installation approach” on page 127, for an environment consisting of the following components:

- Operating system - Windows 2000
- Database server - IBM DB2 UDB Enterprise Edition
- Web server - IBM HTTP Server
- Application server - WebSphere Application Server V4.0, Advanced Edition

The procedures described are intended to be used as working examples in conjunction with the product installation guides for all the possible values that may be unique within your runtime environment. This chapter is organized into the following sections:

- Planning
- Install Windows 2000
- Install Web server
- Install database server
- Install database client

- Install WebSphere Application Server V4.0, Advanced Edition
- Install WebSphere plug-in on remote Web server
- Install new WebSphere node into existing domain
- Configure WebSphere HTTP transport for SSL
- Install WebSphere Application Server - silent mode

9.1 Planning

This section defines the hardware and software used within the Windows 2000 environment to test a number of different WebSphere Application Server V4.0 configuration scenarios.

9.1.1 Hardware and software prerequisites

WebSphere Application Server V4.0, Advanced Edition has the following hardware and software requirements.

Hardware

- 200 MB diskspace (minimum) for WebSphere Application Server
- 350 MB diskspace (minimum) for IBM DB2 Enterprise Edition
- 50 MB diskspace (minimum) for IBM HTTP Server
- 135 MB diskspace (minimum) for TEMP directory
- Greater than 128 MB RAM

Software

- Microsoft Windows 2000 Server, Service Pack 1
- IBM WebSphere Application Server V4.0, Advanced Edition
- IBM DB2 Universal Database V7.2 FP4, Enterprise Edition for Windows
- IBM HTTP Server 1.3.19
- IBM GSKit 5.0.3.52.

Note: For further details on requirements, see the following documentation:

1. DB2 -

   http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winox2unix/support/v7pubs.d2w/en_main

2. WebSphere -

   http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

9.1.2 Software used in our test environment

We used the following software in our test environment:

- Microsoft Windows 2000 Server, Service Pack 1, Build 2195
- IBM WebSphere Application Server V4.0, Advanced Edition
- IBM DB2 Universal Database V7.2 FP4, Enterprise Edition for Windows
- IBM HTTP Server 1.3.19
- IBM GSKit 5.0.3.52 (included in IBM HTTP Server package)

Note: Other Web servers and database software may be used, as documented in the *Product Installation Guide*.

Product installation roots

The variables listed in Table 9-1 are used frequently throughout this redbook to represent the root installation directories of the software components.

Table 9-1 Product installation roots

Variable	Default value	Component
<WAS_HOME>	C:\WebSphere\AppServer	WebSphere
<plugin_install_path>	C:\WebSphere\AppServer	WebSphere HTTP plug-in
<db2_install_path>	C:\Program Files\SQLLIB	DB2 server DB2 client
<http_server_install_path>	C:\IBM HTTP Server	IBM HTTP Server

9.1.3 Hardware used in our test environment

This section describes the hardware used within our test WebSphere Application Server V4.0 environment on Windows 2000.

- Server 1

  - IBM PC 300PL, Model 65653BU
  - 1 633 MHz Pentium III CPU
  - 512 MB RAM
  - 19 GB hard disk
  - 1 IBM 10/100 EtherJet PCI Ethernet adapter
- Server 2
  - IBM PC 300PL, Model 65653BU
  - 1 633 MHz Pentium III CPU
  - 512 MB RAM
  - 19 GB hard disk
  - 1 IBM 10/100 EtherJet PCI Ethernet adapter

9.1.4 Example scenarios

Within this test environment, we describe the tasks necessary to install and configure the following scenarios that are described in Chapter 8, “Installation approach” on page 127:

- Scenario A - basic one-server configuration (server 1), but using the WebSphere embedded Web server in place of the usual stand-alone Web server.
- Scenario B - basic one-server configuration (server 1), but using a stand-alone Web server and the WebSphere HTTP plug-in.
- Scenario D - basic two-server configuration, one server hosting WebSphere and the database server (server 1), the other hosting the Web server and WebSphere HTTP plug-in (server 2). Communication between plug-in and WebSphere is HTTP (unencrypted).
- Scenario E - extension of the basic two-server configuration of Scenario D, with the HTTP communication between the plug-in and WebSphere encrypted using SSL and client authentication mode disabled.
- Scenario F - extension of the secure two-server configuration of Scenario E, with the SSL client authentication mode enabled.
- Scenario G - extension of the secure two-server configuration of Scenario F, with the Web server configured to use SSL for encryption of all communication between itself and Web browsers.
- Scenario H - basic two-server configuration using a single WebSphere administrative domain database.

Installation and configuration tasks

Table 9-2 provides a detailed summary of the tasks and the order in which they must be performed that are necessary to install and configure each of the example scenarios described above.

To install and configure an example scenario, perform the tasks listed for the example in the specified order. Only those tasks that are numbered (non-blank) are to be performed for a particular example scenario. Detailed descriptions of each of these tasks can be found later in this chapter.

Table 9-2 Installation and configuration tasks broken down by scenario

Server	Task	Example scenario						
		A	B	D	E	F	G	H
1	Operating system: Installation of Windows 2000 operating system and any required patches: 1. Install Windows 2000 Operating System 2. Apply any patches as detailed in the WebSphere platform-specific prerequisites See 9.2, “Install Windows 2000” on page 190.	1	1	1	1	1	1	1
1	Web server: Installation and default configuration of IBM HTTP Server: 1. Perform preinstallation tasks 2. Install IBM HTTP Server 3. Configure IBM HTTP Server 4. Verify IBM HTTP Server See 9.3, “Install Web server” on page 190.	-	2	-	-	-	-	2
1	Database server: Installation and configuration of IBM DB2 server necessary to support usage by WebSphere: 1. Perform preinstallation tasks 2. Install DB2 server 3. Verify DB2 server installation 4. Configure DB2 server 5. Set up WebSphere administrative database See 9.4, “Install database server” on page 204.	2	3	2	2	2	2	3

Server	Task	Example scenario						
		A	B	D	E	F	G	H
1	WebSphere Application Server: Installation and configuration of WebSphere necessary to handle requests through the “embedded” Web server: 1. Perform preinstallation tasks 2. Install WebSphere 3. Verify WebSphere installation See 9.6, “Install WebSphere Application Server” on page 221.	3	4	3	3	3	3	4
1	WebSphere Application Server: Creation and configuration of an HTTPS transport on WebSphere default Web container: 1. Create new server certificate keystore 2. Create new self-signed server certificate used for SSL handshaking 3. Create new HTTPS transport for WebSphere Default Server See 9.9, “Configure WebSphere HTTP transport for SSL” on page 245.	-	-	-	4	4	4	-
2	Web server: Installation and default configuration of IBM HTTP Server: 1. Perform preinstallation tasks 2. Install IBM HTTP Server 3. Configure IBM HTTP Server 4. Verify IBM HTTP Server See 9.3, “Install Web server” on page 190.	-	-	4	5	5	5	5

Server	Task	Example scenario						
		A	B	D	E	F	G	H
2	Database client: Installation and configuration of IBM DB2 client necessary to support remote usage by WebSphere: 1. Perform preinstallation tasks 2. Install DB2 client 3. Verify DB2 client installation 4. Configure DB2 client 5. Set up access to remote WebSphere administrative database See 9.5, “Install the database client” on page 216.	-	-	-	-	-	-	6
2	WebSphere Application Server: Installation and configuration of WebSphere necessary to handle requests through the “embedded” Web server: 1. Perform preinstallation tasks 2. Install WebSphere 3. Verify WebSphere installation See 9.6, “Install WebSphere Application Server” on page 221.	-	-	-	-	-	-	7
2	WebSphere HTTP plug-in: Basic installation and configuration of WebSphere HTTP plug-in on remote Web server: 1. Perform preinstallation tasks 2. Install WebSphere plug-in 3. Verify WebSphere plug-in installation 4. Configure for remote WebSphere plug-in 5. Verify remote plug-in configuration See 9.7, “Install the WebSphere plug-in on the remote Web server” on page 236.	-	-	5	6	6	6	-

Server	Task	Example scenario						
		A	B	D	E	F	G	H
2	WebSphere HTTP plug-in: Configuration of plug-in necessary to communicate with HTTPS (client authentication disabled) transport on remote WebSphere: 1. Create new plug-in certificate keystore 2. Import WebSphere self-signed server certicate as a trusted CA 3. Update Web server plug-in to support HTTPS transport and certificate keystore See 9.9, “Configure WebSphere HTTP transport for SSL” on page 245.	-	-	-	7	7	7	-
2	WebSphere HTTP plug-in: Extra configuration of plug-in necessary to support HTTPS transport with client authentication enabled: 1. Create new self-signed client certificate used to authenticate plug-in to server during SSL handshaking See 9.9, “Configure WebSphere HTTP transport for SSL” on page 245.	-	-	-	-	8	8	-
1	WebSphere Application Server: Extra configuration of WebSphere HTTPS transport necessary to support client authentication: 1. Import plug-in self-signed certificate as a trusted CA 2. Configure HTTPS transport to use client authentication See 9.9, “Configure WebSphere HTTP transport for SSL” on page 245.	-	-	-	-	9	9	-
1	WebSphere Application Server: Start up the WebSphere Default Server ready to handle requests: 1. Regenerate Web server plug-in configuration file See 9.9, “Configure WebSphere HTTP transport for SSL” on page 245.	-	-	-	8	10	10	-

Server	Task	Example scenario						
		A	B	D	E	F	G	H
1	Web server: Test that Web server and plug-in can access the current configuration of the WebSphere Default Server: 1. Restart Web server 2. Verify configuration – `http://<server1 hostname>/webapp/examples/showCfg`	-	5	-	-	-	-	8
2	Web server: Test that Web server and plug-in can access the current configuration of the WebSphere Default Server: 1. Restart Web server 2. Verify configuration – `http://<server2 hostname>/webapp/examples/showCfg`	-	-	6	9	11	11	9
1	Web server: Extra configuration of Web server to use SSL encryption for requests: 1. Stop Web server 2. Configure httpd.conf to support SSL for requests 3. Create new keystore 4. Create new self-signed certificate 5. Start Web server See 9.3.5, “Enable SSL encryption for requests (optional)” on page 197.	-	-	-	-	-	12	-
2	Web server: 1. Restart Web server 2. Verify configuration – `https://<host>/` – `https://<host>/webapp/examples/showCfg`	-	-	-	-	-	13	-

9.2 Install Windows 2000

Prior to installing any of the WebSphere components, the proper level of the operating system must be installed.

- Windows 2000 Server + Service Pack 1 - 128-bit encryption.

9.3 Install Web server

This section provides detailed instructions for installing, configuring, and verifying IBM HTTP Server V1.3.19 for Windows 2000.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install IBM HTTP Server.
3. Configure IBM HTTP Server.
4. Verify IBM HTTP Server.
5. Enable SSL encryption for requests (optional).

Note: Whether or not a particular task needs to be performed, and the order in which tasks must be performed, is identified during the planning stage. The tasks used depend on the needs of the particular topology or scenario you are installing. See 9.1, “Planning” on page 182 for details on those tasks to be performed for each example.

9.3.1 Preinstallation tasks

Prior to installing IBM HTTP Server V1.3.19, the following checks and tasks must be completed on the IBM HTTP Server machine:

1. Create groups and users.
2. Check that IP ports are unused.

Create groups and users

To create the required groups and users, perform the following steps:

1. Create a Windows 2000 user with the following settings:
   - Locally defined (not a member of a Windows domain)
   - Member of Administrators group.

You can create local users and assign group memberships by clicking **Control Panel -> Administrative Tools -> Computer Management -> System Tools -> Local Users and Groups**.

2. Assign the following rights to this user:
   - `Act as part of the Operating System`
   - `Log on as a Service`

   You can assign user rights by clicking **Control Panel -> Administrative Tools -> Local Security Policy -> Local Policies -> User Rights Assignment**.

Tip: We suggest creating the user `wasadmin` to run both the IBM HTTP Server and WebSphere. The remainder of this chapter assumes that `wasadmin` is used.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing services on the server that use the following IP ports:
   - 80 (standard HTTP port)
   - 443 (standard HTTPS port)
   - 8008 (IBM HTTP Server Administration port)

   We suggest using the following command for this task:

```
D:\> netstat -an
```

9.3.2 Install IBM HTTP Server

To install the Web server V1.3.19, complete the following steps on the Web server machine:

1. Log on as an administrator user in the local server domain (not part of a Windows domain).
2. Insert the IBM WebSphere Application Server V4.0, Advanced Edition CD into the CD-ROM drive. This CD-ROM also contains the IBM HTTP Server.
3. Using the Windows Explorer, switch to the \httpd directory on the CD. Double-click **Setup** to start the install.
4. In the Choose Setup Language window, select your national language from the drop-down menu and click **OK**.

Note: The remainder of this chapter assumes that U.S. English was selected.

5. Review the content of the Welcome window and click **Next**.
6. Review the content of the Software License Agreement window, and if you accept the conditions, click **Yes**.
7. In the Choose Destination Location window, shown in Figure 9-1 on page 192, select the destination folder, then click **Next**. We selected **D:\IBM HTTP Server**.

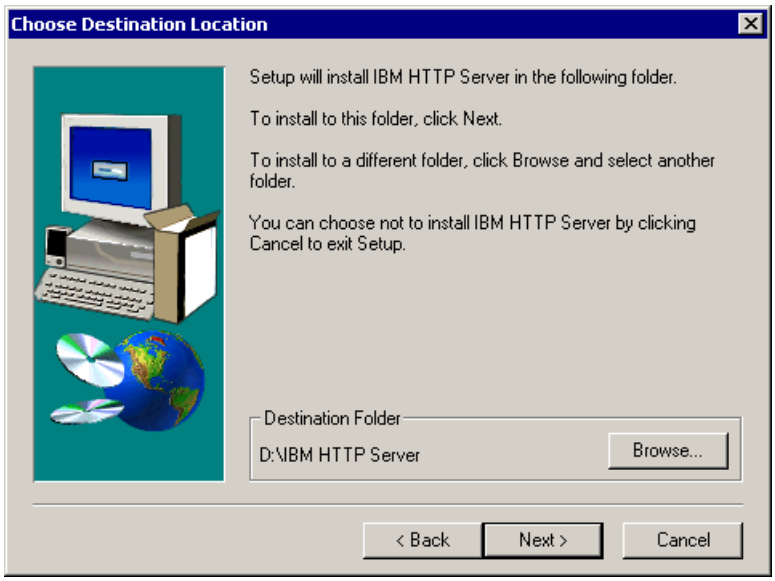


Figure 9-1 Choose the destination directory for Web server software

8. In the Setup Type window, shown in Figure 9-2, select **Typical**, then click **Next**.

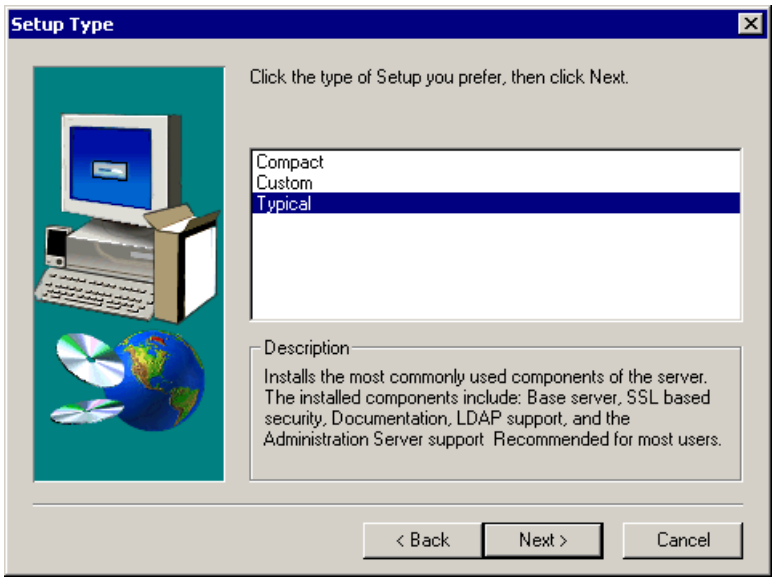


Figure 9-2 Choose Typical installation option

9. In the Select Program Folder window, accept the default folder and click **Next**.
10. In the Information for Service Setup window, shown in Figure 9-3, enter the following, and then click **Install**:
    - User ID: `wasadmin`
    - Password:`<wasadmin_password>`
    - Enter the password again for verification: `<wasadmin_password>`

Note: This is the user ID that the IBM HTTP Server will run as under Windows. In our example, we created a Windows user called `wasadmin` with the necessary privileges during the preinstallation steps.

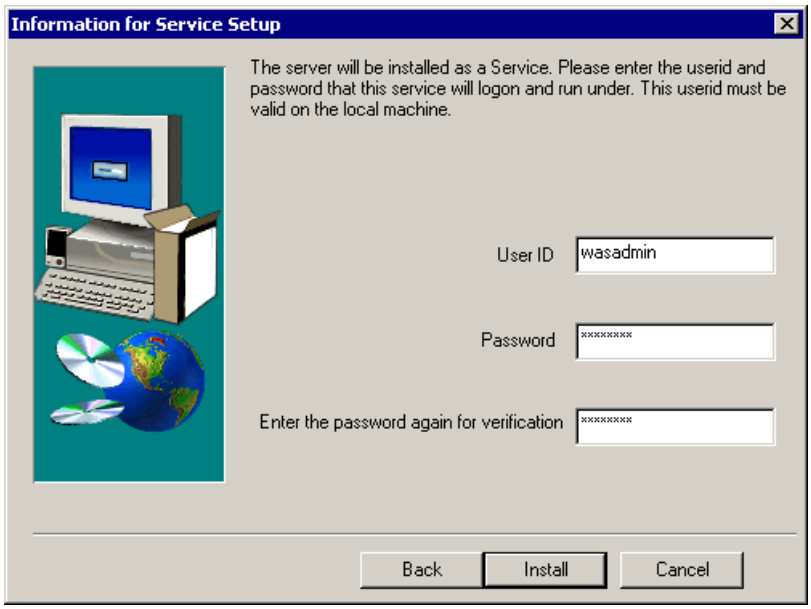


Figure 9-3 Specify username and password for IBM HTTP Server services

11. The IBM HTTP Server installation program copies files from the CD to your machine.
12. When the Setup Complete window appears, select **Yes, I will restart my computer now**, then click **Finish**.
13. When the system restarts, log in to Windows as a user with administrative privileges.

9.3.3 Configure IBM HTTP Server

After installation of IBM HTTP Server V1.3.19, the following configuration tasks must be completed:

1. Create IBM HTTP Server admin account.
2. Update httpd.conf.

Create admin account

To create the IBM HTTP Server administration user, complete the following steps:

1. Open a command window, by clicking **Start** -> **Programs** -> **Command Prompt**.
2. Change directory to <http_server_install_path>.

3. Create the administration user by typing the following commands:

```
D:\IBM HTTP Server> htpasswd -m conf\admin.passwd admin
New password: <admin_password>
Re-type new password: <admin_password>
```

   Where `admin` is the IBM HTTP Server administration user ID.

Note: Any user ID can be used as the IBM HTTP Server administration user ID.

4. Close the command prompt window.

Update httpd.conf

The IBM HTTP Server configuration file httpd.conf must be updated to reflect the following:

1. Edit the <http_server_install_path>\conf\httpd.conf configuration file using a text editor.
2. Set the value of the ServerName variable to the fully qualified DNS name of the server, that is <hostname.domain.com>.
3. Save the changes and exit.
4. Reboot the system.

9.3.4 Verify IBM HTTP Server

In order to verify the IBM HTTP Server V1.3.19 installation, perform the following checks:

1. Check that services are running.
2. Check request handling.

Check that services are running

To check that the IBM HTTP Server services are running, perform the following steps:

1. Check that the Windows services listed in Table 9-3 have been added and are running.

Table 9-3 IBM HTTP Server Windows services

Service name	Status	Startup mode	Log on as...
IBM HTTP Administration	Started	Automatic	wasadmin

Service name	Status	Startup mode	Log on as...
IBM HTTP Server	Started	Automatic	wasadmin

2. If not running, issue the following commands:

```
D:\> net start "IBM HTTP Server"
D:\> net start "IBM HTTP Administration"
```

Check request handling

To check IBM HTTP Server request handling, perform the following steps:

1. Using a Web browser, request the following URL representing the IBM HTTP Server home page:

   ```
   http://<hostname.domain.com>/
   ```

   The window shown in Figure 9-4 will be displayed if the IBM HTTP Server has been installed and configured correctly.

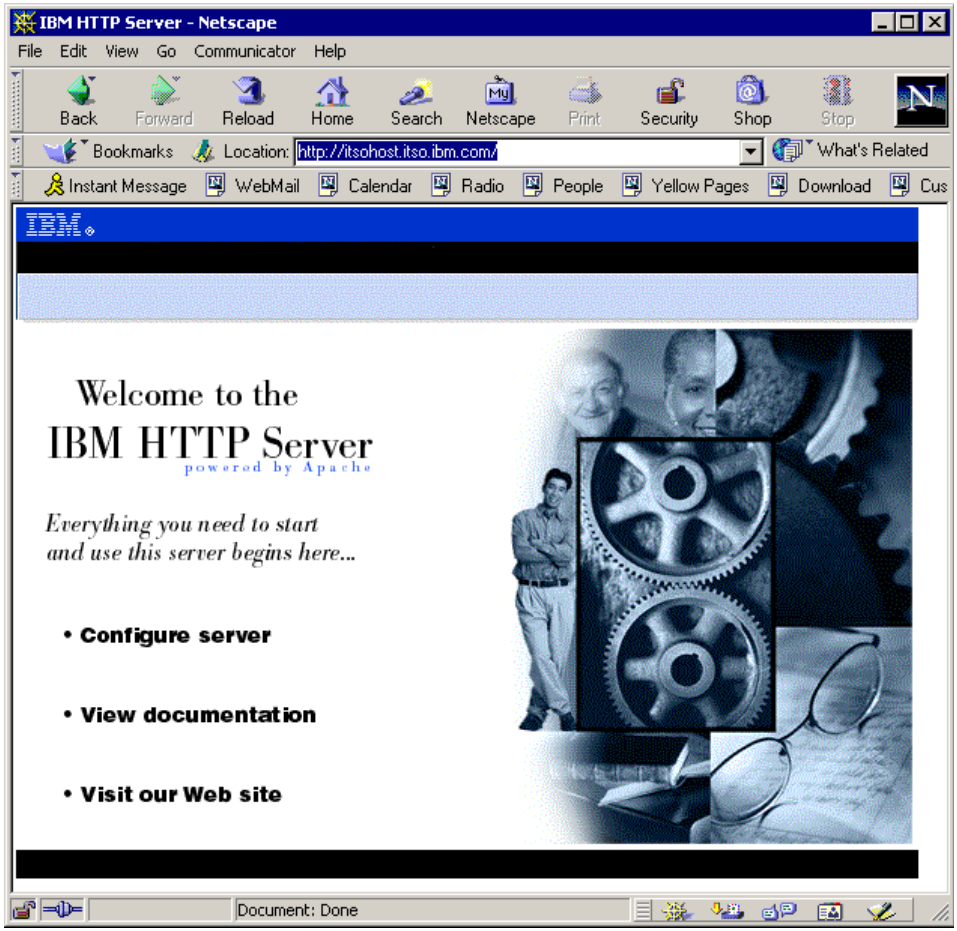


Figure 9-4 Home page request handled by Web server

9.3.5 Enable SSL encryption for requests (optional)

In this section we provide detailed instructions for creating a certificate, installing the certificate and configuring an IBM HTTP Server for SSL. In our example, we will create a self-signed test certificate. For a production environment you will need to request a real certificate from a certificate authority, such as VeriSign.

Enabling SSL for communication between the IBM HTTP Server and a Web browser is a multistep process:

1. Stop IBM HTTP Server process.
2. Configure httpd.conf to add SSL support.

3. Create new keystore (certificate trust) database.
4. Create new self-signed certificate.
5. Start IBM HTTP Server process.
6. Verify SSL configuration.

Stop IBM HTTP Server process

To stop the IBM HTTP Server process, issue the following command:

```
D:\> net stop "IBM HTTP Server"
```

Configure httpd.conf to add SSL support

To configure SSL for the IBM HTTP Server, complete the following steps:

1. Use the sample httpd.conf.sample file, which includes SSL entries, as a starting point to enable SSL for the IBM HTTP Server.
   a. Change to the <http_server_install_path>\conf directory.
   b. Back up the existing httpd.conf file by renaming it to httpd.conf.bak.
   c. Rename httpd.conf.sample to httpd.conf.
2. Modify the httpd.conf file using a text editor. Ensure that the following lines are uncommented by removing the # symbol:

   ```
   LoadModule ibm_ssl_module modules/IBMModuleSSL<encrypt_level>.dll
   ```

   (Where <encrypt_level> is the appropriate encryption level for your locale. For example, IBMModuleSSL128.dll for 128-bit encryption in the US and Canada.)

   ```
   Listen 443
   <VirtualHost hostname.domain.com:443>
   ```

   (You must substitute your fully qualified host name in this line, for example <VirtualHost itsohost.itso.ibm.com:443>.)

   ```
   SSLEnable
   </VirtualHost>
   SSLDisable
   Keyfile "<http_server_install_path>/ssl/webclient.kdb"
   SSLV2Timeout 100
   SSLV3Timeout 1000
   ```

   The value of the KeyFile parameter is the absolute path to the CMS format keystore database file of your choosing. In this example we assume that a webclient.kdb file is created in the ssl subdirectory of the IBM HTTP Server installation.

3. Ensure the following settings have been disabled by adding the # symbol to the start of each line:

```
#AfpaEnable
#AfpaCache on
#AfpaLogFile <log_file_path>
```

> **Note:** The above AFPA options must be disabled in order for SSL encryption mode to operate correctly.

4. Save the changes.

Create a new keystore (certificate trust) database

To create a new SSL keystore database file, complete the following steps:

1. Start the Web server IBM Key Management Utility by clicking **Start** -> **Programs** -> **IBM HTTP Server** -> **Start Key Management Utility**.
2. Select **Key Database File** from the menu bar, then select **New**.
3. In the New window, enter the following and then click **OK**:
   - Key Database Type: `CMS key database file`
   - File Name: `webclient.kdb` (must be the same as in httpd.conf)
   - Location: `<http_server_install_path>\ssl\`
4. In the Password Prompt window, shown in Figure 9-5 on page 200, enter the following, then click **OK** to continue:
   - Password: password to protect keystore file contents
   - Check **Set expiration time?** and enter number of days if the password should expire. If no expiration is required, uncheck this setting.

> **Note:** Although not required in a development environment, it is strongly recommended that all keystores used in a production environment set an expiration period.

   - Check **Stash the password to a file?**

> **Note:** The IBM HTTP Server accesses the password protected keystore file <filename>.kdb using the password contained in the <filename>.sth stashfile. Consequently, the stash option must be enabled.

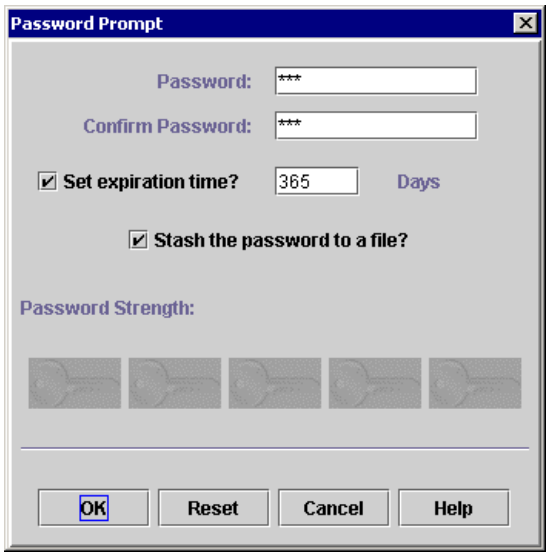


Figure 9-5 Specify password and expiration date of keystore file

5. Click **OK** when the Information window appears with the message:

```
The password has been encrypted and saved in the file:
<http_server_install_path>\ssl\webclient.sth
```

Create new self-signed certificate

To create a new self-signed certificate, perform the following steps:

1. Start the Web server IBM Key Management Utility by clicking **Start** -> **Programs** -> **IBM HTTP Server** -> **Start Key Management Utility**.
2. Select **Key Database File** from the main menu, then select **Open**. Specify the keystore database file. Our example uses <http_server_install_path>\ssl\webclient.kdb.
3. Select **Create** from the menu bar, then select **New Self-Signed Certificate**.

Note: If you are enabling SSL for a production environment, select **New Certificate Request** instead. It is strongly recommended that self-signed digital certificates not be used in production.

4. In the Create New Self-Signed Certificate window, shown in Figure 9-6 on page 201, enter the following values, then click **OK**.
   - Key Label: `<user defined label>`
   - Version: `X509 V3`

- Key Size: `1024`
- Common Name: `<hostname.domain.com>`
- Organization: `IBM`
- Organization Unit: `ITSO`

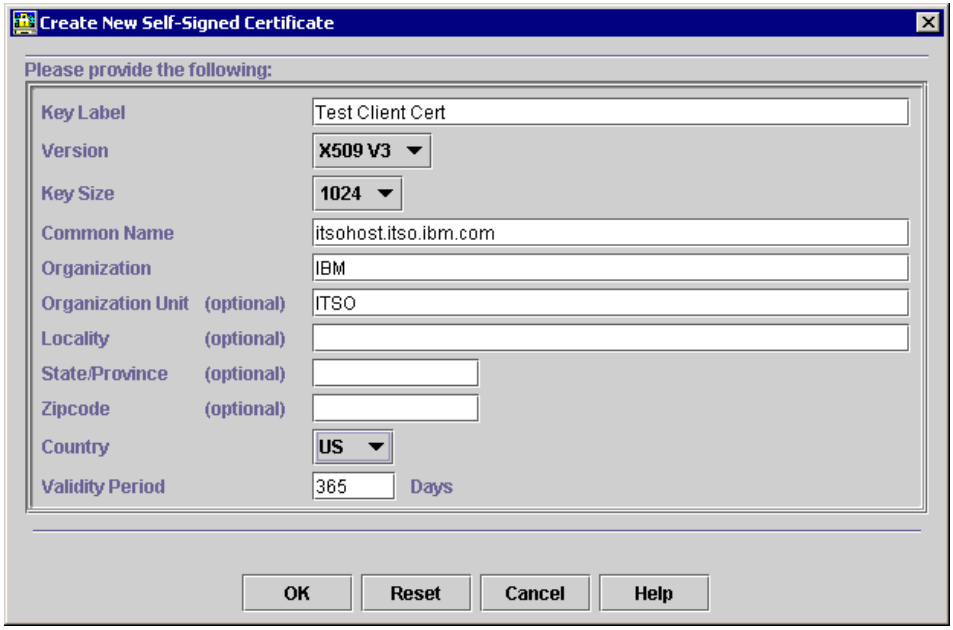


Figure 9-6 Specify settings for new self-signed certificate

5. The new certificate should be listed in the Personal Certificates pane, as shown in Figure 9-7 on page 202.

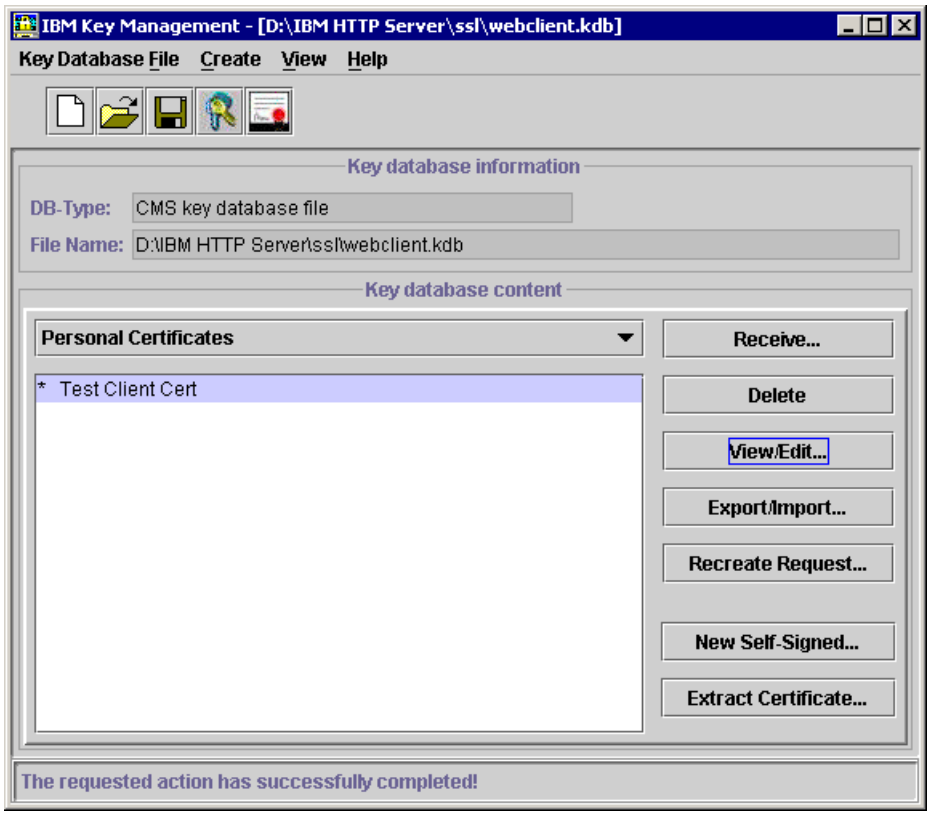


Figure 9-7 New self-signed certificate listed in keystore contents

6. Close the Web server IBM Key Management Utility.

Start IBM HTTP Server process

To start the IBM HTTP Server process, issue the following command:

```
D:\> net start "IBM HTTP Server"
```

Verify SSL configuration

To verify the IBM HTTP Server SSL configuration, perform the following steps:

1. Using a Web browser request the following URL, representing the IBM HTTP Server home page:

   ```
   https://<hostname.domain.com>/
   ```

2. Since the test certificate we are using is self-signed, the browser should display a window similar to the one shown in Figure 9-8 on page 203. Click **Next**.

Figure 9-8 Browser prompt when certificate with unknown CA is received

3. The next window should display **Certificate for** and **Signed by** values for the certificate that match those used when creating the certificate. Click **Next**.
4. In the next window, select **Accept this certificate for this session**, then click **Next**.
5. In the next window, click **Next**.
6. In the next window, click **Finish** to accept the certificate.
7. If the IBM HTTP Server has been correctly configured for SSL, then the window shown in Figure 9-9 on page 204 will be displayed in the Web browser.

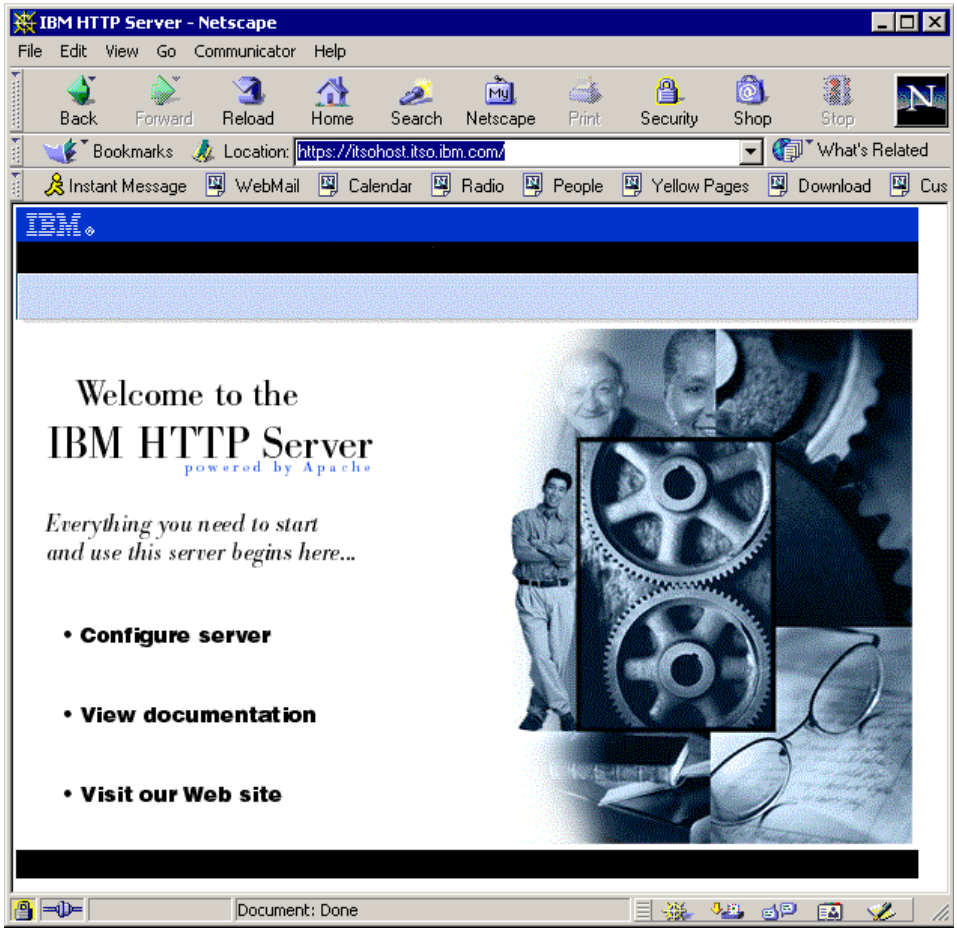


Figure 9-9 Secured request for home page handled by Web server

9.4 Install database server

This section provides detailed instructions for installing, configuring, and verifying IBM DB2 Universal Database, Enterprise Edition for Windows 2000.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install DB2 server.
3. Verify DB2 server installation.
4. Configure DB2 server.

5. Set up WebSphere administrative database.

9.4.1 Preinstallation tasks

Prior to installing IBM DB2 Universal Database, Enterprise Edition for Windows, the following tasks must be completed on the database server machine:

1. Create groups and users.
2. Check that IP ports are unused.

Create groups and users

To create the required groups and users, perform the following steps:

1. Create a Windows 2000 user with the following settings:
   - Locally defined (not a member of a Windows domain)
   - Member of Administrators group.

   You can create local users and assign group memberships by clicking **Control Panel -> Administrative Tools -> Computer Management -> System Tools -> Local Users and Groups**.

Important: You must use a password that is compatible with DB2. DB2 supports passwords of 8 or fewer characters, and which do not contain the characters “<“ or “>”.

2. Assign the following rights to this user:
   - `Act as part of the Operating System`
   - `Log on as a Service`
   - `Create a token object`
   - `Increase quotas`
   - `Replace a process level token`

   You can assign user rights by clicking **Control Panel -> Administrative Tools -> Local Security Policy -> Local Policies -> User Rights Assignment**.

Tip: We suggest creating an account called db2admin. The remainder of this chapter assumes that this is the case.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing active services that use the following IP ports on the server:
   - 523 (DB2 Administration Server)
   - 50000 (DB2 instance connection port)
   - 50001 (DB2 instance interrupt port)
   - 50002 (DB2 Control Server)

   We suggest using the following command for this task:

```
D:\> netstat -an
```

9.4.2 Install the DB2 server

In order to install IBM DB2 Universal Database, Enterprise Edition for Windows, perform the following steps on the database server machine:

1. Log on as the db2admin user.
2. Insert the IBM DB2 Universal Database, Enterprise Edition for Windows CD.
3. If the install program does not start automatically, run the **`setup`** program under the root directory of the CD.
4. In the Select Products window, shown in Figure 9-10 on page 206, select only **DB2 Enterprise Edition**, then click **Next**.

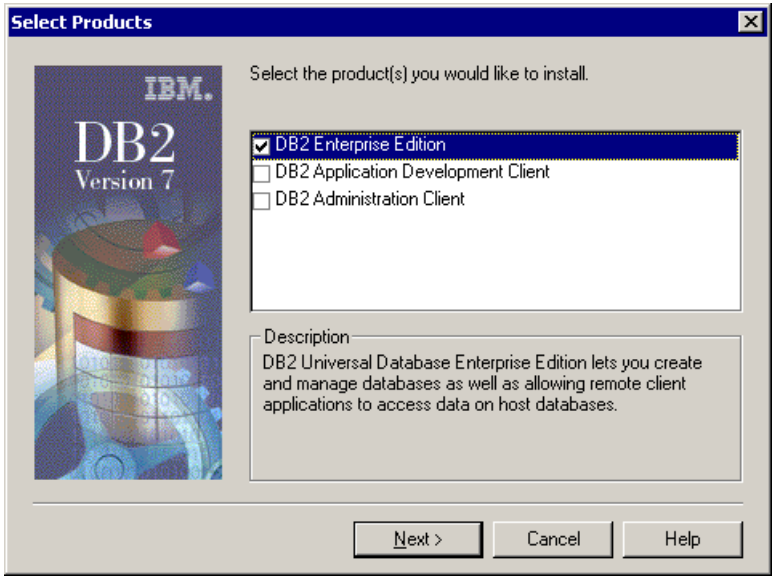


Figure 9-10 Select DB2 products to install

5. In the Select Installation Type window, shown in Figure 9-11, select the **Typical** option, then click **Next**.

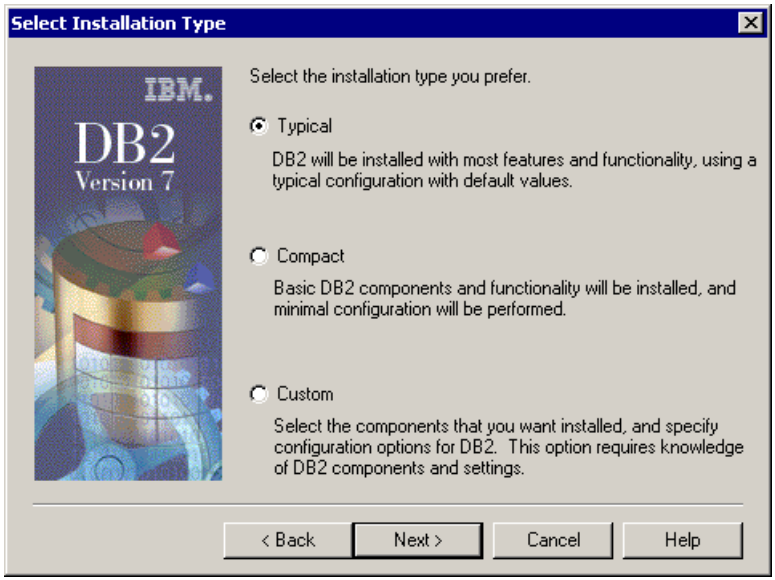


Figure 9-11 Select the DB2 installation type

6. In the Choose Destination Location window, shown in Figure 9-12, select the destination folder, then click **Next**. We selected **D:\SQLLIB**.

Figure 9-12 Choose the destination directory for DB2 components

7. In the Enter Username and Password for Control Center Server window, shown in Figure 9-13 on page 209, enter the following, then click **Next**:
   - Username: `db2admin`
   - Password: `<db2admin password>`
   - Confirm password: `<db2admin password>`
   - Check the **Use the same values for the remaining DB2 Username and Password settings** option.

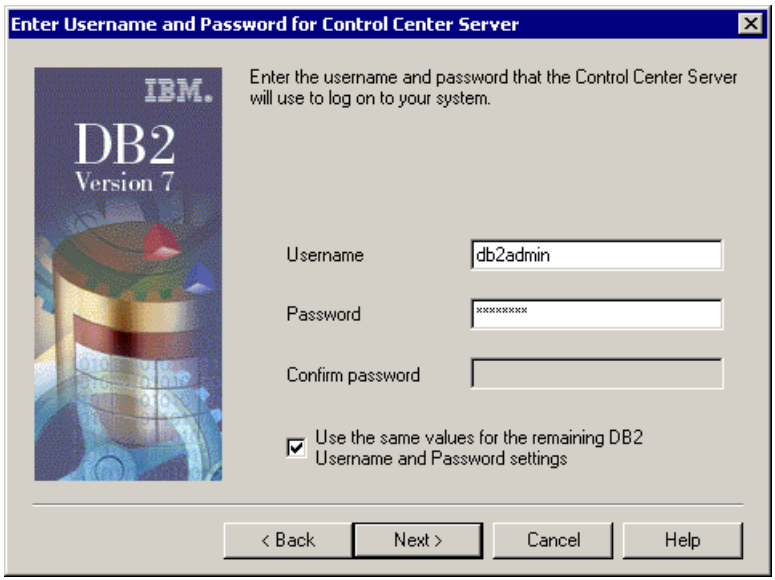


Figure 9-13 Specify username and password for DB2 services

8. In the Start Copying Files window, check that the correct components have been selected. If yes, click **Next** to start the installation process.

   The installation performs the default configuration for users, groups and an instance to be used by DB2.

9. In the Install OLAP Starter Kit window, shown in Figure 9-14, select **Do not install OLAP Starter Kit**, then click **Continue**.

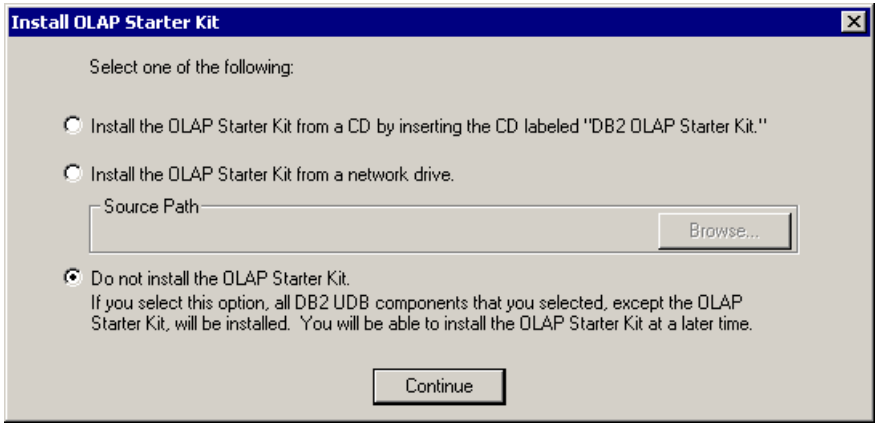


Figure 9-14 Select not to install the DB2 OLAP Starter Kit

10. In the Setup Complete window, click **Finish**.
11. When the First Steps window appears, click **Exit**.
12. Restart the system.
13. When the system restarts, log in locally under the db2admin account.

9.4.3 Verify the DB2 server installation

To verify the DB2 server installation, complete the following tasks:

1. Check the DB2 release level.
2. Check the DB2 service name.
3. Check the database manager configuration.

Check the DB2 release level

Check that DB2 has the correct internal release level to meet WebSphere requirements:

1. Start the DB2 command window by clicking **Start** -> **Programs** -> **IBM DB2** -> **Command Window**.
2. Enter the following command:

   ```
   D:\> db2level
   ```

   It should generate output similar to the following:

   ```
   DB21085I  Instance "DB2" uses DB2 code release "SQL07021" with level
   identifier "03020105" and informational tokens "DB2 v7.1.0.43", "n010504"
   and "WR21254a"
   ```

3. An internal release level of 7.1.0.43 should be indicated.
4. Close the DB2 command window.

Check the DB2 service name

Check the service name recorded in the TCP/IP `services` file:

1. Open the `services` file located in the <windir>\system32\drivers\etc directory, and find the entries that have comments referring to the DB2 instance connection port.
2. Locate the service name in the first column that corresponds to the lower port number. For example, if the following services were displayed:

   ```
   db2cdb2 50000/tcp #Connection port for DB2 instance db2
   db2idb2 50001/tcp #Interrupt port for DB2 instance db2
   ```

3. Record the `db2cdb2` service name for later use.

Check the database manager configuration

Check the service name is recorded in the database manager configuration:

1. Start the DB2 command window by clicking **Start** -> **Programs** -> **IBM DB2** -> **Command Window**.
2. Enter the following command:

   ```
   D:\> db2 get dbm cfg | more
   ```

3. Look at the output for the SVCENAME entry. Verify that a value exists, and that it matches the value for the service name recorded above from the services file. For example, something similar to the following should be displayed:

   ```
   TCP/IP Service name (SVCENAME)=db2cdb2
   ```

4. If the value does not match, update the database manager configuration using the following command in the DB2 command window:

   ```
   D:\> db2 update dbm cfg using svcename <service_name>
   D:\> db2stop
   D:\> db2start
   ```

   Where `<service_name>` must be replaced with the service name.

5. Close the DB2 command window.

9.4.4 Configure the DB2 server

After the DB2 installation, a number of configuration tasks must be performed on the database server machine, so that WebSphere is able to use DB2 as the repository for its administrative database:

1. Stop unused Windows services.
2. Update JDBC level.
3. Configure TCPIP communication mode.

Stop unused services

To conserve system memory, you can choose to start only those DB2-related Windows services that are used by most WebSphere applications. The startup type for the service can be set to automatic, manual, or disabled.

1. Table 9-4 lists the services settings as they are after installation, as well as our recommended settings.

Table 9-4 IBM DB2 Windows services

Service name	Startup mode after installation	Startup mode recommended setting
DB2-DB2	Automatic	Automatic
DB2-DB2CTLSV	Automatic	Automatic
DB2-DB2DAS00	Automatic	Automatic
DB2 Governor	Manual	Manual
DB2 JDBC Applet Server	Automatic	Automatic
DB2 Applet Server Control Center	Manual	Manual
DB2 License Server	Automatic	Automatic
DB2 Security Server	Automatic	Manual
Warehouse Logger	Automatic	Manual
Warehouse Server	Automatic	Manual

2. To change the startup of one of these DB2 Windows services, use the Windows 2000 Services control panel.

Update the JDBC level

IBM WebSphere Application Server V4.0 requires JDBC2.0, whereas the default installation of IBM DB2 uses JDBC1.2. To update the DB2 JDBC level, complete the following steps:

1. Stop the **DB2 JDBC Applet Server** Windows service as follows:

```
D:\> net stop "DB2 JDBC Applet Server"
```

2. In a command window, change to the <db2_install_path>\java12 directory where you installed DB2 and the type the following command:

```
<db2_install_path>\java12\> usejdbc2.bat
```

3. Output similar to that in Figure 9-15 should be obtained.

```
D:\SQLLIB\java12>dir
 Volume in drive D is Apps
 Volume Serial Number is 1234-5678

 Directory of D:\SQLLIB\java12

20/10/2001  03:15p      <DIR>          .
20/10/2001  03:15p      <DIR>          ..
06/05/2001  10:27a             676,339 jdbc20.exe
05/05/2001  12:53a               1,441 usejdbc1.bat
05/05/2001  12:53a               3,217 usejdbc2.bat
               3 File(s)        680,997 bytes
               2 Dir(s)   1,305,538,560 bytes free

D:\SQLLIB\java12>usejdbc2
UnZipSFX 5.31 of 31 May 1997, by Info-ZIP (Zip-Bugs@lists.wku.edu).
  inflating: db2java.zip
  inflating: db2jdbc.dll
  inflating: db2ccs.exe
  inflating: db2jd.exe
  inflating: db2jds.exe
        1 file(s) copied.
        1 file(s) copied.
        1 file(s) copied.
        1 file(s) copied.
        1 file(s) copied.
        1 file(s) copied.

D:\SQLLIB\java12>
```

Figure 9-15 Expected output from usejdbc2.bat

Note: If the output of usejdbc2 indicates that any of the files failed to copy successfully, then the JDBC2 update failed. If this occurs, stop all DB2 services and then repeat the above steps. If you see any “access denied” or “process cannot access...” errors and the JDBC Applet Server is indeed not running, then some other (non-DB2) process has locked the db2java.zip file for some reason.

4. Start the DB2 JDBC Applet Server Windows service as follows:

   ```
   D:\> net start "DB2 JDBC Applet Server"
   ```

5. Check the contents of the <db2_install_path>\java12\inuse file. If JDBC 2.0 is being used, the file will contain:

   ```
   JDBC 2.0
   ```

Configure the TCPIP communication mode

The DB2 server may need to be reconfigured to use TCP/IP as its primary communication method:

1. Open a DB2 command window by clicking **Start** -> **Programs** -> **IBM DB2** -> **Command Window**.
2. Check whether TCPIP is the current DB2 communication method. The following command should return a value of TCPIP:

   ```
   D:\> db2set DB2COMM
   ```

3. If not TCPIP, reset the DB2COMM DB2 environment variable:

   ```
   D:\> db2set DB2COMM=TCPIP
   ```

4. Close the DB2 command window.

9.4.5 Set up the WebSphere administrative database

In order to set up a database in DB2 to use as the WebSphere administrative repository, the following tasks need to be performed on the database server:

1. Create the WebSphere database.
2. Create the WebSphere access account.

Create the WebSphere database

The following steps create the WebSphere Application Server repository database, also known as the WebSphere database. The database will be populated with WebSphere schema and default values in a later task.

To set up the WebSphere database, complete the following steps:

1. Open a DB2 command window by clicking **Start** -> **Programs** -> **IBM DB2** -> **Command Window**.
2. Create the WebSphere database and configure its heap size to suit WebSphere requirements:

   ```
   D:\> db2 create db was1
   D:\> db2 update db cfg for was1 using applheapsz 256
   ```

   Note: Although the repository database created here is called was1, any valid DB2 database name can be used.

3. Check that the new database is known to DB2:

   ```
   D:\> db2 list db directory
   ```

   This command should give output containing the following:

```
Database 1 entry:
 Database alias                 = WAS1
 Database name                  = WAS1
 Database drive                 = D:\DB2
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Indirect
 Catalog node number            = 0
```

4. In order to access the administrative database via TCP/IP, the DB2 node must first be cataloged:

```
D:\> db2 catalog tcpip node <node_name> remote <local_hostname> server
<service_name>
```

Important: The <service_name> used to catalog the node must be the same as the database instance connection port name in the Windows services file. The <node_name> chosen can be any valid DB2 node name.

5. The administrative database must now be cataloged as part of this TCP/IP node:

```
D:\> db2 catalog db was1 as was at node <node_name>
```

6. Check that the database TCPIP alias is known to DB2:

```
D:\> db2 list db directory
```

This should give output containing the following:

```
Database 2 entry:
 Database alias                 = WAS
 Database name                  = WAS1
 Node name                      = <node_name>
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Remote
 Catalog node number            = -1
```

7. Verify connection to the local database via TCP/IP:

```
D:\> db2 connect to was user <db2admin_user> using <db2admin_passwd>
D:\> db2 disconnect current
```

where <db2admin_user> and <db2admin_passwd> are the DB2 administration account and password respectively on the local DB2 server.

8. Close the DB2 command window.

Create the WebSphere access account

A new user must be set up in the WebSphere administrative database for use by WebSphere for all database access:

1. Create a new local Windows user account, <was_user>.

Tip: We suggest that a user called “was” be used as the WebSphere DB2 access account. The remainder of this chapter assumes this is the case.

2. Open a DB2 command window by clicking **Start** -> **Programs** -> **IBM DB2** -> **Command Window**.
3. Connect to the WebSphere database as the DB2 administrator:

```
D:\> db2 connect to was user <db2admin_user> using <db2admin_passwd>
D:\> db2 grant connect,createtab on database to user <was_user>
D:\> db2 disconnect was
```

4. Test the connection to the WebSphere database using the <was_user> account:

```
D:\> db2 connect to was user <was_user> using <was_password>
```

   This should give output containing the following:

```
Database Connection Information
 Database server        = DB2/NT <db2_version>
 SQL authorization ID   = <was_user>
 Local database alias   = WAS
```

Note: This test mimics the method used by the WebSphere administrative server to access the WAS database.

5. Disconnect from the current database:

```
D:\> db2 disconnect current
```

6. Close the DB2 command window.

9.5 Install the database client

This section provides detailed instructions for installing, configuring, and verifying IBM DB2 Client for Windows 2000.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install DB2 client.
3. Verify DB2 client installation.
4. Configure DB2 client.
5. Set up access to remote WebSphere administrative database.

9.5.1 Preinstallation tasks

Prior to installing the IBM DB2 client, the following tasks must be completed:

1. Create groups and users.
2. Check IP ports are unused.

Create groups and users

To create the required groups and users, perform the following steps:

1. Create a Windows 2000 user with the following settings:
   - Locally defined (not a member of a Windows domain)
   - Member of Administrators group.

   You can create local users and assign group memberships by clicking **Control Panel -> Administrative Tools -> Computer Management -> System Tools -> Local Users and Groups**.

Important: You must use a password that is compatible with DB2. DB2 supports passwords of 8 or fewer characters, and which do not contain the characters “<“ or “>”.

2. Assign the following rights to this user:
   - `Act as part of the Operating System`
   - `Log on as a Service`
   - `Create a token object`
   - `Increase quotas`
   - `Replace a process level token`

   You can assign user rights by clicking **Control Panel -> Administrative Tools -> Local Security Policy -> Local Policies -> User Rights Assignment**.

Tip: We suggest creating an account called “db2admin”.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing active services that use the following IP port on the server: 50000 (DB2 instance connection port).

   We suggest using the following command for this task:

   ```
   D:\> netstat -an
   ```

9.5.2 Install the DB2 client

In order to install the DB2 client for Windows, perform the following steps:

1. Log on as an administrator user in the local server domain (not part of a Windows domain).
2. Insert the IBM DB2 Universal Database, Enterprise Edition for Windows CD-ROM.
3. If the install program does not start automatically, run the **`setup`** program under the root directory of the CD-ROM.
4. In the Installation window, click **Install**.
5. In the Select Products window, shown in Figure 9-16, select the **DB2 Administration Client** only, then click **Next**.

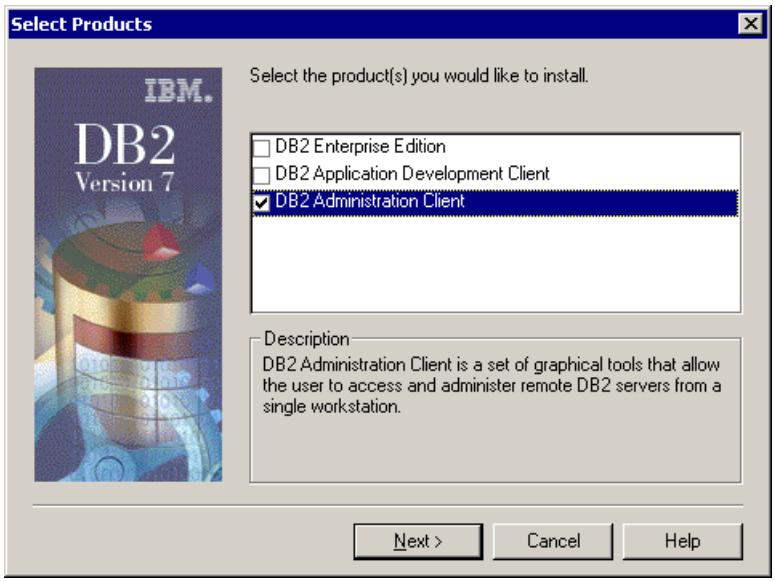


Figure 9-16 Select DB2 products to install

6. In the Select Installation Type window, select the **Typical** option, then click **Next**.
7. In the Choose Destination Location window, select the destination folder, then click **Next**. We selected **D:\SQLLIB**.
8. In the Configure NetBIOS window, shown in Figure 9-17 on page 219, ensure the **Configure NetBIOS for connection to DB2 server** setting is unselected, then click **Next**.

Figure 9-17 Configure NetBIOS settings for DB2 client

9. In the Enter Username and Password for Control Center Server window, enter the following and then click **Next**.
   - Username: `db2admin`
   - Password: `<db2admin password>`
   - Confirm password: `<db2admin password>`
   - Check the **Use the same values for the remaining DB2 Username and Password settings** option
10. In the Start Copying Files window, check that the correct components have been selected. If yes, click **Next** to start the installation process.
11. In the Setup Complete window, click **Finish**.
12. When the First Steps window appears, click **Exit**.

9.5.3 Verify the DB2 client installation

To verify the DB2 client installation, complete the following tasks:

1. Check the DB2 release level, as described in “Check the DB2 release level” on page 210.

9.5.4 Configure the DB2 client

After the DB2 client installation, a number of configuration tasks must be performed so that WebSphere is able to use it to access a remote DB2 database:

1. Stop unused Windows services.
2. Update the JDBC level.

Stop unused services

To conserve system memory, you can choose to start only those DB2-related Windows services that are used by most WebSphere applications. The startup type for the service can be set to automatic, manual, or disabled.

1. The table below lists the services settings as they are after installation, as well as our recommended settings.

Table 9-5 IBM DB2 Windows services

Service name	Startup mode after installation	Startup mode recommended setting
DB2 JDBC Applet Server	Automatic	Automatic
DB2 Applet Server Control Center	Manual	Manual
DB2 Security Server	Automatic	Manual

2. To change the startup of one of these DB2 Windows services, use the Windows 2000 Services control panel.

Update the JDBC level

IBM WebSphere Application Server V4.0 requires JDBC2.0, whereas the default installation of IBM DB2 uses JDBC1.2. To update the DB2 JDBC level, follow the steps described in “Update the JDBC level” on page 212.

9.5.5 Set up access to the remote administrative database

The following steps set up TCP/IP access to a remote DB2 database. The database will be installed as part of the DB2 server installation on the other server. Complete the following steps:

1. Edit the <windir>\system32\drivers\etc\services file to add an entry for the DB2 connection port of the remote server:

```
db2cdb2 50000/tcp # Connection port of remote DB2 instance db2
```

2. Open a DB2 command window by clicking **Start** -> **Programs** -> **IBM DB2** -> **Command Window**.

3. In order to access the administrative database via TCP/IP, the remote DB2 node must first be cataloged:

```
D:\> db2 catalog tcpip node <node_name> remote <remote_hostname> server
<service_name>
```

Important: The <service_name> specified in the catalog command must be the same as the entry added to the Windows services file. The <node_name> chosen can be any valid DB2 node name.

4. The administrative database must now be cataloged as part of this remote TCP/IP node:

```
D:\> db2 catalog db was1 as was at node <node_name>
```

5. Verify connection to the remote database via TCP/IP:

```
D:\> db2 connect to was user <db2admin_user> using <db2admin_passwd>
```

where <db2admin_user> and <db2admin_passwd> are the the DB2 administration account and password respectively on the remote DB2 server.

6. Test connection to the remote database using the WebSphere access account. This access method mimics how WebSphere will access the administrative database.

```
D:\> db2 connect to was user <was_user> using <was_password>
D:\> db2 disconnect was
```

Important: See “Create the WebSphere access account” on page 215 for details on the <was_user> access account.

7. Close the DB2 command window.

9.6 Install WebSphere Application Server

This section provides detailed instructions for installing, configuring, and verifying WebSphere Application Server V4.0, Advanced Edition for Windows 2000.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install WebSphere.
3. Verify WebSphere installation.

9.6.1 Preinstallation tasks

Prior to installing IBM WebSphere Application Server V4.0, the following checks and tasks need to be completed on the WebSphere server machine:

1. Create groups and users.
2. Check that IP ports are unused.
3. Stop the Web server processes.

Create groups and users

To create the required groups and users, perform the following steps:

1. If you have not already done so, create a Windows 2000 user under which the WebSphere service will be run, as follows:
   - Locally defined (not a member of a Windows domain)
   - Member of Administrators group.

   You can create local users and assign group memberships by clicking **Control Panel -> Administrative Tools -> Computer Management -> System Tools -> Local Users and Groups**.
2. Assign the following rights to this user:
   - `Act as part of the Operating System`
   - `Log on as a Service`

   You can assign user rights by clicking **Control Panel -> Administrative Tools -> Local Security Policy -> Local Policies -> User Rights Assignment**.

Tip: We suggest creating the user “wasadmin”.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing active services that use the following IP ports on the server:
   - 900 (bootstrap port)
   - 9000 (Location Service Daemon)
   - 9080 (default application server)

   We suggest using the following command for this task:

```
D:\> netstat -an
```

Stop the Web server processes

The IBM HTTP Server process must be stopped while WebSphere is installed. The WebSphere installation changes the httpd.conf configuration file as part of the Web server plug-in component installation.

1. Issue the command:

```
D:\> net stop "IBM HTTP Server"
```

9.6.2 Install WebSphere

To install IBM WebSphere Application Server V4.0, Advanced Edition using the GUI installer interface, complete the following steps on the WebSphere server machine:

Tip: The WebSphere installer (setup.exe) also provides a non-GUI, scripted or "silent" mode of operation. See 9.10, "Install WebSphere Application Server - silent mode" on page 257 for details.

1. Log on as an administrator user in the local server domain (not part of a Windows domain).
2. Insert the IBM WebSphere Application Server V4.0, Advanced Edition CD.
3. Start the WebSphere Application Server installation by double-clicking **Setup** from the root of the CD.
4. In the Choose Setup Language window, select your national language from the drop-down menu (English is selected by default) and click **OK**.
5. In the WebSphere Application Server Attention window, read the warnings and then click **Next** to continue.
6. In the Installation Options window, shown in Figure 9-18 on page 224, select **Custom Installation**, and then click **Next**.

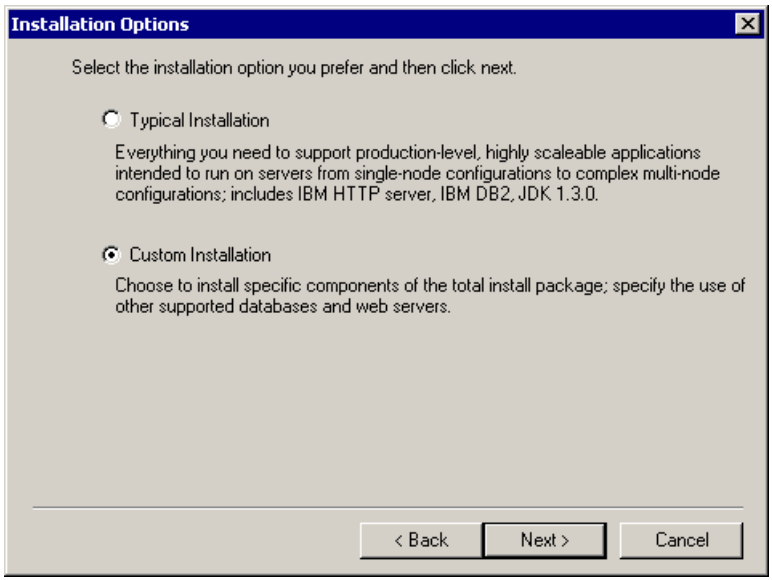


Figure 9-18 Choose Installation options

7. In the Choose Application Server Components window, shown in Figure 9-19 on page 225, select all components except the IBM HTTP Server, then click **Next**.

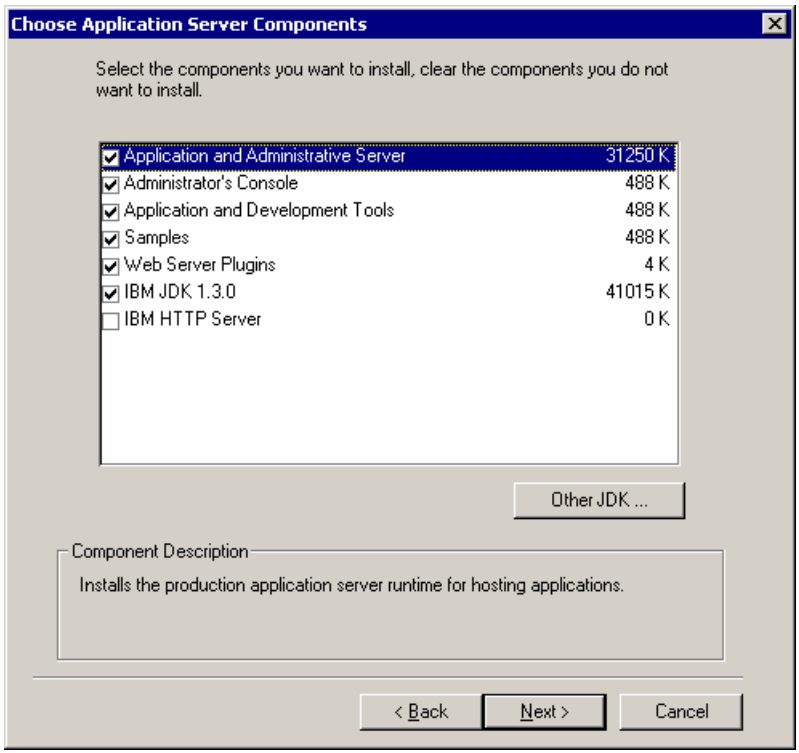


Figure 9-19 Choose Application Server components to install

Note: The Samples are optional, but are useful during verification of the WebSphere installation.

8. In the Choose Web server Plugins window, shown in Figure 9-20 on page 226, select only the **IBM HTTP Server** option and then click **Next**.

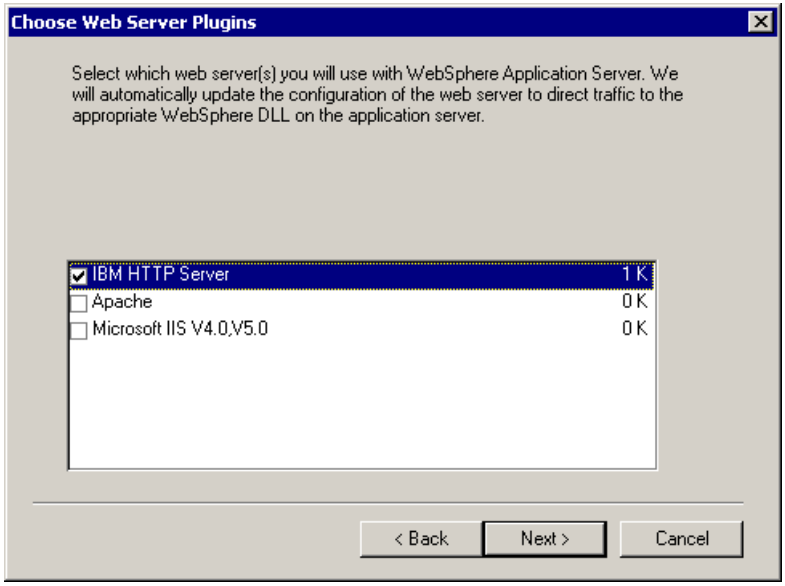


Figure 9-20 Choose Web Server Plugins to install

9. In the Security Options window, shown in Figure 9-21 on page 227, enter the username and password of the Windows account under which the WebSphere Application Server is to run as a service. This is the account created during the WebSphere preinstallation tasks. Click **Next** to continue.

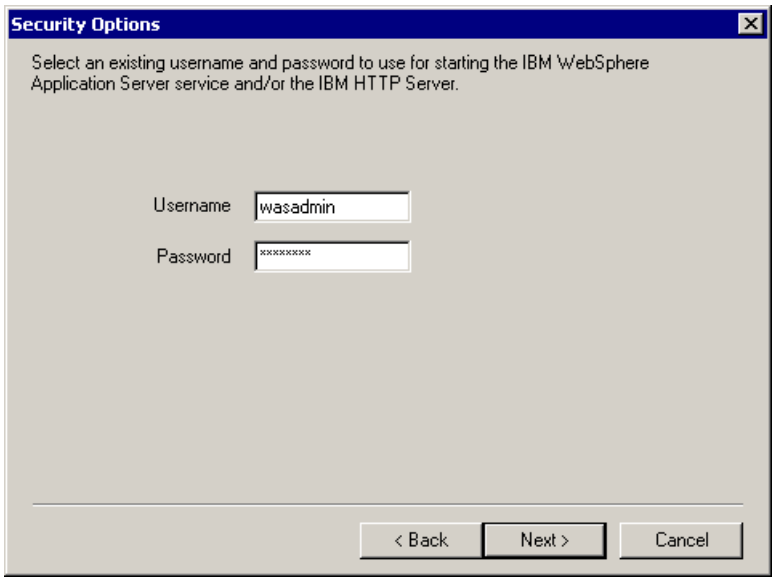


Figure 9-21 Specify username and password for WebSphere services

10. In the Product Directory window, select the destination directory, then click **Next**. We selected **D:\WebSphere\AppServer**.

Note: Although WebSphere can be installed at any location, this chapter assumes that WebSphere will be installed in the path: D:\WebSphere\AppServer.

11. In the Database Options window, shown in Figure 9-22 on page 228, enter the following and then click **Next**:
    - Database Type: DB2
    - Database Name: <was_database>

      Enter the name of the database in the DB2 database server, as created in 9.4.5, “Set up the WebSphere administrative database” on page 214.
    - Database User ID: <was_user>
    - Password: <was_password>
    - Path: <db2_install_path>

      Since the WebSphere installation program automatically determines the location of the DB2 database software, accept the default value here.
    - Remote Database: leave unselected

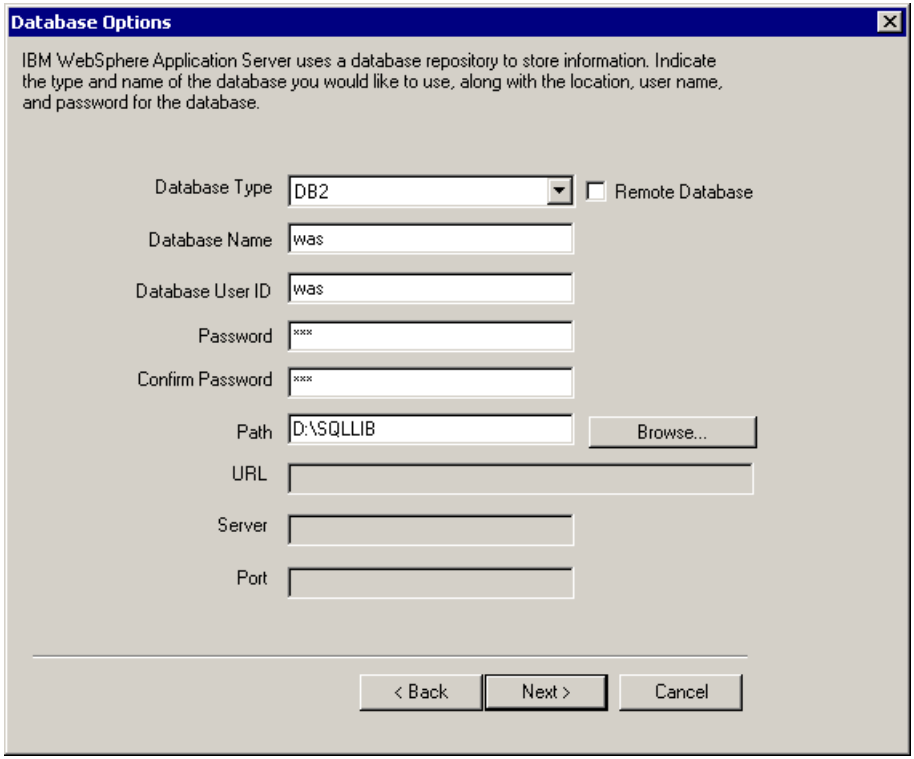


Figure 9-22 Specify Database Options for DB2

Important: If an alert is displayed indicating that the DB2 does not meet the requirements of WebSphere, you will need to exit the WebSphere installation and update the DB2 software (server or client) to the required release level. Once completed, restart the WebSphere installation from the beginning.

12. In the Select Program Folder window, accept the default and click **Next**.
13. In the Install Options Selected window, check that the selected options are the same as in Figure 9-23 on page 229. If the same, click **Next** to start the installation process.

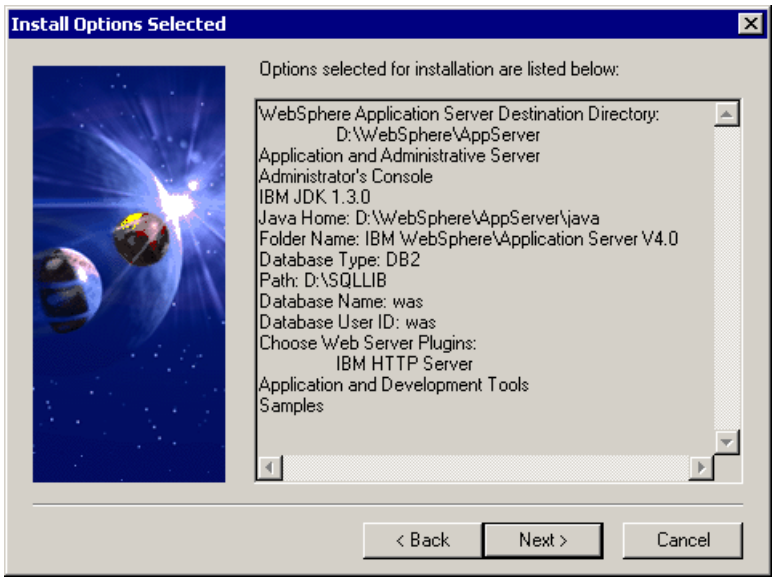


Figure 9-23 Check selected WebSphere install options

14. In the Setup Complete window, click **Finish** to complete the installation.
15. In the Restarting Windows window, select **No, I will restart my computer later** and then click **OK**.
16. Rename the createwasdb.scr file in the <WAS_HOME>\bin directory to createwasdb.bak.
17. To verify that WebSphere Application Server has installed properly, check <WAS_HOME>\logs\wssetup.log and ensure that no errors have occurred (`Total Errors:0`). There should be an `Install Complete` statement at the end of the log to indicate a successful installation.
18. Reboot the system.

9.6.3 Verify the WebSphere installation

In order to verify the installation of IBM WebSphere Application Server V4.0, Advanced Edition, the following tasks must be completed in order:

1. Check the installation log.
2. Check that Windows services are present.
3. Check the admin.config settings.
4. Check the setupCmdLine.bat settings.
5. Check the Web server configuration file changes.

6. Start the WebSphere administrative server processes.
7. Start WebSphere Default Server.
8. Regenerate Web server plug-in settings.
9. Restart Web server processes.
10. Verify Web server plug-in configuration.

Check the installation log

Check that the installation log <WAS_HOME>\logs\wssetup.log does not contain any errors.

Check that Windows services are present

Check that the Windows service shown in Table 9-6 is present.

Table 9-6 WebSphere Windows services

Service name	Status	Startup mode	Log on as...
IBM WS AdminServer 4.0	-	manual	wasadmin

Check the admin.config settings

To check the admin.config settings, perform the following steps:

1. Check that the repository database settings are correct for the database type (DB2), instance (was), and user ID (was) used in our test environment:

```
com.ibm.ejs.sm.adminServer.dbdataSourceClassName=COM.ibm.db2.jdbc.DB2Connec
tionPoolDataSource
com.ibm.ejs.sm.adminServer.dbserverName=
com.ibm.ejs.sm.adminServer.dbportNumber=
com.ibm.ejs.sm.adminServer.dbdatabaseName=<was_database>
com.ibm.ejs.sm.adminServer.dbuser=<was_user>
com.ibm.ejs.sm.adminServer.dbpassword=<was_password>
com.ibm.ejs.sm.adminServer.dbdisable2Phase=true
```

2. Check that the DB2 path-related parameters listed in Table 9-7 contain the absolute path of the DB2 java directory: <db2_install_path>\java.

Table 9-7 DB2-related paths required in admin.config

Parameter	Must contain path...
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs	D:/SQLLIB/java/db2java.zip
com.ibm.ejs.sm.util.process.Nanny.path	D:\\SQLLIB\\bin D:\\SQLLIB\\function

3. Check that the WebSphere schema and initial configuration (for example, the Default Server) will be written to the repository database on startup, as listed in Table 9-8.

Table 9-8 Required schema creation and initial configuration flags

Parameter	Required value...
com.ibm.ejs.sm.adminServer.createTables	true
install.initial.config	true

Check the setupCmdLine.bat settings

To check the setupCmdLine.bat settings, perform the following steps:

1. Check that the database driver setting is correct for the database type (DB2), for example:

```
SET DBDRIVER_JARS=D:\SQLLIB\java\db2java.zip
```

Check the Web server configuration file changes

To check the Web server configuration file changes, complete the following steps:

1. Check that following required settings have been added to the IBM HTTP Server configuration file (httpd.conf) as a result of the WebSphere installation:

```
LoadModule ibm_app_server_http_module
D:/WebSphere/AppServer/bin/mod_ibm_app_server_http.dll
WebSpherePluginConfig D:\WebSphere\AppServer\config\plugin-cfg.xml
```

2. If not, manually add the above lines to the end of the httpd.conf file and save the changes.

Start the WebSphere administrative server processes

The WebSphere administrative server needs to be started in order to test the installation as well as the connectivity between WebSphere and the WebSphere administrative database hosted in DB2:

1. Run the WebSphere administrative server by issuing the following command:

```
<WAS_HOME>\bin\adminserver.bat
```

2. The startup of WebSphere administrative server is successful if the following conditions are met:
   a. The IBM WS AdminServer 4.0 entry in the Windows 2000 services control panel shows a status of Started.
   b. The last line of the <WAS_HOME>\logs\tracefile file is similar to the following:

```
[01.10.19 15:19:15:415 EDT] 1649bd69 Server        A WSVR0023I: Server
__adminServer open for e-business
```

Start WebSphere Default Server

The WebSphere installation type (Typical) chosen for our test environment installs a default application server (Default Server) into the WebSphere administrative domain. We use this application server and its Web applications to verify that the WebSphere installation is working correctly.

To start the Default Server, perform the following steps:

1. Start the WebSphere Administrative Console by issuing the following command:

   ```
   <WAS_HOME>\bin\adminclient.bat
   ```

2. Right-click **Default Server** under the <hostname> node and select **Start** from the pop-up menu, as shown in Figure 9-24.

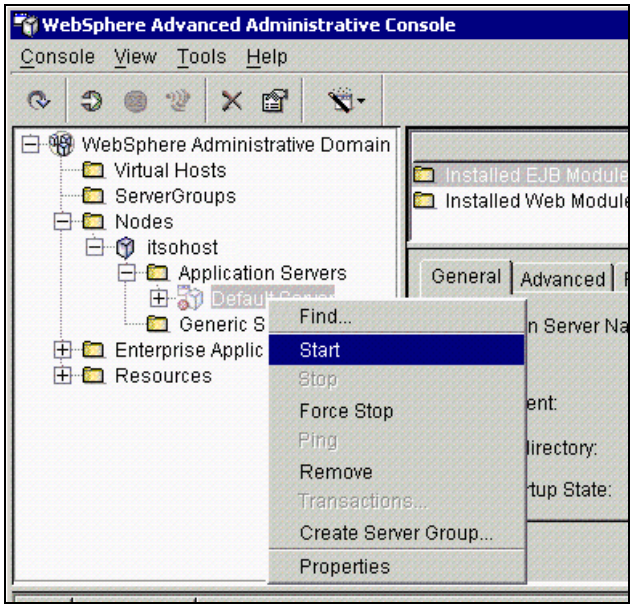


Figure 9-24 Starting the WebSphere Default Server

3. The Default Server has been successfully started if the following conditions are met:

   a. The last line in the administrative console event messages pane (shown in Figure 9-25) shows the following message:

      ```
      Transport http is listening on port 9,080.
      ```

b. The last line of the <WAS_HOME>\logs\Default_Server_stdout.log file is similar to the following:

```
[01.10.23 21:18:36:750 EDT] 57834a95 Server        A WSVR0023I: Server
Default Server open for e-business
```

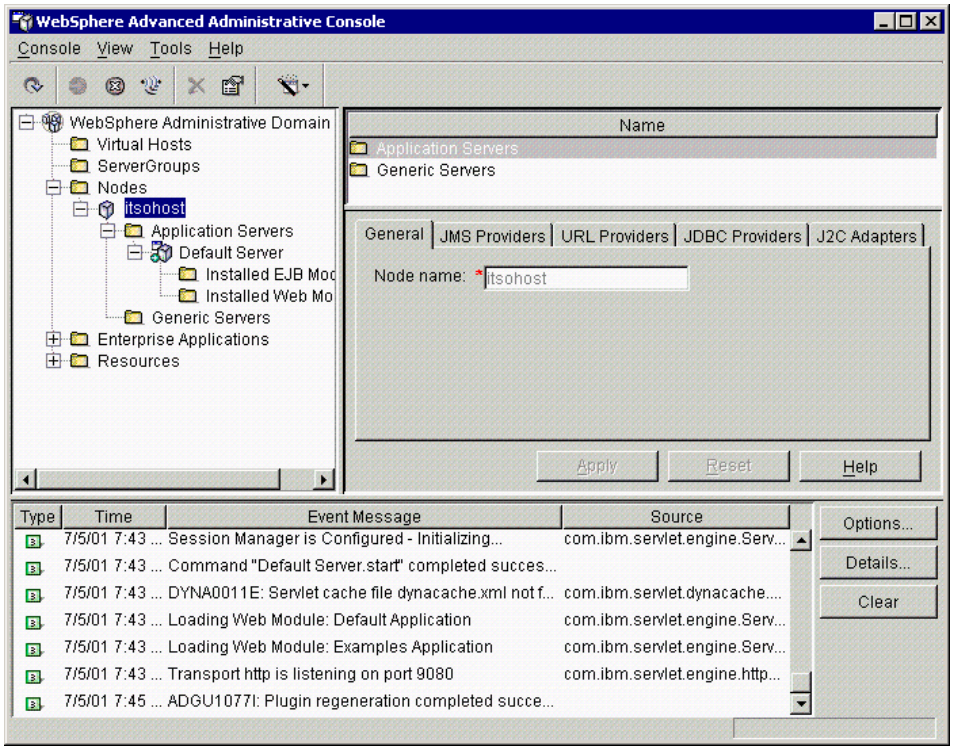


Figure 9-25 Successful startup of application server

4. Verify that the Default Server Web container has been properly installed and configured by accessing its servlets through the Web server “embedded” within the WebSphere V4.0 Web container:

   a. Using a Web browser, request the following URL:

      `http://<hostname>:9080/servlet/snoop`

      A window similar to the one shown in Figure 9-26 on page 234 should be displayed in your browser.

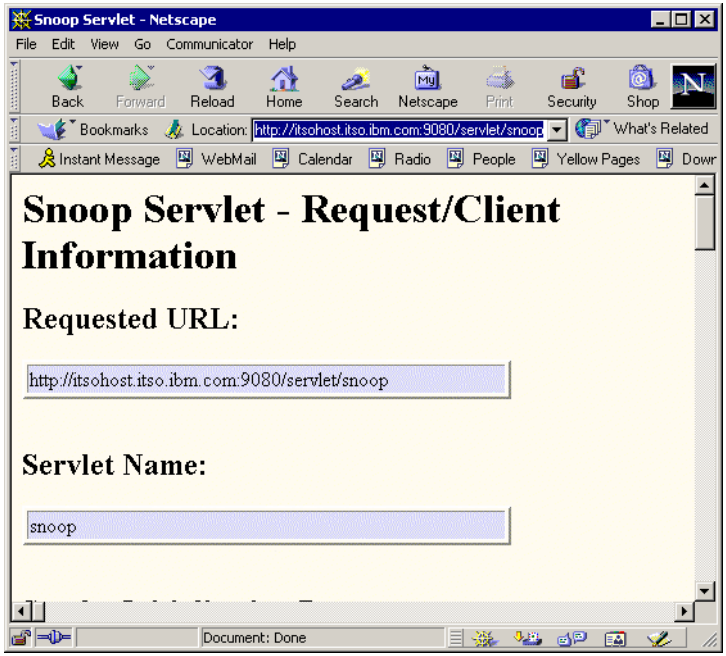


Figure 9-26 Snoop servlet accessed through embedded Web server

b. Using a Web browser, request the following URL:

```
http://<hostname>:9080/webapp/examples/showCfg
```

Note: The embedded Web server is a new feature introduced with WebSphere V4.0. In previous releases, a stand-alone Web server was required in order to access any resource hosted in WebSphere.

5. Close the WebSphere Administrative Console.

Regenerate the Web server plug-in settings

Before the Default Server can be accessed from a stand-alone Web server (such as IBM HTTP Server) the Web server plug-in settings file <WAS_HOME>\config\plugin-cfg.xml must be regenerated to reflect the following settings used by the Web server plug-in:

- Virtual host settings
- Application server transports
- Web container URIs

Perform the following steps:

1. Run the WebSphere Administrative Console by issuing the following command:

   ```
   <WAS_HOME>\bin\adminclient.bat
   ```

2. Right-click the node <hostname> that contains the Default Server application server and select **Regen Webserver Plugin** from the pop-up menu, as shown in Figure 9-27.

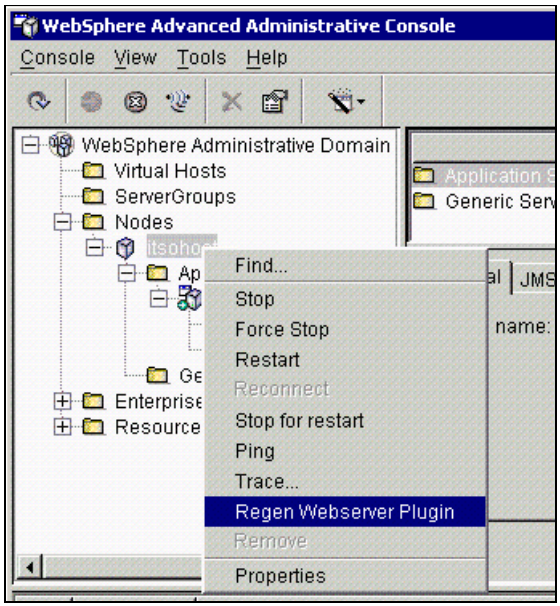


Figure 9-27 Regenerate Web server plug-in settings

Tip: WebSphere provides a command-line tool that can be used to regenerate the Web server plug-in configuration without having to run the WebSphere Administrative Console:

```
<WAS_HOME>\bin\GenPluginCfg.bat -adminNodeName <hostname>
```

3. Check that the content of the <WAS_HOME>\config\plugin-cfg.xml file has been updated to include the URIs of servlets contained within Default Server.

Tip: The plug-in regeneration command generates a <Server> element for the Default Server that contains a CloneID attribute:

```
<Server CloneID="stsul7n0" Name="Default Server">
   <Transport Hostname="itsohost" Port="9080" Protocol="http"/>
</Server>
```

In a non-cloned environment this attribute can be removed, resulting in performance improvements in the Web server plug-in.

Restart the Web server processes

The IBM HTTP Server process must be restarted before the Web server plug-in configuration can be tested.

1. Issue the commands:

```
D:\> net stop "IBM HTTP Server"
D:\> net start "IBM HTTP Server"
```

Verify the Web server plug-in configuration

The Web server plug-in configuration can be verified by requesting a servlet through the Web server that has already been successfully requested through the Web container's embedded Web server:

1. Using a Web browser, request a servlet URL, such as:

```
http://<hostname>/servlet/snoop
```

or:

```
http://<hostname>/webapp/examples/showCfg
```

9.7 Install the WebSphere plug-in on the remote Web server

This section provides detailed instructions for installing, configuring, and verifying the WebSphere HTTP plug-in on a remote Web server.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install WebSphere plug-in.
3. Verify WebSphere plug-in installation.
4. Configure for remote WebSphere plug-in.

5. Verify remote plug-in configuration.

9.7.1 Preinstallation tasks

The plug-in installation updates the httpd.conf configuration file, so the IBM HTTP Server processes on the Web server machine must be stopped before installing the plug-in.

To stop the IBM HTTP Server processes, issue the command:

```
D:\> net stop "IBM HTTP Server"
```

9.7.2 Install the WebSphere plug-in

To install the WebSphere plug-in, complete the following steps on the Web server machine:

1. Log in as an administrator user in the local server domain (not part of a Windows domain).
2. Insert the IBM WebSphere Application Server V4.0, Advanced Edition CD.
3. Start the WebSphere Application Server installation by double-clicking **Setup** from the root of the CD.
4. In the Choose Setup Language window, select your national language from the drop-down menu (English is selected by default) and click **OK**.
5. In the WebSphere Application Server Attention window, read the warnings and then click **Next** to continue.
6. In the Installation Options window, select **Custom Installation**, and then click **Next**.
7. In the Choose Application Server Components window, select only the Web server plug-in option, and then click **Next**.
8. In the Choose Web Server Plugins window, shown in Figure 9-28 on page 238, select only the **IBM HTTP Server** option and then click **Next**.

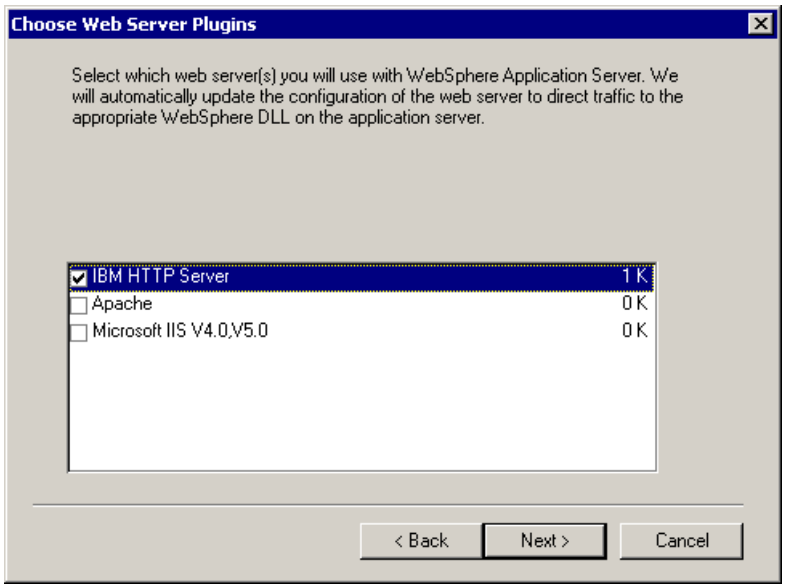


Figure 9-28 Choose Web server plug-ins to install

9. In the Security Options window, enter the username and password of the Windows account under which the IBM HTTP Server is run as a service. Click **Next** to continue.
10. In the Product Directory window, select the destination directory, then click **Next**. We selected **D:\WebSphere\AppServer**.

Important: WebSphere and the WebSphere HTTP plug-in can use different installation paths on the two servers. However, for simplicity and to reduce the amount of editing required for the plugin-cfg.xml file, we recommend that the plug-in installation use the same path as the full WebSphere installation.

11. In the Database Options window, accept the defaults and click **Next**. None of the database settings are actually used by the Web server plug-in.
12. In the Select Program Folder window, accept the default and click **Next**.
13. In the Install Options Selected window, shown in Figure 9-29 on page 239, check that the selected components are correct. If yes, click **Next** to start the installation process.

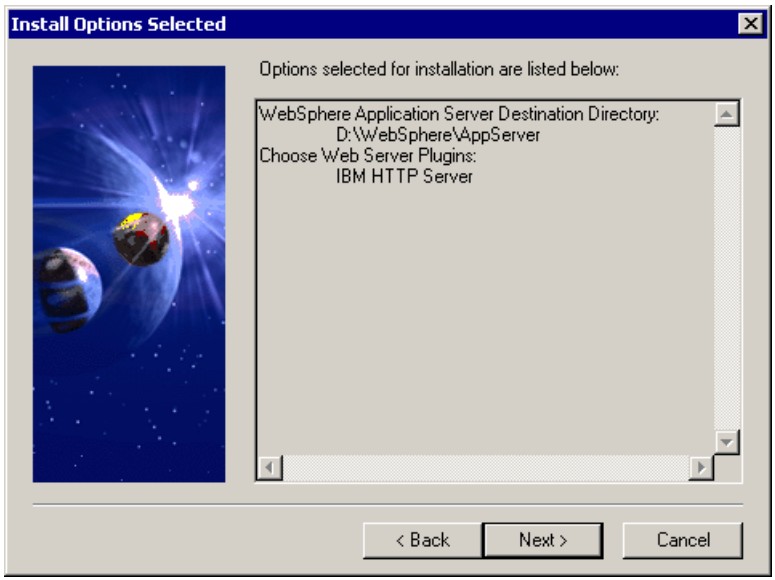


Figure 9-29 Check selected WebSphere plug-in install options

14. In the Setup Complete window, click **Finish** to complete the installation.
15. In the Restarting Windows window, select **Yes, I will restart my computer now**, and then click **OK**.

9.7.3 Verify the WebSphere plug-in installation

In order to verify the installation of the WebSphere HTTP plug-in, on the Web server machine check Web server configuration file changes.

Check the Web server configuration file changes

To check that required settings have been added to the IBM HTTP Server configuration file (httpd.conf), follow the steps described in “Check the Web server configuration file changes” on page 231.

9.7.4 Configure for the remote WebSphere plug-in

In order to support requests from a WebSphere HTTP plug-in installed on a remote Web server, the following tasks must be performed:

1. Configure the WebSphere virtual host.
2. Regenerate the Web server plug-in settings.
3. Copy the plug-in settings to the remote server.

4. Restart Web server.

Configure the WebSphere virtual host

WebSphere uses virtual hosts to map HTTP requests for a particular host name and port number to selected Web applications. To map requests from the remote Web server to the required virtual host, complete the following steps:

1. Log in as a local administrator on the WebSphere Application Server server machine.
2. Run the WebSphere Administrative Console by issuing the following command:

   ```
   <WAS_HOME>\bin\adminclient.bat
   ```

3. Select the **Virtual Hosts** folder in the tree pane of the administrative console.
4. In the details pane, select the **default_host** virtual host.
5. Add three new entries to the Host Aliases list of the default_host virtual host, as shown in Figure 9-30:

   ```
   <Web server hostname>:<port#>
   <Web server hostname.domain.com>:<port#>
   <Web server IP address>:<port#>
   ```

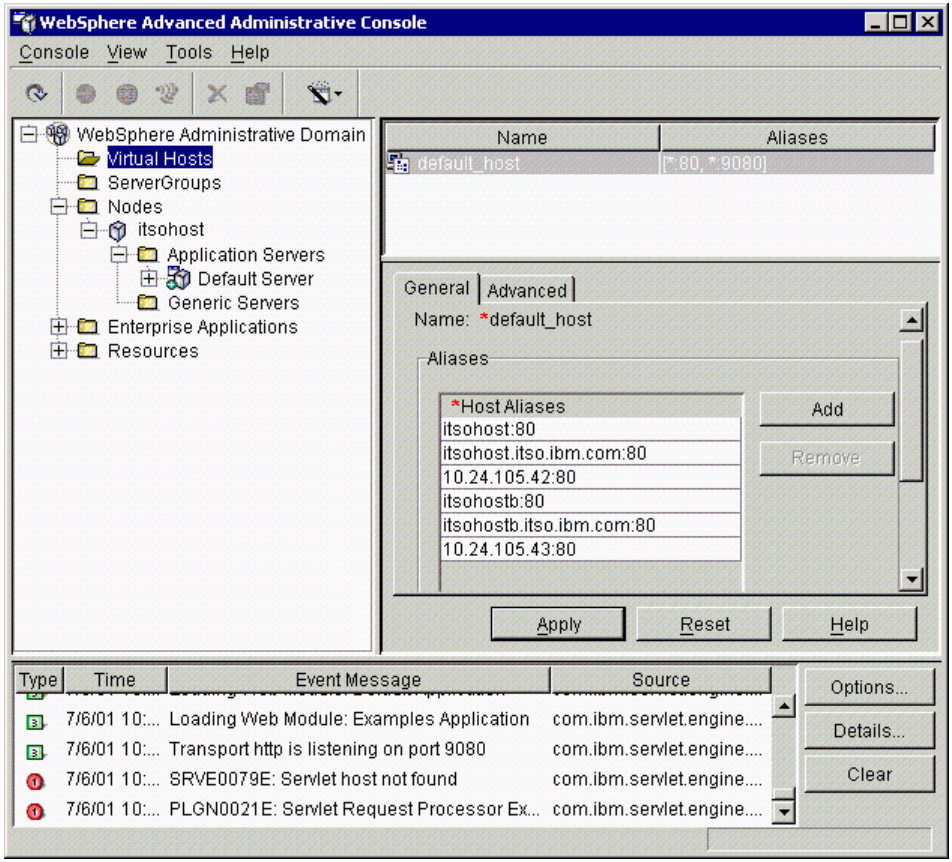


Figure 9-30 Add required aliases to virtual host

6. Click **Apply** to save changes.

Important: Always remember to click the **Apply** button when changing settings with the administrative console. Failure to do so will result in all changes being lost, even if you just click into another resource in the console.

7. Restart each of the application servers that are associated with this virtual host.

Regenerate the Web server plug-in settings

Before the Default Server can be accessed from a remote Web server, the Web server plug-in settings file needs to be regenerated. To regenerate the Web server plug-in settings:

1. Follow the steps described in “Regenerate the Web server plug-in settings” on page 234.
2. Check that the content of the <WAS_HOME>\config\plugin-cfg.xml file has been updated to include the remote Web server’s host name among the virtual hosts settings.

Copy the plug-in settings to the remote server

Next, the regenerated Web server plug-in settings must be copied to the remote Web server:

1. Copy the <WAS_HOME>\config\plugin-cfg.xml file from the WebSphere server machine across to the <WAS_HOME>\config directory on the remote Web server machine

Important: WebSphere and the WebSphere HTTP plug-in can have different installation paths on the two servers. However, for simplicity and to reduce editing of the plugin-cfg.xml file, we recommend that the plug-in installation use the same path as the full WebSphere installation.

Restart the Web server

Restart the Web server if you want it to load the new plug-in settings immediately, or wait until the plug-in dynamically reloads. (The default refresh interval is 60 seconds.)

1. Restart the Web server by issuing the following command:

```
D:\> net start "IBM HTTP Server"
```

9.7.5 Verify the remote plug-in configuration

In order to verify the configuration of the WebSphere HTTP plug-in installed on a remote Web server, the following tasks must be performed:

1. Check the plug-in logs.
2. Test the connection to WebSphere.

Check the plug-in logs

To check the plug-in logs, complete the following steps:

1. Log in as a local administrator on the Web server machine.
2. The location of the WebSphere HTTP plug-in’s log is specified by the <Log> element of the plugin-cfg.xml configuration file. By default, this is set to the following:

```
<Log LogLevel="Warning" Name="<plugin_install_path>\logs\native.log"/>
```

3. Check the contents of this log file. If the plug-in has been correctly configured, then the following lines will be written to the end of the file:

```
[Tue Jul 03 11:45:03 2001] 000000df 000000e3 - PLUGIN: Plugins loaded.
[Tue Jul 03 11:45:03 2001] 000000df 000000e3 - PLUGIN:
--------------------System Information----------------------
[Tue Jul 03 11:45:03 2001] 000000df 000000e3 - PLUGIN: Bld date: Jun 27
2001, 17:50:28
[Tue Jul 03 11:45:03 2001] 000000df 000000e3 - PLUGIN: Web server:
IBM_HTTP_SERVER/1.3.19  Apache/1.3.19 (Win32)
[Tue Jul 03 11:45:03 2001] 000000df 000000e3 - PLUGIN: Hostname = ITSOHOST
[Tue Jul 03 11:45:03 2001] 000000df 000000e3 - PLUGIN: OS version 4.0,
build 1381, 'Service Pack 6'
[Tue Jul 03 11:45:03 2001] 000000df 000000e3 - PLUGIN:
------------------------------------------------------------
```

Note: See Appendix C, “The plugin-cfg.xml file definitions” on page 1071 for more details.

Test the connection to WebSphere

To test the remote Web server connection the WebSphere, complete the following steps:

1. Using a Web browser, request the following URL:

   ```
   http://<Web server hostname>/servlet/snoop
   ```

2. If both the Web server and WebSphere have been correctly configured to support remote HTTP plug-in access, then a window similar to Figure 9-31 will be obtained.

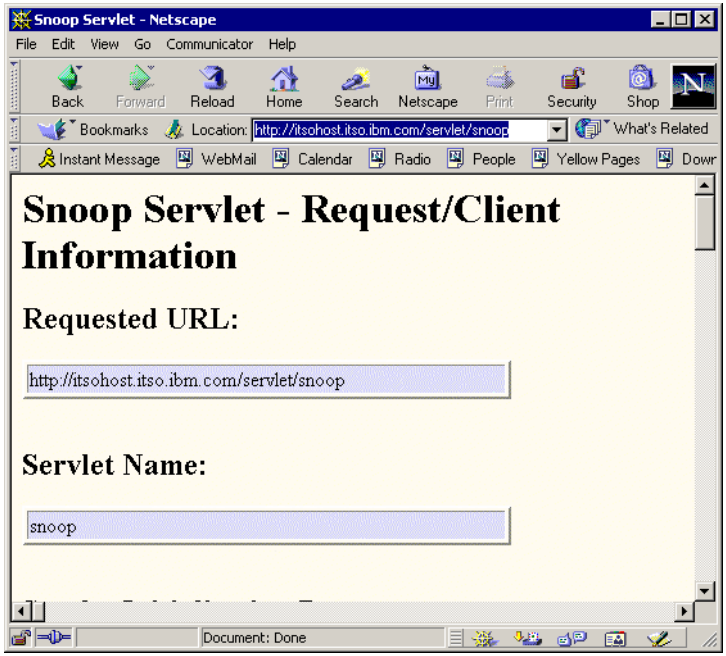


Figure 9-31 Request handled through the remote HTTP plug-in

9.8 Install the new WebSphere node into the existing domain

This section provides detailed instructions for installing, configuring, and verifying a new WebSphere node into an existing WebSphere V4.0 administrative domain.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install WebSphere.
3. Verify WebSphere installation.

9.8.1 Preinstallation tasks

Prior to installing the new WebSphere node, the basic preinstallation tasks described in 9.6.1, “Preinstallation tasks” on page 222 need to be completed.

Important: This section assumes that the IBM HTTP Server and DB2 client (in place of the DB2 server) are both already installed and configured on the server on which you will be installing this new WebSphere node.

9.8.2 Install WebSphere

To install the new WebSphere node into an existing WebSphere domain, complete the basic installation tasks specified in 9.6.2, “Install WebSphere” on page 223, except for the following:

1. In the Database Options window, specify:
   - Database Name: specify DB2 TCPIP alias to the remote database.
   - Path: specify the installation path of the DB2 client software (this should be automatically detected and presented as the default).
   - Remote Database: leave this unselected. The DB2 client will be configured to present a “local” alias to the remote DB2 database. See 9.5, “Install the database client” on page 216 for details.

9.8.3 Verify the WebSphere installation

In order to verify the new WebSphere node installed into an existing WebSphere domain, complete the basic verification tasks specified in 9.6.3, “Verify the WebSphere installation” on page 229, except for the following:

1. For each new node sharing an existing administrative database, ensure the database tables are not created and the default resources are not installed a second time. Edit the <WAS_HOME>\bin\admin.config file and set the values of the following settings as shown:

```
install.initial.config=false
com.ibm.ejs.adminServer.createTables=false
```

9.9 Configure WebSphere HTTP transport for SSL

Each transport of a WebSphere V4.0 Web container can be configured to support SSL encryption of the HTTP communication (HTTPS). The SSL functionality can be run in one of two modes:

- Client authentication disabled
- Client authentication enabled

Client authentication disabled only requires that the server (WebSphere) send a certificate to the client (WebSphere HTTP plug-in) in order to authenticate itself during SSL handshaking. This configuration requires the following tasks:

1. Creation of a JKS format keyfile (key store) for use by the WebSphere transport.
2. Creation of a self-signed certificate stored in the JKS keyfile.
3. Configuration of a new (or reconfiguration of existing) WebSphere transport to use SSL encryption and the above keyfile for the required keyfile settings.
4. Creation of a CMS format keyfile (key store) for use by the plug-in.
5. Import of the server's self-signed certificate as a trusted CA (certificate authority) within the plug-in's keyfile.

Client authentication enabled is an extension of the client authentication disabled case. In this case, the client (plug-in) is also required to send a certificate to the server in order to authenticate itself during SSL handshaking. This configuration requires the following additional tasks to the client authentication disabled case:

6. Creation of the self-signed certificate in the plug-in's CMS keyfile.
7. Import of the client's self-signed certificate as a trusted CA (certificate authority) into the server's keyfile.
8. Reconfiguration of the transport's SSL settings to specify the JKS keyfile as the trustfile, as well as enabling of the client authentication option.

In either case, after performing the above tasks, the Web server plug-in configuration must then be updated by performing these remaining tasks:

9. Regenerate the Web server plug-in configuration file.
10. Restart the Web server.

9.9.1 Client authentication disabled

To configure a WebSphere HTTP transport for SSL with client authentication disabled, complete the following tasks.

Create the WebSphere keystore

To create a WebSphere keystore database file, complete the following steps on the WebSphere server machine:

1. Log in as a local administrator.
2. Start the WebSphere IBM Key Management Utility by clicking **Start** -> **Programs** -> **IBM WebSphere** -> **Application Server V4.0 AE** -> **IKeyMan**.

3. Select **Key Database File** from the menu bar, then select **New**.
4. In the New window, enter the following settings and then click **OK**:
   - Key Database Type: `JKS`
   - File Name: `server.jks`
   - Location: `<WAS_HOME>\etc\`
5. In the Password Prompt window, enter the following, then click **OK** to continue
   - Password: password to protect keystore file contents
   - Check **Set expiration time?** and enter the number of days before the password should expire. If no expiration is required, uncheck this setting.

Note: Although not required in a development environment, it is strongly recommended that all keystores used in a production environment set an expiration period. Also, remember this password, because it will be needed for the WebSphere transport settings window in a later step.

6. Close the keystore database file.

Create a new self-signed server certificate

To create a new self-signed certificate, perform the following steps on the WebSphere server machine:

1. Log in as a local administrator.
2. Start the WebSphere IBM Key Management Utility by clicking **Start** -> **Programs** -> **IBM WebSphere** -> **Application Server V4.0 AE** -> **IKeyMan**.
3. Select **Key Database File** from the menu bar, then select **Open**.
4. Specify the <WAS_HOME>\etc\server.jks file.
5. Select **Create** from the menu bar, then select **New Self-Signed Certificate**.

Note: If you are enabling SSL for a production environment, select **New Certificate Request** instead. It is strongly recommended that self-signed digital certificates not be used in production.

6. In the Create New Self-Signed Certificate window, shown in Figure 9-32 on page 248, enter the following values, then click **OK**.
   - Key Label: `<user-defined label>`
   - Version: `X509 V3`
   - Key Size: `1024`

- Common Name: `<hostname.domain.com>`
- Organization: `IBM`
- Organization Unit: `ITSO`

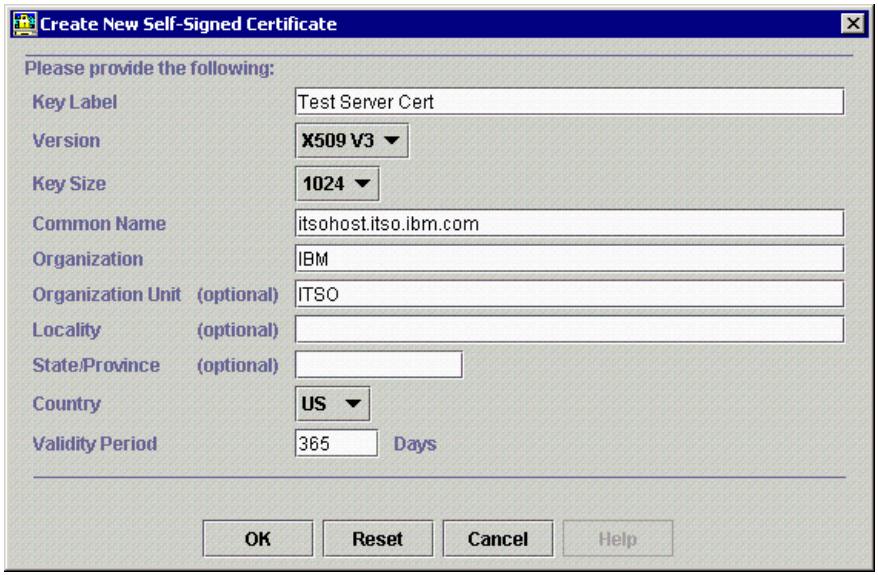


Figure 9-32 Self-signed certificate settings

7. The certificate will be listed in the Personal Certificates pane of the utility.
8. Close the keystore and exit the WebSphere IBM Key Management Utility.

Configure WebSphere transport

To create a new WebSphere transport and configure it to use SSL encryption, perform the following steps on the WebSphere server machine:

1. Log in as a local administrator.
2. Start the WebSphere Administrative Console by clicking **Start** -> **Programs** -> **IBM WebSphere** -> **Application Server V4.0 AE** -> **Administrator's Console**.
3. Right-click the **Default Server** application server and select the **Stop** from the pop-up menu.
4. Select the **Services** tab of the Default Server settings in the properties pane.
5. Select the **Web Container Service** entry from the Service List, then click **Edit Properties**, as shown in Figure 9-33 on page 249.

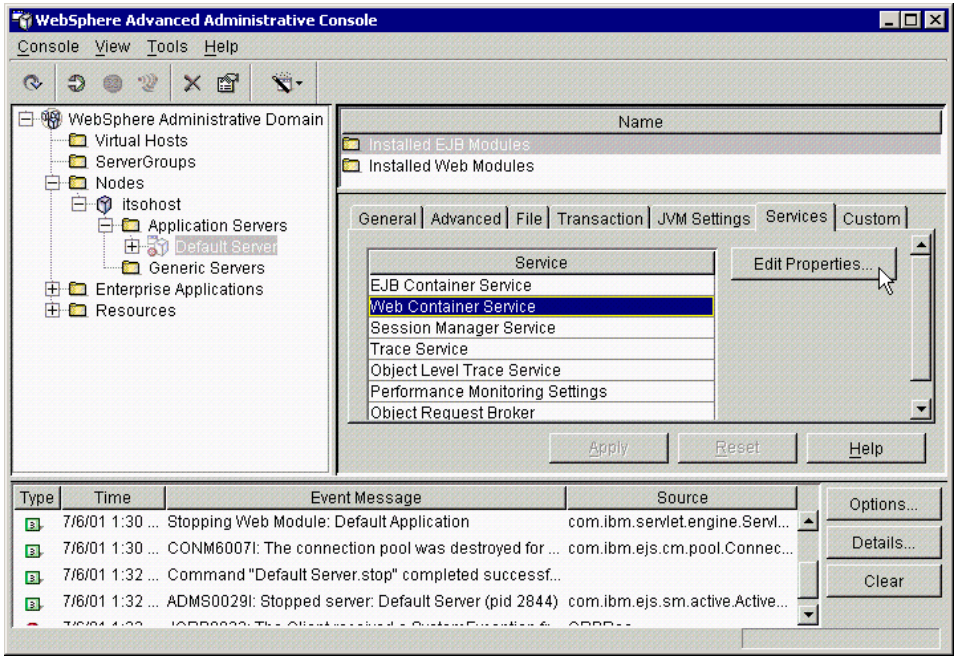


Figure 9-33 Select Web Container service settings

6. In the Web Container Service window, shown in Figure 9-34 on page 250, select the **Transport** tab, then click **Add**.
7. In the HTTP Transport Properties window, select the **General** tab, then enter the following information. Once complete, click **OK**.
   - Transport host: *
   - Transport port: `9081` (must not be in use on WebSphere server machine)
   - Enable SSL: enable this setting
   - Use global SSL default configuration: disable this setting - we will be using our own keystore and settings
   - Key file name: <WAS_HOME>\etc\server.jks
   - Key file password: password used to protect server.jks
   - Confirm password: password used to protect server.jks
   - Key file format: `JKS`
   - Enable client authentication: disable this setting
   - Security level: `HIGH`

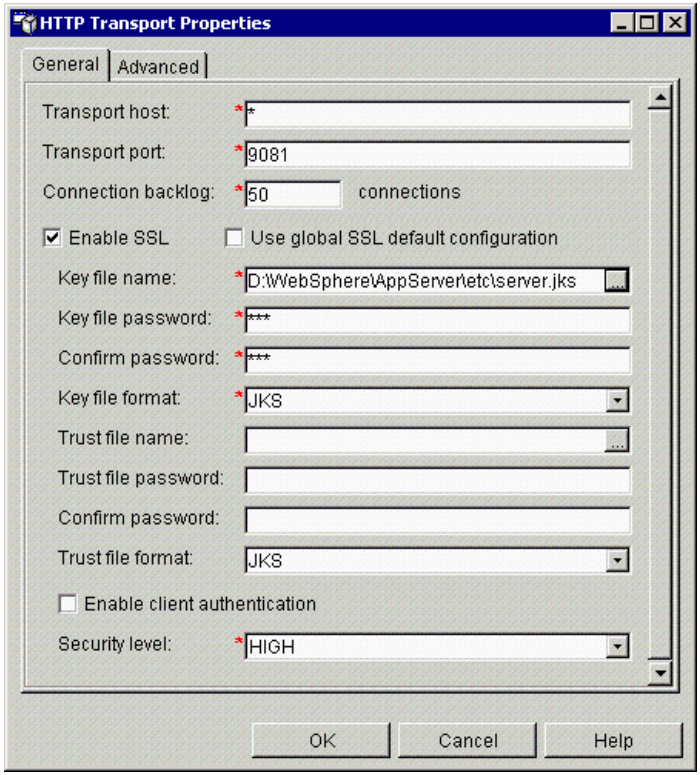


Figure 9-34 Enable SSL encryption (client authentication disabled) for HTTP transport

8. In the Web Container Service window there is now a new (9081) transport, as shown in Figure 9-35 on page 251. Click **OK**.

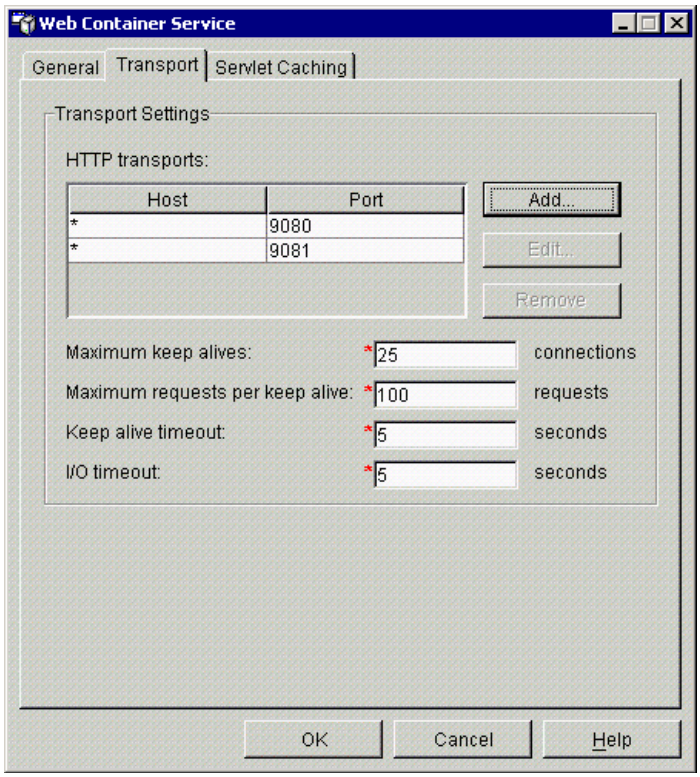


Figure 9-35 Web Container Service with a new transport

9. In the properties pane of the main administrative console window, click **Apply** to save changes to the Default Server configuration.
10. Right-click the **Default Server** application server and select **Start** from the pop-up menu.
11. The status of the new HTTPS transport will be indicated by an entry in the console's event messages. If execution is successful, an entry similar to the one highlighted in Figure 9-36 on page 252 should be displayed.

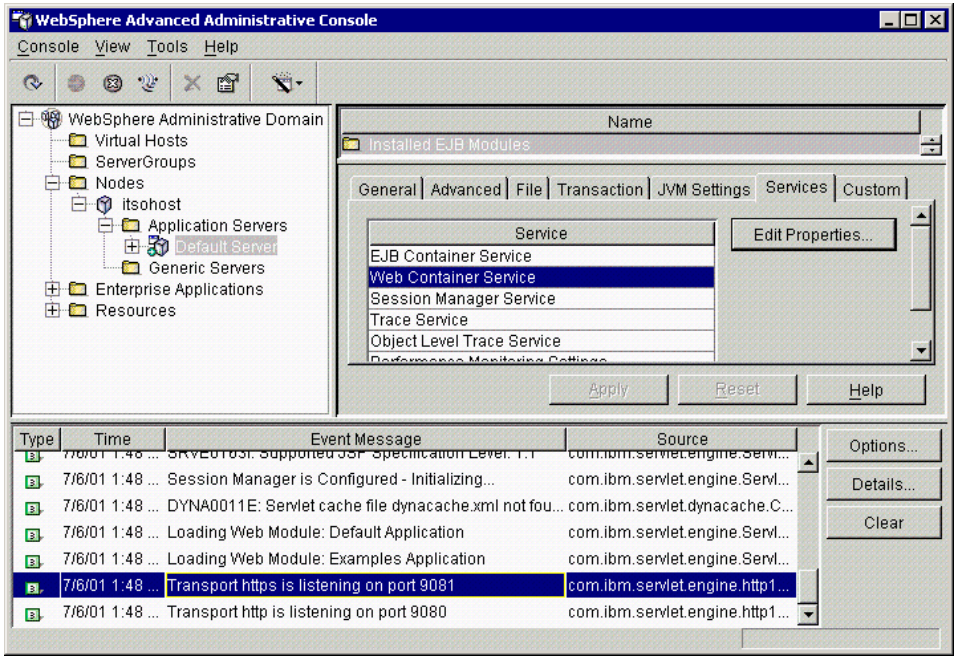


Figure 9-36 Successful execution of HTTPS transport on port 9081

Create a plug-in keystore

Note: You can create your own plug-in keystore as described here, or you can use the default plug-in keystore, <WAS_HOME>/etc/plugin-key.kdb. However, when using your own keystore file name and location, you need to manually edit (and maintain) plugin-cfg.xml.

To create a new plug-in keystore database file, complete the following steps on the Web server machine:

1. Log in as a local administrator.
2. Start the Web server IBM Key Management Utility by clicking **Start** -> **Programs** -> **IBM HTTP Server** -> **Start Key Management Utility**.
3. Select **Key Database File** from the menu bar, then select **New**.
4. In the New window, enter the following and then click **OK**:
   - Key Database Type: CMS key database file
   - File Name: client.kdb
   - Location: <plugin_install_path>\etc\

5. In the Password Prompt window, enter the following, then click **OK** to continue.
   - Password: password to protect keystore file contents
   - Check **Set expiration time?** and enter number of days before the password should expire. If no expiration is required, uncheck this setting.

     Tip: Although not required in a development environment, it is strongly recommended that all keystores used in a production environment set an expiration period.

   - Check **Stash the password to a file?**

     Important: The IBM HTTP Server accesses the password-protected keystore file <filename>.kdb using the password contained in the <filename>.sth stashfile. Consequently, the stash option must be enabled.

6. When the Information window appears with the following message, click **OK**:

   ```
   The password has been encrypted and saved in the file:
   <plugin_install_path>\etc\client.sth
   ```

7. Close the keystore and exit the IBM Key Management Utility.

Import the server certificate as CA

In order to import the self-signed server certificate as a trusted CA into the plug-in's keystore, complete the following steps:

1. Start the WebSphere IBM Key Management Utility by clicking **Start** -> **Programs** -> **IBM WebSphere** -> **Application Server V4.0 AE** -> **IKeyMan**.
2. Select **Key Database File** from the menu bar, then select **Open**.
3. Specify the <WAS_HOME>\etc\server.jks file.
4. Select the certificate, then click **Extract Certificate...** to export the certificate as a Base64 encoded ASCII file.
5. In the Extract Certificate to a File window, enter the following values, then click **OK**.
   - Data type: `Base64-encoded ASCII data`
   - Certificate file name: `server-cert.arm`
   - Location: `<WAS_HOME>\etc\`
6. Close the WebSphere keystore and Key Management Utility.

7. Start the Web server IBM Key Management Utility by clicking **Start** -> **Programs** -> **IBM HTTP Server** -> **Start Key Management Utility**.
8. Select **Key Database File** from the menu bar, then select **Open**.
9. Specify the <plugin_install_path>\etc\client.kdb file.
10. Goto the Signer Certificates pane, then click **Add**, as shown in Figure 9-37.

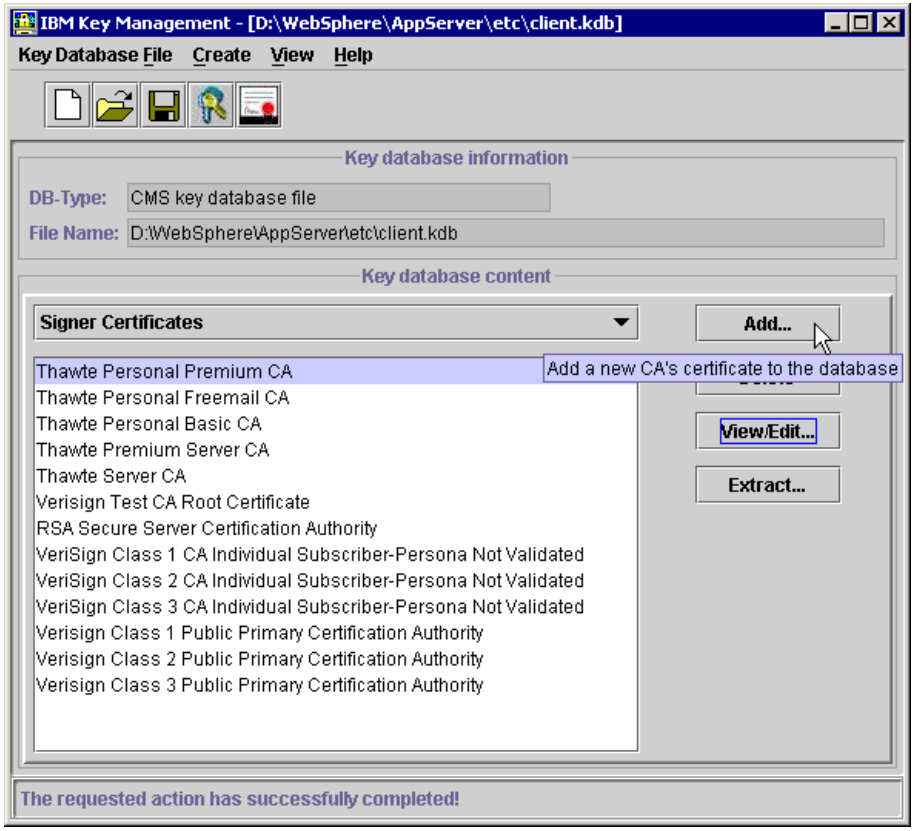


Figure 9-37 Import a new CA certificate into plug-in keystore

11. In the Add CA's Certificate from a File window, enter the following and then click **OK**.
    - Data type: Base64-encoded ASCII data
    - Certificate file name: server-cert.arm
    - Location: <WAS_HOME>\etc\
12. The server's CA certificate will now be listed with the plug-in keystore trusted CAs.

9.9.2 Client authentication enabled

To configure a WebSphere HTTP transport for SSL with client authentication enabled, complete the following additional tasks to the client authentication disabled case.

Create a new self-signed client certificate

Follow the method described in “Create a new self-signed server certificate” on page 247 for the creation of the self-signed server certificate.

Important: Only use the Web server GSK IKeyMan utility to manipulate the CMS format plug-in keystore file.

Import a client certificate as CA

Follow the method described in “Import the server certificate as CA” on page 253 for the import of the self-signed server certificate into the plug-in’s keystore.

Important: The appropriate IKeyMan utility must be used for each of the keystore format files:

- Server keystore (JKS format) - use WebSphere IKeyMan utility
- Plug-in keystore (CMS format) - use Web server GSK IKeyMan utiity

Reconfigure WebSphere transport

Follow the method described in “Configure WebSphere transport” on page 248 for the setup of an SSL enabled transport on the Default Server Web container, except with the following changes:

1. In the HTTP Transport Properties window, shown in Figure 9-38, add or change the following settings:
   - Trust file name: <WAS_HOME>\etc\server.jks
   - Trust file password: password used to protect server.jks
   - Confirm password: password used to protect server.jks
   - Trust file format: JKS
   - Enable client authentication: enable this setting

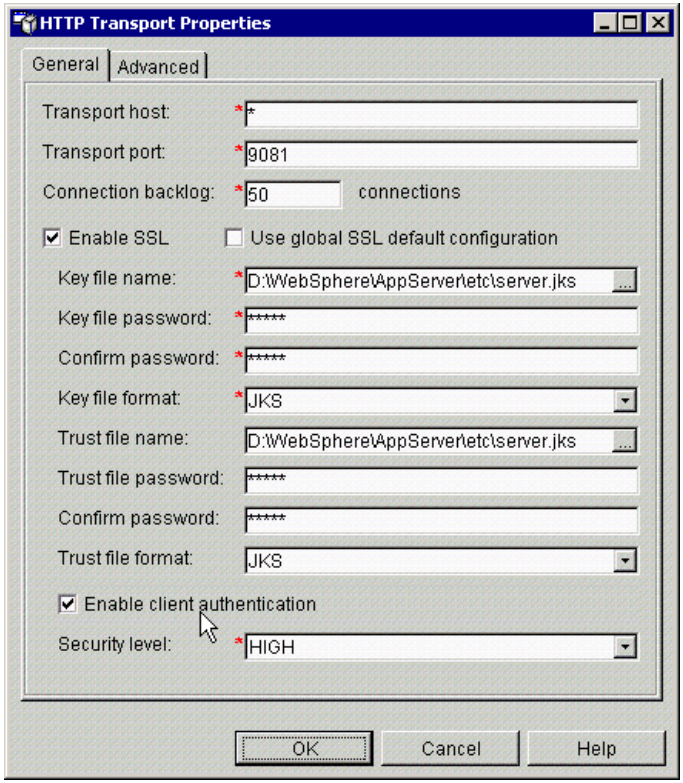


Figure 9-38 SSL settings for client authentication enabled mode

9.9.3 Update the Web server plug-in configuration

Regenerate the Web server plug-in file

1. Log in as a local administrator to the WebSphere server machine.
2. Run the WebSphere Administrative Console by issuing the following command:

   ```
   <WAS_HOME>\bin\adminclient.bat
   ```

3. Right-click the node <hostname> that contains the Default Server application server.
4. Choose the **Regen Webserver Plugin** command, as shown in Figure 9-27 on page 235.
5. Edit the generated <WAS_HOME>\config\plugin-cfg.xml file so that the transport element's keyring and stashfile properties reflect the path to the plug-in keystore database we created above.

```
<Server Name="Default Server">
   <Transport Hostname="localhost" Port="9081" Protocol="https">
      <Property name="keyring"
value="<plugin_install_path>/etc/client.kdb"/>
      <Property name="stashfile"
value="<plugin_install_path>/etc/client.sth"/>
   </Transport>
</Server>
```

6. If the Web server plug-in is located on a different machine from the WebSphere Application Server, copy the edited plugin-cfg.xml file to the <plugin_install_path>\config directory of the Web server machine.

Restart Web server

Restart the IBM HTTP Server by issuing the following commands on the Web server machine:

```
D:\> net stop "IBM HTTP Server"
D:\> net start "IBM HTTP Server"
```

9.10 Install WebSphere Application Server - silent mode

You can silently install WebSphere Application Server to a local machine. To do the silent installation:

1. Locate the file setup.iss on the product CD or, if you downloaded the product, among downloaded and extracted files. If the file is on a CD, copy the file to a directory on your machine. Then, jot down the full path name for setup.iss.
2. Open an editor on the file setup.iss and change the parameters in the file as appropriate for your choice of products and configuration. Descriptions for the parameters are given in Table 9-9 on page 258.
3. Uninstall any versions of WebSphere Application Server or any downlevel versions of IBM HTTP Server on your machine.
4. Ensure that all software prerequisites are met.
5. At a command line, enter the following:

   ```
   setup <path>\setup.iss -s
   ```

   Do not use the -f1 option of the **setup.iss** command.
6. After the WebSphere Application Server installation program runs, reboot your machine.

Should you encounter errors during installation, examine the setup log file wssetup.log.

Table 9-9 Parameters that you can set in setup.iss

Setup option	Parameter description
Language	To override the language specified by the system locale, set `Lang=` to a number that corresponds to a language such as English, Spanish, French, German, Italian, Japanese, Portuguese, Korean, Chinese, or Taiwanese. For example, for English, specify `Lang=0009`. See the comment for `Lang=` in setup.iss to determine the correct number for a language.
Rebooting after installation	Under `[RebootDialog-0]`, to have your system reboot after installation, set `Result=` to `1`. By default, your system will not reboot after installation completes (`Result=0`).

WebSphere Application Server directory	Under `[SelectTargetDir]`, specify the main directory for WebSphere Application Server (`szDir=`). The defaults are:
	`[SelectTargetDir]`
	`szDir=C:\WebSphere\AppServer`
	`Result=1`

Components selection	Under `[Components]`, set a component to `0` if you do not want the component installed. Keep the default of `component=1` to have the installation program install the component. The components include:
	Server Application and Administrative Server
	Tools Application and Development Tools
	Admin Administrator's Console
	Java IBM JDK
	Samples Samples
	The defaults are:
	`Server=1`
	`Tools=1`
	`Admin=1`
	`Java=1`
	`Samples=1`

Security	Under `[Security]`, specify a valid user ID (`szUser=`) and a valid password (`szPassword=`). Do not use the characters < or > in the user ID or password. Optionally, specify an administrative console host (`szHostName=`); leave the parameter value blank for a local host. The defaults are:
	`[Security]`
	`szUser=`
	`szPassword=`
	`szHostName=`

Setup option	Parameter description
Database	Under [Database], specify the following: szType= The database product. The valid values include: DB2, Informix, Merant, Oracle and Sybase. szName= The name of database to be used with WebSphere Application Server. The default for DB2 UDB, Informix, Merant and Sybase is was40. For Oracle, the default is orcl. szUser= Your database user ID. szPassword= The password for the user ID. szPath= The main directory for the database. For example, for DB2 UDB use D:\SQLLIB. szURL= The URL for the database. This parameter is required only for Oracle, which uses the value jdbc:oracle:thin:@<hostname>:1521:<szName>. szServer= The server for the database. This parameter is required for Informix, Merant, Oracle and Sybase. szPort= The port for the database. This parameter is required for Informix, Merant, Oracle and Sybase. The defaults resemble: [Database] szType=DB2 szName=was40 szUser=db2admin szPassword=db2admin szPath=c:\sqllib szURL= szServer= szPort=
IBM Distributed Debugger	For [OLT], specify 0 for Result= if you do not want the IBM Distributed Debugger installed. The debugger provides object level trace. The default is 1, which installs the debugger. The default install directory is C:\IBMDebug; to install to a different directory, specify szDir=<drive_letter>:\<different_directory>. The defaults are: [OLT] Result=1 szDir=

Setup option	Parameter description
Other JDK	Under [SelectJdkDir], you can specify an alternative JDK to the one that is shipped with WebSphere Application Server by designating the JDK home directory (szDir=). The defaults are: [SelectJdkDir] szDir=
Configuring IBM HTTP Server	For [SelectIBMHttpServer], if you want to install the plug-in for IBM HTTP Server, specify 1 for Result=. Result=0 does not install the plug-in. The installation program will automatically determine the location of the Web server so you do not have to specify the directory. However, you can override auto-detection of the Web server by specifying the path for the httpd.conf file (szDir=). The defaults are: [SelectIBMHttpServer] Result=0 szDir= If you want IBM HTTP Server installed, also change the values for [HTTPServer].
Configuring Apache HTTP Server	Under [SelectApache], if you want to install the plug-in for the Apache HTTP Server, specify 1 for Result=. Result=0 does not install the plug-in. The installation program will automatically determine the location of the Web server so you do not have to specify the directory. However, you can override auto-detection of the Web server by specifying the path for the httpd.conf file (szHttpDir=) and the srm.conf file (szSrmDir=). The defaults are: [SelectApache] Result=0 szHttpDir= szSrmDir=
Microsoft Internet Information Server (IIS) 4.0/5.0 plug-in	Under [SelectIIS45], if you will need the IIS 4.0/5.0 plug-in, specify 1 for Result=. Result=0 does not install the plug-in. The defaults are: [SelectIIS45] Result=0
Lotus Domino 5.0 plug-in	Under [SelectDominoDir], if you will need the Domino 5.0 plug-in, specify 1 for Result=. Result=0 does not install the plug-in. The installation program will automatically determine the location of the Web server so you do not have to specify the directory. However, you can override auto-detection of Web server by specifying the path for the httpd.cnf file (szDir=). The defaults are: [SelectDominoDir] Result=0 szDir=

Setup option	Parameter description
iPlanet 4.0 plug-in	Under [SelectiPlanet40_Dir], if you will need the iPlanet 4.0 plug-in, specify 1 for Result=. Result=0 does not install the plug-in. The installation program will automatically determine the location of the Web server so you do not have to specify the directory. However, you can override auto-detection of the Web server by specifying the path to the obj.conf file (szDir=). Set UseSSL= to 1 to enable SSL security; to keep SSL security disabled, use 0. Note that enabling SSL on a non-secure Web server will prevent the plug-in from working. The defaults are: [SelectiPlanet40_Dir] szDir= Result=0 UseSSL=0
IBM HTTP Server installation	Under [HTTPServer], if you want IBM HTTP Server installed, specify 1 for Result=. Also, specify the directory to which IBM HTTP Server should be installed (szDir=). You should not change the parameter for the installation image (szInstallDir=). The defaults are: [HTTPServer] Result=0 szDir= szInstallDir="HTTPD" If you want IBM HTTP Server installed, also change the values for [SelectIBMHttpServer] so the plug-in is installed.

10

AIX installation steps

This chapter provides detailed procedures for installing, configuring, and verifying a number of the scenarios described in Chapter 8, “Installation approach” on page 127, for an environment consisting of the following components:

- Operating system - AIX 4.3.3
- Database server - IBM DB2 UDB Enterprise Edition
- Web server - IBM HTTP Server
- Application server - WebSphere Application Server V4.0, Advanced Edition

The procedures described are intended to be used as working examples in conjunction with the product installation guides for all the possible values that may be unique within your runtime environment. This chapter is organized into the following sections:

- Planning
- Install AIX
- Install Web server
- Install database server
- Install database client
- Install WebSphere Application Server V4.0, Advanced Edition
- Install WebSphere plug-in on remote Web server

- Install the new WebSphere node into an existing domain
- Configure WebSphere HTTP transport for SSL
- Install WebSphere Application Server - silent mode

10.1 Planning

This section defines the hardware and software used within the AIX environment to test a number of different WebSphere Application Server V4.0 configuration scenarios.

10.1.1 Hardware and software prerequisites

WebSphere Application Server V4.0, Advanced Edition has the following hardware and software requirements.

Hardware

- 150 MB diskspace (minimum) for WebSphere Application Server
- 300 MB diskspace (minimum) for IBM DB2 Enterprise Edition
- 50 MB diskspace (minimum) for IBM HTTP Server

Software

- AIX 4.3.3.0 + prerequisites (described 10.2, “Install AIX” on page 272)
- IBM WebSphere Application Server V4.0, Advanced Edition
- IBM DB2 Universal Database V7.2 FP4, Enterprise Edition for AIX
- IBM HTTP Server 1.3.19
- IBM GSKit 5.0.3.52

Note: For further details on requirements, see the following documentation:

1. DB2 -

   http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winox2unix/support/v7pubs.d2w/en_main

2. WebSphere -

   http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

10.1.2 Software used in our test environment

We used the following software in our test environment:

- AIX 4.3.3 + prerequisites
- IBM WebSphere Application Server V4.0, Advanced Edition
- IBM DB2 Universal Database V7.2 FP4, Enterprise Edition for AIX
- IBM HTTP Server 1.3.19
- IBM GSKit 5.0.3.52 (included in IBM HTTP Server package)
- IBM JDK 1.3.0 (included in WebSphere Application Server package)

Note: Other Web servers and database software may be used, as documented in the Product Installation Guide.

Product installation roots

The variables listed in Table 10-1 are used frequently throughout this documentation to represent the root installation directories of the software components.

Table 10-1 Product installation roots

Variable	Default value	Component
<WAS_HOME>	/usr/WebSphere/AppServer	WebSphere
<plugin_install_path>	/usr/WebSphere/AppServer	WebSphere HTTP plug-in
<db2_install_path>	/home/db2inst1	DB2 server DB2 client
<http_server_install_path>	/usr/HTTPServer	IBM HTTP Server

Installation variables

The variable in Table 10-2 is used frequently throughout this documentation to represent settings used during the installation of software components.

Table 10-2 Common installation setting

Variable	Default value	Component
db2_instance_owner	db2inst1	DB2 server

10.1.3 Hardware used in our test environment

This section describes the hardware used within our test WebSphere Application Server V4.0 environment on AIX.

- Server 1
  - IBM RS/6000 43P Model 150

  - 1 433 MHz CPU
  - 512 MB RAM
  - 18 GB hard disk
  - 1 IBM 10/100 Mbps Ethernet PCI Adapter
- Server 2
  - IBM RS6000 43P Model 150
  - 1 433 MHz CPU
  - 768 MB RAM
  - 18 GB hard disk
  - 1 IBM 10/100 Mbps Ethernet PCI Adapter

10.1.4 Example scenarios

Within this test environment, we describe the tasks necessary to install and configure the following scenarios that are described in Chapter 8, “Installation approach” on page 127:

- Scenario A - basic one-server configuration (server 1), but using the WebSphere embedded Web server in place of the usual stand-alone Web server.
- Scenario B - basic one-server configuration (server 1), but using a stand-alone Web server and the WebSphere HTTP plug-in.
- Scenario D - basic two-server configuration, one server hosting WebSphere and the database server (server 1), the other hosting the Web server and WebSphere HTTP plug-in (server 2). Communication between plug-in and WebSphere is HTTP (unencrypted).
- Scenario E - extension of the basic two-server configuration of Scenario D, with the HTTP communication between the plug-in and WebSphere encrypted using SSL and client authentication mode disabled.
- Scenario F - extension of the secure two-server configuration of Scenario E, with the SSL client authentication mode enabled.
- Scenario G - extension of the secure two-server configuration of Scenario F, with the Web server configured to use SSL for encryption of all communication between itself and Web browsers.
- Scenario H - basic two-server configuration using a single WebSphere administrative domain database.

Installation and configuration tasks

Table 10-3 provides a detailed summary of the tasks and the order in which they must be performed that are necessary to install and configure each of the example scenarios described above.

To install and configure an example, perform the tasks listed for the example in the specified order. Only those tasks that are numbered (non-blank) are to be performed for a particular example. Detailed descriptions of each of these tasks can be found later in this chapter.

Table 10-3 Installation and configuration tasks broken down by scenario

Server	Task	Example scenario						
		A	B	D	E	F	G	H
1	Operating system: Installation of AIX operating system and any required patches: 1. Install AIX 2. Install any patches and filesets as detailed in the WebSphere platform-specific prerequisites See 10.2, “Install AIX” on page 272.	1	1	1	1	1	1	1
1	Web server: Installation and configuration of IBM HTTP Server: 1. Perform preinstallation tasks 2. Install IBM HTTP Server 3. Configure IBM HTTP Server 4. Verify IBM HTTP Server See 10.3, “Install Web server” on page 274.	-	2	-	-	-	-	2

Server	Task	Example scenario						
		A	B	D	E	F	G	H
1	Database server: Installation and configuration of IBM DB2 server necessary to support usage by WebSphere: 1. Perform preinstallation tasks 2. Install DB2 server 3. Verify DB2 server installation 4. Configure DB2 server 5. Set up WebSphere administrative database See 10.4, “Install the database server” on page 283.	2	3	2	2	2	2	3
1	WebSphere Application Server: Installation and configuration of WebSphere necessary to handle requests through the “embedded” Web server: 1. Perform preinstallation tasks 2. Install WebSphere 3. Verify WebSphere installation See 10.6, “Install WebSphere Application Server” on page 305.	3	4	3	3	3	3	4
1	WebSphere Application Server: Creation and configuration of an HTTPS transport on WebSphere default Web container: 1. Create new server certificate keystore 2. Create new server self-signed certificate used for SSL handshaking 3. Create new HTTPS transport for WebSphere Default Server See 10.9, “Configure the WebSphere HTTP transport for SSL” on page 323.	-	-	-	4	4	4	-

Server	Task	Example scenario						
		A	B	D	E	F	G	H
2	Web server: Installation and configuration of IBM HTTP Server: 1. Perform preinstallation tasks 2. Install IBM HTTP Server 3. Configure IBM HTTP Server 4. Verify IBM HTTP Server See 10.3, “Install Web server” on page 274.	-	-	4	5	5	5	5
2	Database client: Installation and configuration of IBM DB2 client necessary to support remote usage by WebSphere: 1. Perform preinstallation tasks 2. Install DB2 client 3. Verify DB2 client installation 4. Configure DB2 client 5. Set up access to remote WebSphere administrative database See 10.5, “Install the database client” on page 298.	-	-	-	-	-	-	6
2	WebSphere Application Server: Installation and configuration of WebSphere necessary to handle requests through the “embedded” Web server: 1. Perform preinstallation tasks 2. Install WebSphere 3. Verify WebSphere installation See 10.6, “Install WebSphere Application Server” on page 305.	-	-	-	-	-	-	7

Server	Task	Example scenario						
		A	B	D	E	F	G	H
2	WebSphere HTTP plug-in: Basic installation and configuration of WebSphere HTTP plug-in on remote Web server: 1. Perform preinstallation tasks 2. Install WebSphere plug-in 3. Verify WebSphere plug-in installation 4. Configure for remote WebSphere plug-in 5. Verify remote plug-in configuration See 10.7, “Install the WebSphere plug-in on a remote Web server” on page 315.	-	-	5	6	6	6	-
2	WebSphere HTTP plug-in: Configuration of plug-in necessary to communicate with HTTPS (client authentication disabled) transport on remote WebSphere: 1. Create new plug-in certificate keystore 2. Import WebSphere self-signed server certificate as trusted CA 3. Update Web server plug-in configuration file to support HTTPS transport and certificate keystore See 10.9, “Configure the WebSphere HTTP transport for SSL” on page 323.	-	-	-	7	7	7	-
2	WebSphere HTTP plug-in: Extra configuration of plug-in necessary to support HTTPS transport with client authentication enabled: 1. Create new self-signed client certificate used to authenticate plug-in to server during SSL handshaking See 10.9, “Configure the WebSphere HTTP transport for SSL” on page 323.	-	-	-	-	8	8	-

Server	Task	Example scenario						
		A	B	D	E	F	G	H
1	WebSphere Application Server: Extra configuration of WebSphere HTTPS transport necessary to support client authentication: 1. Import plug-in self-signed certificate as a trusted CA 2. Configure HTTPS transport to use client authentication See 10.9, “Configure the WebSphere HTTP transport for SSL” on page 323.	-	-	-	-	9	9	-
1	WebSphere Application Server: Startup the WebSphere Default Server ready to handle requests: 1. Regenerate Web server plug-in configuration file See 10.9, “Configure the WebSphere HTTP transport for SSL” on page 323.	-	-	-	8	10	10	-
1	Web server: Test that Web server and plug-in can access the current configuration of the WebSphere Default Server: 1. Restart Web server 2. Verify configuration – `http://<server1 hostname>/webapp/examples/showCfg`	-	5	-	-	-	-	8
2	Web server: Test that Web server and plug-in can access the current configuration of the WebSphere Default Server: 1. Restart Web server 2. Verify configuration – `http://<server2 hostname>/webapp/examples/showCfg`	-	-	6	9	11	11	9

Server	Task	Example scenario						
		A	B	D	E	F	G	H
1	Web server: Extra configuration of Web server to use SSL encryption for requests: 1. Stop Web server 2. Configure httpd.conf to support SSL for requests 3. Create new keystore 4. Create new self-signed certificate 5. Start Web server See 10.3.5, “Enable SSL encryption for requests (optional)” on page 281.	-	-	-	-	-	12	-
2	Web server: 1. Restart Web server 2. Verify configuration – `https://<host>/` – `https://<host>/webapp/examples/showCfg`	-	-	-	-	-	13	-

10.2 Install AIX

Prior to installing any of the WebSphere components, the proper level of the operating system must be installed.

- AIX 4.3.3 maintenance level
- AIX 4.3.3 additional filesets required by WebSphere V4.0
- Extra tools

10.2.1 AIX 4.3.3 maintenance level

The WebSphere V4.0 minimum supported level of AIX 4.3.3 is maintenance Release 6.

1. Determine whether the server currently has the required maintenance release (or newer) installed by issuing the following command:

```
instfix -ik 4330-06_AIX_ML
```

2. If the required maintenance release is installed, the command will generate the following output:

   ```
   All filesets for 4330-06_AIX_ML were found
   ```

3. If no output is generated, then you must upgrade AIX to the required maintenance release before continuing. For details on how to perform such an upgrade, please see the AIX product documentation.

Tip: The AIX maintenance releases can be obtained from the following IBM FTP site:

ftp://ftp.software.ibm.com/aix/fixes/v4/

10.2.2 AIX 4.3.3 additional filesets required by WebSphere V4.0

There are a number of prerequisite filesets of specific versions required for the installation of WebSphere Application Server V4.0. The required filesets and version numbers are listed in Table 10-4. You can check this list using the prereq.properties file supplied in the installation package.

Table 10-4 AIX filesets required by WebSphere V4.0

Fileset	Require this version or newer...
bos.rte.net	4.3.3.2
bos.rte.libc	4.3.3.55
bos.net.tcp.client	4.3.3.28
X11.base.rte	4.3.3.25
X11.motif.lib	4.3.3.26
X11.base.lib	4.3.3.26

1. Determine whether these filesets are currently installed by using the following command:

   ```
   lslpp -L | grep <fileset>
   ```

2. If the output does not include the required version of the fileset (or newer), then the fileset must be upgraded before continuing. For details on how to perform such an upgrade, please see the AIX product documentation.

Tip: AIX filesets can be obtained from the following IBM FTP site:

ftp://ftp.software.ibm.com/aix/fixes/v4/

At this location, you will need to select the particular directory to which the fileset belongs, for example X11 directory for the X-windows components.

10.2.3 Extra tools

Table 10-5 lists the utility that may be required during the install of the software components, which are not installed by default on AIX.

Table 10-5 Required utility program

Utility	Install under...	Permissions	Available from...
unzip	/usr/bin	rwx------	http://www.info-zip.org/pub/infozip/UnZip.html#AIX

10.3 Install Web server

This section provides detailed instructions for installing, configuring, and verifying IBM HTTP Server V1.3.19 for AIX.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install IBM HTTP Server.
3. Configure IBM HTTP Server.
4. Verify IBM HTTP Server.
5. Enable SSL encryption for requests (optional).

Note: Whether or not a particular task needs to be performed, and the order in which tasks must be performed, is identified during the planning stage and is dependent upon the needs of the particular topology or scenario being installed. See 10.1, “Planning” on page 264 for details on those tasks to be performed for each example.

10.3.1 Preinstallation tasks

Prior to installing IBM HTTP Server V1.3.19, the following checks and tasks must be completed on the IBM HTTP Server machine:

1. Check that IP ports are unused.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing services on the server that use the following IP ports:
   - 80 (standard HTTP port)
   - 443 (standard HTTPS port)
   - 8008 (IBM HTTP Server Administration port)

   We suggest using the following command for this task:

   ```
   # netstat -an | grep LISTEN
   ```

10.3.2 Install IBM HTTP Server

To install the IBM HTTP Server V1.3.19, complete the following steps on the Web server machine:

1. Log in as root.
2. Start a terminal session.
3. Insert the IBM WebSphere Application Server V4.0 CD-ROM into the CD-ROM drive. This media also contains the IBM HTTP Server.
4. Mount the CD-ROM:

   ```
   # mount -r -v cdrfs /dev/cd0 /mnt
   ```

5. Change to the ihs_128 subdirectory of the root installation directory on the CD.
6. Start the installation using the “smitty” tool:

   ```
   # smitty install_all
   ```

7. When the Install and Update from ALL Available Software window appears, enter the following in the INPUT device / directory for software field:

   ```
   ./
   ```

   Press Enter
8. In the Software to install field, press F4 for a list. Highlight the packages by pressing F7 to select each of the packages listed in Table 10-6.

Table 10-6 IBM HTTP Server - packages to install

Package	Description
gskkm	AIX certificate and SSL base runtime.

Package	Description
HTTP_server.admin	The server administrator used to configure the IBM HTTP Server.
HTTP_server.base	The IBM HTTP Server.
HTTP_server.frca	The Fast Response Cache Accelerator.
HTTP_server.html.*locale*	The IBM HTTP Server documentation (where *locale* is your country code - for example en_US for U.S. English).
HTTP_server.man.en_US	The IBM HTTP Server server manual. Note that this is available only in U.S. English.
HTTP_server.modules **Note**: read the description for this package	The modules for IBM HTTP Server. Select HTTP Server.Fast-CGI and HTTP Server.MT Module; if you simply select the entire module package your instance will not start. The module HTTP Server.LDAP is required only when using LDAP SSL integration with the IBM HTTP Server.
HTTP_server.msg.*locale*.admin	The message catalog (where *locale* is your country code). Choose en_US locale for U.S. English.
HTTP_server.msg.*locale*.ssl.core	The message catalog, where *locale* is the code page value for your locale). Choose en_US locale for U.S. English.
HTTP_server.ssl.128 or HTTP_server.ssl.56	Determines whether you will have 128-bit encryption or 56-bit encryption.
HTTP_server.ssl.core	Required in order to install SSL modules with encryption levels.

Important: Double-check that you have selected the correct packages. Due to the number of selections, it is easy to make a mistake that will result in the IBM HTTP Server not working properly.

9. Press Enter to close the list.
10. Press the Tab key to toggle the value of the following settings to Yes:
    - VERIFY install and check file sizes
    - DETAILED output

11. Press Enter.
12. When the `ARE YOU SURE?` message appears, press Enter.
13. The command status window appears. When the installation is complete, the Command field at the top of the window changes from Running to OK.
14. Scroll to the Installation Summary section at the bottom of the listing. In the Result column, you should see either SUCCESS or Already Installed next to the name of each component. If you do not, correct the problem.
15. Press F10 to return to the system prompt.
16. Unmount the CD-ROM:

```
# cd /
# umount /mnt
```

10.3.3 Configure the IBM HTTP Server

After the installation of IBM HTTP Server V1.3.19, the following configuration tasks must be completed on the IBM HTTP Server machine:

1. Create the IBM HTTP Server admin account.
2. Create the UNIX runtime account.
3. Update httpd.conf.
4. Restart the IBM HTTP Server.

Create the HTTP server admin account

The administration account is used to access the HTTP Administration Server Configuration GUI. To create the account, perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Change the directory to the <http_server_install_path>/bin directory.
4. Create the administration user by typing the following commands::

```
# ./htpasswd -m ../conf/admin.passwd admin
New password: <admin_password>
Re-type new password: <admin_password>
```

   Where `admin` is the IBM HTTP Server administration userid.

Create the UNIX runtime account

Although the IBM HTTP Server process is started under the root account, it should be configured to, then switched to, run under another account. A UNIX account needs to be created specifically for this purpose.

Perform the following steps to create the UNIX account and configure the IBM HTTP Server:

1. Log in as root.
2. Start a terminal session.
3. Change the directory to the <http_server_install_path>/bin directory.
4. Run the setupadm script:

   ```
   # ./setupadm
   ```

5. Answer the prompts as follows:
   a. Please supply a User ID to run the Administration Server. For example:

      User ID: `www`

      Press Enter.

   b. Please supply a GROUP NAME to run the Administration Server. For example:

      Group Name: `www`

      Press Enter.

   c. Please supply the Directory containing the files for which a change in the permissions is necessary. The default is /usr/HTTPServer/conf.

      Press Enter to accept the default.

   d. To perform the change, enter 1. To quit with no changes, enter 2 (default).

      Enter 1 to perform changes.

      Press Enter.

   e. Configuration file: '/usr/HTTPServer/conf/admin.conf' will be saved. Do you wish to update the Administration Server Configuration file. To perform the change, enter 1. To exit with no change, enter 2 (default).

      Enter 1 to update configuration file.

      Press Enter.

   f. Do you wish to run Admin Server and IHS Server in Language other than English? For a language other than English, enter 1. For English, enter 2 (default).

      Press Enter to accept the default (English).

6. The setupadm program returns to a system prompt.

Update httpd.conf

The IBM HTTP Server configuration file httpd.conf must be updated to reflect the following:

- The fully qualified host name of the server.
- The UNIX account to run under.

To update the IBM HTTP Server configuration file, complete the following steps:

1. Edit the <http_server_install_path>/conf/httpd.conf file and update the settings listed in Table 10-7.

Table 10-7 httpd.conf required settings

Setting	Required value...
User	www
Group	www
ServerName	<hostname.domain.com>

2. Save the changes and exit.

Restart the IBM HTTP Server

The IBM HTTP Server must be restarted in order for the above configuration changes to take effect:

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <http_server_install_path>/bin
# ./apachectl restart
```

10.3.4 Verify the IBM HTTP Server

In order to verify the IBM HTTP Server V1.3.19 installation, perform the following checks on the IBM HTTP Server machine:

1. Check the process status.
2. Check request handling.

Check the process status

To check IBM HTTP Server process status, perform the following steps:

1. Check that the HTTP Server processes are running by issuing the following command:

   ```
   # ps -ef | grep httpd
   ```

   The output should list a number of processes.

2. Check that the HTTP Server is registered to listen on port 80 and is therefore ready to handle requests:

```
# netstat -an | grep LISTEN | grep 80
```

Check request handling

To check IBM HTTP Server request handling, perform the following steps:

1. Using a Web browser, request the following URL representing the IBM HTTP Server home page:

   ```
   http://<hostname.domain.com>/
   ```

   The window shown in Figure 10-1 will be displayed if the IBM HTTP Server has been installed and configured correctly.

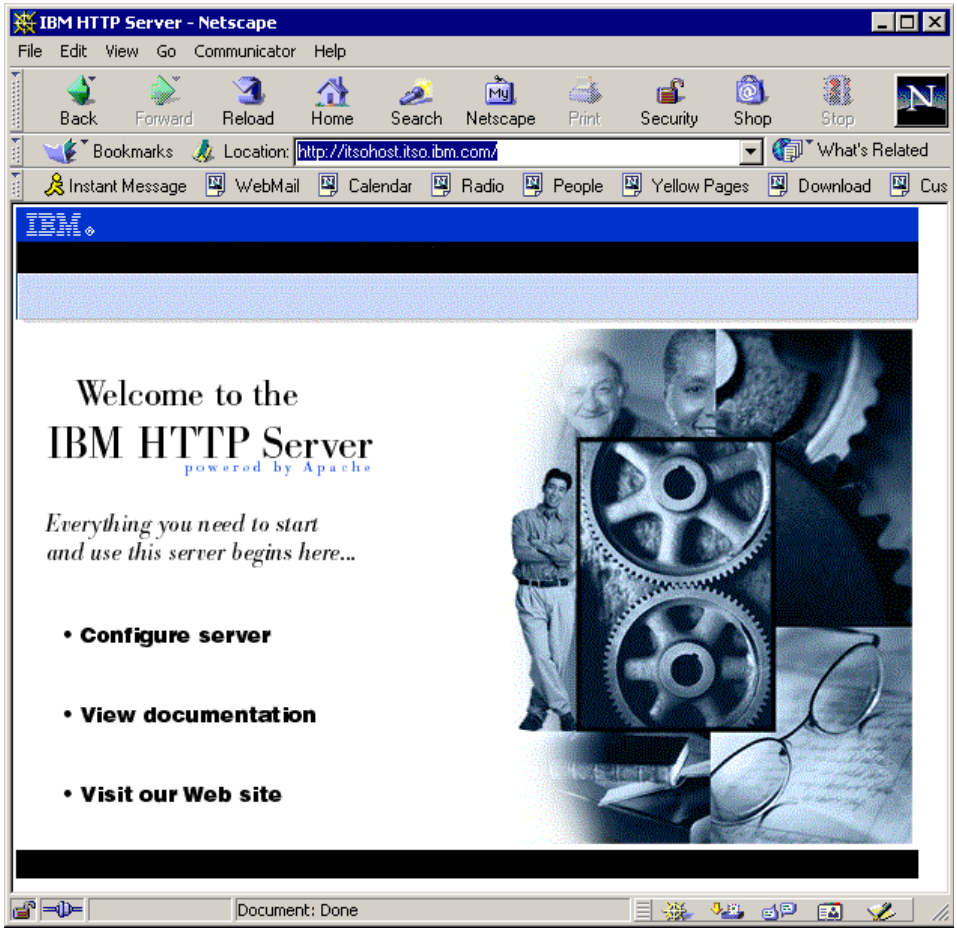


Figure 10-1 Home page request handled by Web server

10.3.5 Enable SSL encryption for requests (optional)

In this section we provide detailed instructions for creating a certificate, installing the certificate and configuring an IBM HTTP Server for SSL. In our example, we will create a self-signed test certificate. For a production environment you will need to request a real certificate from a certificate authority, such as VeriSign.

Enabling SSL for communication between the IBM HTTP Server and a Web browser is a multistep process:

1. Stop the IBM HTTP Server process.
2. Configure httpd.conf to add SSL support.
3. Create a new keystore (certificate trust) database file.
4. Create a new self-signed certificate.
5. Start the IBM HTTP Server process.
6. Verify the SSL configuration.

Stop the IBM HTTP Server process

To stop the IBM HTTP Server process, perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <http_server_install_path>/bin
# ./apachectl stop
```

Configure httpd.conf to add SSL support

To configure SSL for the IBM HTTP Server, complete the following steps:

1. Use the sample httpd.conf.sample file, which includes SSL entries, as a starting point to enable SSL for the IBM HTTP Server.
   a. Change to the <http_server_install_path>/conf directory.
   b. Back up the existing httpd.conf file by renaming it to httpd.conf.bak.
   c. Rename httpd.conf.sample to httpd.conf.
2. Modify the httpd.conf file using a text editor. Ensure that the following lines are uncommented by removing the # symbol:

   ```
   LoadModule ibm_ssl_module libexec/mod_ibm_ssl_<encrypt_level>.so
   ```

   Where <encrypt_level> is the appropriate encryption level for your locale. For example, mod_ibm_ssl_128.so for 128-bit encryption in the US and Canada.

```
AddModule mod_ibm_ssl.c
Listen 443
<VirtualHost hostname.domain.com:443>
```

You must substitute your fully qualified host name in this line, for example `<VirtualHost itsohost.itso.ibm.com:443>`.

```
SSLEnable
</VirtualHost>
SSLDisable
Keyfile "<http_server_install_path>/ssl/webclient.kdb"
SSLV2Timeout 100
SSLV3Timeout 1000
```

The value of the KeyFile parameter is the absolute path to the CMS format keystore database file of your choosing. In this example we assume that a webclient.kdb file is created in the ssl subdirectory of the IBM HTTP Server installation.

3. Ensure the following settings have been disabled by adding the # symbol to the start of each line:

```
#AfpaEnable
#AfpaCache on
#AfpaLogFile <log_file_path>
```

The above AFPA options must be disabled in order for SSL encryption mode to operate correctly.

4. Save the changes.

Create a new keystore (trust certificate) database

To create a new SSL keystore database, use the method described in “Create a new keystore (certificate trust) database” on page 199. The method used is the same for all platforms, except for the following:

1. To run the Web server IBM Key Management Utility on AIX, issue the following commands:

```
# JAVA_HOME="/usr/jdk_base" ; export JAVA_HOME
# cd /usr/bin
# ./gsk5ikm
```

Create a new self-signed certificate

To create a new self-signed certificate, use the method described in “Create new self-signed certificate” on page 200. The method used is the same for all platforms, except for the following:

1. To run the Web server IBM Key Management Utility on AIX, issue the following commands:

```
# JAVA_HOME="/usr/jdk_base" ; export JAVA_HOME
# cd /usr/bin
# ./gsk5ikm
```

Start the IBM HTTP Server process

To start the IBM HTTP Server process, perform the following steps:

1. Issue the following commands:

```
# cd <http_server_install_path>/bin
# ./apachectl start
```

Verify the SSL configuration

To verify the SSL configuration of the IBM HTTP Server, use the method described in “Verify SSL configuration” on page 202. The method used is the same for all platforms.

10.4 Install the database server

This section provides detailed instructions for installing, configuring, and verifying IBM DB2 Universal Database, Enterprise Edition for AIX.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install the DB2 server.
3. Verify the DB2 server installation.
4. Configure the DB2 server.
5. Set up the WebSphere administrative database.

10.4.1 Preinstallation tasks

Prior to installing IBM DB2 Universal Database, Enterprise Edition for AIX, the following checks and tasks need to be completed:

1. Check the paging space.
2. Check that IP ports are unused.
3. Create the volume set for the DB2 instance.
4. Check the filesystem free space.

Check the paging space

There must be a minimum of 128 MB of paging space. By default, the AIX configuration assistant will suggest that you create a paging file twice the size of the RAM in your system.

1. Log in as root.
2. Start a terminal session.
3. Display paging space:

   ```
   # lsps -a
   ```

4. If the total active paging space is not greater than 128 MB, increase the paging space.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing active services that use the following IP ports on the server:
   - 523 (DB2 server)
   - 50000 (DB2 instance connection port)
   - 50001 (DB2 instance interrupt port)
   - 50002 (DB2 Control Server)

   We suggest using the following command for this task:

   ```
   # netstat -an | grep LISTEN
   ```

Create the volume set for the DB2 instance

We recommend that you create a separate file system for the DB2 instance. This will provide the instance owner with more control and the ability to set the storage size allocated to the databases.

Note: The default database location for DB2 is the instance owner's home directory.

Perform the following steps to create a new logical volume and file system for this purpose:

1. Create a logical volume.
2. Create the journal file system - mount point.
3. Allocate storage for the DB2 instance.
4. Set the file system owner to the DB2 instance owner.

Create a logical volume

This procedure provides instructions for creating a logical volume on AIX.

1. Log in as root.
2. Start a terminal session.
3. Use the smitty utility to access the AIX volume configuration:

   ```
   # smitty storage
   ```

4. Select `Logical Volume Manager` and press Enter.
5. Select `Logical Volumes` and press Enter.
6. Select `Add a Logical Volume` and press Enter.
7. Select `VOLUME Group name:` and press F4 for a listing:
   a. Select `rootvg` and press Enter.
8. When a Logical Volume window appears, enter the following fields:
   – Logical Volume NAME: dblv1
   – Number of LOGICAL PARTITIONS: `1`
9. Press Enter and wait until the status changes to Command: OK.
10. Press the F10 key to return to the system prompt.

Create the journal file system - mount point

Create a file system with the mount point as the home directory of the DB2 instance owner. The DB2 instance owner will be created in subsequent steps.

This procedure provides instructions for creating a logical volume on AIX.

1. Use the smitty utility to access the AIX volume configuration:

   ```
   # smitty storage
   ```

2. Select **File Systems** and press Enter.
3. Select **Add -> Change -> Show -> Delete File Systems** and press Enter.
4. Select **Journal File Systems** and press Enter.
5. Select **Add a Journaled File System on a Previously Defined Logical Volume** and press Enter.
6. Select **Add a Standard Journaled File System** and press Enter.
7. When the Add a Standard Journaled File System window appears, enter the following:
   – LOGICAL VOLUME name: press F4 for listing, select **dblv1**
   – MOUNT POINT: `/home/<db2_instance_owner>`

Important: Although any valid account name could be used for <db2_instance_owner>, DB2 convention specifies that the instance owner be called db2inst<x>, where x is the instance number.

In our test environment, and throughout this chapter, an instance name of db2inst1 is used.

- Mount AUTOMATICALLY at system restart: `yes`

Tip: Press the Tab key to toggle between yes and no.

8. Press Enter and wait until the status changes to Command: OK.
9. Press the F10 key to return to the system prompt.

Allocate storage for the DB2 instance

Mount the file system prior to allocating storage to the newly created file system. By default, AIX allocates storage in 512 byte blocks. In our test environment we will create a file system that is 600,000 blocks (300 MB).

1. Mount the /home/<db2_instance_owner> file system:

```
# mount /dev/dblv1 /home/<db2_instance_owner>
```

2. Allocate file system storage space to the journal file system.

```
# chfs -a size='600000' /home/<db2_instance_owner>
```

Set the file system owner to the DB2 instance owner

The DB2 installer program can be used to create the DB2 instance owner and group, but it will not set the home directory ownership if the home directory already exists. We need to create the instance owner and group so we can manually set the home directory ownership before installing DB2.

1. Logged in as root, use the smitty utility to create the DB2 instance owner group (db2iadm1):

```
# smitty mkgroup
```

2. When the Add a Group window appears, enter the following:
   - Group NAME: `db2iadm1`
   - ADMINISTRATIVE group? `true`
   - Group ID: `201`
   - USER list: `root`
   - ADMINISTRATOR list: `<leave blank>`

3. Press Enter and wait until the status changes to Command: OK.
4. Press the F10 key to return to the system prompt.
5. Use the smitty utility to create the DB2 instance owner user:

   ```
   # smitty mkuser
   ```

6. When the Add a User window appears, enter the required details. We used the following settings:
   - User NAME: <db2_instance_owner>
   - User ID: 201
   - ADMINISTRATIVE USER? false
   - Primary GROUP: db2iadm1
   - Group SET: db2iadm1,staff
   - ADMINISTRATIVE GROUPS: <leave blank>
   - ROLES: <leave blank>
   - Another user can SU TO USER?: true
   - SU GROUPS: ALL
   - HOME directory: /home/<db2_instance_owner>
   - Initial PROGRAM: /usr/bin/ksh

   We accepted the default entry for the remaining fields.
7. Press Enter and wait until the status changes to Command: OK.
8. Press the F10 key to return to the system prompt.
9. Set the owner of /home/<db2_instance_owner> to the DB2 instance owner user:

   ```
   # chown -R <db2_instance_owner>:db2iadm1 /home/<db2_instance_owner>
   ```

10. Set the file permissions of /home/<db2_instance_owner>:

    ```
    # chmod -R 755 /home/<db2_instance_owner>
    ```

11. Set an initial password for the DB2 instance owner user:

    ```
    # passwd <db2_instance_owner>
    Changing password for "<db2_instance_owner>"
    <db2_instance_owner>'s New password:
    Enter the new password again:
    ```

This password has to be changed the first time the user logs in.

12. Log in as the DB2 instance owner user and change the password:

    ```
    # su - <db2_instance_owner>
    $ passwd
    ```

```
Changing password for "<db2_instance_owner>"
<db2_instance_owner>'s Old password:
<db2_instance_owner>'s New password:
Enter the new password again:
$ exit
```

Note: The password used must meet DB2 requirements: 8 characters or less and not containing the characters “<“ or “>”.

Check the filesystem free space

The filesystems listed in Table 10-8 need to be checked to ensure you have sufficient free space available to install DB2 and WebSphere.

Table 10-8 Filesystem free space requirements

Filesystem mount point	Recommended blocks of free space...
/	100,000
/home	100,000
/home/<db2_instance_owner>	600,000
/usr	1,000,000

Note: The filesystems are used for the following purposes:

/ - The / (root) directory may be used for temporary files during the installation.

/usr - The /usr directory contains the DB2 application files, as compared to the DB2 instance stored under /home/<db2_instance_owner>.

1. Before increasing a filesystem’s size, view the existing free space by using the **df** command.

   ```
   # df
   ```

2. Calculate the size required:

   ```
   new_size = current_size + (desired_free_space - free_space)
   ```

3. Increase the size:

   ```
   # chfs -a size='<new_size>' /<file_system_mount_point>
   ```

10.4.2 Install the DB2 server

In order to install IBM DB2 Universal Database, Enterprise Edition for AIX, perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Load the IBM DB2 Universal Database, Enterprise Edition for AIX CD-ROM into the CD-ROM drive and mount the CD.

   ```
   # mount -r -v cdrfs /dev/cd0 /mnt
   ```

4. Start the DB2 installer program:

   ```
   # cd /mnt
   # ./db2setup
   ```

5. In the Install DB2 window, select only the following option:
   - DB2 UDB Enterprise Edition

Navigation Tips:

- Press Tab to move between available options and fields.
- Highlight and press Enter to select an option.

6. Highlight **Customize...** for the DB2 Product Library option and press Enter.
7. In the DB2 Product Library window, highlight the appropriate option for your locale under the DB2 Product Library (HTML) section, then highlight **OK** and press Enter.
8. Highlight **OK** and press Enter.
9. In the Create DB2 Services window, select the **Create a DB2 Instance** option, highlight **OK** and press Enter.
10. In the DB2 instance authentication window, enter the following:
    - User Name: `<db2_instance_owner>`
    - User ID: `<use default UID>`
    - Group Name: `db2iadm1`
    - Group ID: `<use default GID>`
    - Home Directory: `/home/<db2_instance_owner>`
    - Password: `<leave blank>`
    - Verify Password: `<leave blank>`

    Leave the Password and Verify Password fields blank because the db2_instance_owner is an existing user.
11. Highlight **OK** and press Enter.
12. In the `Fence User` window, enter the following:

- User Name: `db2fenc1`
- User ID: `<use default UID>`
- Group Name: `db2fadm1`
- Group ID: `<use default GID>`
- Home Directory: `/home/db2fenc1`
- Password: `<db2fenc1_password>`
- Verify password: `<db2fenc1_password>`

Important: The DB2 installer uses the above information to automatically perform the following operations:

- Creates a group db2fadm1.
- Creates a user db2fenc1 with primary group db2fadm1.
- Sets `db2fenc1` password to `<db2fenc1_password>` value. The password used must meet DB2 requirements: eight characters or less and not containing the characters “<“ or “>”.
- Changes the ownership (owner:group) of the /home/db2fenc1 directory to be db2fenc1:db2fadm1.

13. Highlight **OK,** and press Enter.
14. In the DB2 Warehouse Control Database window, select the **Do not set up DB2 Warehouse Control Database** option, then highlight **OK,** and press Enter.
15. In the `Create DB2 Services` window, highlight the **Create Administration Server** option, then enter the following:
    - User Name: `db2as`
    - User ID: `<use default UID>`
    - Group Name: `db2asgrp`
    - Group ID: `<use default GID>`
    - Home Directory: `/home/db2as`
    - Password: `<db2asgrp_password>`
    - Verify password: `<db2asgrp_password>`

Important: The DB2 installer uses the above information to automatically perform the following operations:

- Creates a group db2asgrp.
- Creates a user db2as with primary group db2asgrp.
- Sets db2as password to <db2as_password> value. The password used must meet DB2 requirements: eight characters or less and not containing the characters “<“ or “>”.
- Changes the ownership (owner:group) of the /home/db2as directory to be db2as:db2asgrp.

16. Highlight **OK** and press Enter.
17. A message window appears indicating that DB2SYSTEM will be set to <hostname>. Highlight **OK** and press Enter.
18. Back in the Create DB2 Services window, highlight **OK** and press Enter.
19. The Summary Report window is displayed, listing the product components to be installed. Highlight **Continue** and press Enter.
20. A warning window appears indicating this is your last chance to stop. Highlight **OK** and press Enter.
21. The db2setup program installs the selected components. Depending on the speed of your processor, this can take up to 15 minutes.
22. You may be prompted to register the product. Complete the registration, then exit back to the install window.
23. When the install completes, a notice window informs you whether the installation was successful. Highlight **OK** and press Enter.
24. Scan the Status Report to ensure that all components were installed successfully. Highlight **OK** and press Enter.
25. In the DB2 Installer window, highlight **Close** and press Enter.
26. A window appears asking `Do you want to exit the DB2 Installer?` Highlight **OK** and press Enter.
27. The DB2 installation is now complete.
28. Unmount the CD-ROM:

```
# cd /
# umount /mnt
```

10.4.3 Verify the DB2 server installation

To verify the DB2 server installation, complete the following tasks:

1. Check the home directory permissions.
2. Check the DB2 instance owner profile.
3. Check the DB2 instance symbolic links.
4. Check the DB2 release level.
5. Check the DB2 service name.
6. Check the database manager configuration.

Check the home directory permissions

Check that the home directory ownerships have been correctly set up by the db2setup program, as listed in Table 10-9.

Table 10-9 DB2 home directory required permissions

Home directory path	Owner	Group	Permissions
/home/<db2_instance_owner>	<db2_instance_owner>	db2iadm1	drwxr-sr-x
/home/db2fenc1	db2fenc1	db2fadm1	drwxr-xr-x
/home/db2as	db2as	db2asgrp	drwxr-xr-x

Check the DB2 instance owner profile

The DB2 server installation should set up the .profile environment file of the <db2_instance_owner> so that the DB2 environment is set up when the user logs in.

1. The following content should have been added to .profile:

```
if [ -f /home/<db2_instance_owner>/sqllib/db2profile ]; then
   . /home/<db2_instance_owner>/sqllib/db2profile
fi
```

2. If not present, manually edit the <db2_instance_user> .profile file to add the above content.

Check the DB2 instance symbolic links

The DB2 server installation automatically creates a DB2 instance <db2_instance_owner> under the /home/<db2_instance_owner> directory. As part of the instance creation, db2setup should create symbolic links in the /home/<db2_instance_owner>/sqllib directory to files under /usr/lpp/db2_07_01.

Perform the following steps to check whether the symbolic links have been created:

1. Log in as root.
2. Start a terminal session.
3. Change the directory to /home/<db2_instance_owner>/sqllib.
4. Check whether a number of symbolic links exist pointing to files under /usr/lpp/db2_07_01.

   ```
   # ls -l
   ```

5. If not, issue the following commands:

   ```
   # cd /usr/lpp/db2_07_01/cfg
   # ./db2ln
   ```

Check the DB2 release level

Check that DB2 has the correct internal release level to meet WebSphere requirements:

1. Change to the user <db2_instance_owner>:

   ```
   # su - <db2_instace_owner>
   ```

2. Enter the following command:

   ```
   $ db2level
   ```

   This should generate output similar to the following:

   ```
   DB21085I  Instance "db2inst1" uses DB2 code release "SQL07021" with level
   identifier "03020105" and informational tokens "DB2 v7.1.0.43", "s010504"
   and "U475375a".
   ```

3. An internal release level of 7.1.0.43 should be indicated.

Check the DB2 service name

Check the service name recorded in the TCP/IP services file:

1. Open the /etc/services file and locate the entries that have comments referring to the DB2 instance connection port.
2. Locate the service name in the first column that corresponds to the lower port number. For example, if the following services were displayed:

   ```
   db2cdb2inst1 50000/tcp # Connection port for DB2 instance db2inst1
   db2idb2inst1 50001/tcp # Interrupt port for DB2 instance db2inst1
   ```

3. Record the `db2cdb2inst1` service name for later use.

Check the database manager configuration

Check that the service name is recorded in the database manager configuration:

1. Change to the user <db2_instance_owner>:

   ```
   # su - <db2_instance_owner>
   ```

2. Enter the following command:

   ```
   $ db2 get dbm cfg | grep SVCENAME
   ```

3. Verify that the SVCENAME value matches the service name recorded above from the services file. For example, something similar to the following should be displayed:

   ```
   TCP/IP Service name (SVCENAME)=db2cdb2inst1
   ```

4. If the value does not match, update the database manager configuration using the following commands:

   ```
   $ db2 update dbm cfg using svcename <service_name>
   $ db2stop
   $ db2start
   ```

   Where <service_name> must be replaced with the service name.

10.4.4 Configure the DB2 server

After the DB2 server installation, a number of configuration tasks must be performed so that WebSphere is able to use it as the repository for its administrative database:

1. Update the root administrative groups.
2. Update the JDBC level.
3. Configure the TCP/IP communication mode.
4. Verify the DB2 environment.
5. Update the root environment file.

Update the root administrative groups

The DB2 server installation should add the following administrative groups to the root user:

- db2asgrp

Perform the following steps to check whether the root account's administrative groups have been amended:

1. Log in as root.
2. Start a terminal session.
3. Use the smitty tool to update the root account configuration:

   a. Issue the following command:

```
# smitty chuser
```

b. In the User NAME field, type `root` and press Enter.

c. In the ADMINISTRATIVE GROUPS field, add `db2asgrp` to the list of groups, then press Enter.

d. When the process is complete, press F10 to exit smitty.

Update the JDBC level

IBM WebSphere Application Server V4.0 requires the use of JDBC2.0, whereas the default installation of IBM DB2 uses JDBC1.2. To update the DB2 JDBC level, complete the following steps:

1. Change the user to <db2_instance_owner>:

```
# su - <db2_instance_owner>
```

2. Add the following content to the end of the <db2_instance_owner> .profile environment file:

```
if [ -f /home/<db2_instance_owner>/sqllib/java12/usejdbc2 ] ; then
   . /home/<db2_instance_owner>/sqllib/java12/usejdbc2
fi
```

Configure the TCP/IP communication mode

The DB2 server may need to be reconfigured to use TCP/IP as its primary communication method:

1. Change the user to <db2_instance_owner>:

```
# su - <db2_instance_owner>
```

2. Check whether TCP/IP is the current DB2 communication method. The following command should return a value of tcpip:

```
$ db2set DB2COMM
```

3. If not TCP/IP, reset the DB2COMM DB2 environment variable:

```
$ db2set DB2COMM=tcpip
```

Verify the DB2 environment

After the above configuration steps, we need to check that the environment being set up by the db2profile and usejdbc2 scripts is correct:

1. Change to the user <db2_instance_owner>:

```
# su - <db2_instance_owner>
```

2. Issue the following command:

```
# set | grep [Dd][Bb]2
```

3. Check that the environment variables in this output match the values in Table 10-10.

Table 10-10 DB2 server required environment variables

Environment variable	Required value
DB2DIR	/usr/lpp/db2_07_01
DB2INSTANCE	<db2_instance_owner>
INSTHOME	/home/<db2_instance_owner>
LD_LIBRARY_PATH	:/home/<db2_instance_owner>/sqllib/lib
LIBPATH	.....:/home/<db2_instance_owner>/sqllib/java12:/home/<db2_instance_owner>/sqllib/lib
CLASSPATH	/home/<db2_instance_owner>/sqllib/function:/home/<db2_instance_owner>/sqllib/java12/db2java.zip:/home/<db2_instance_owner>/sqllib/java/runtime.zip
PATH	.....:/home/<db2_instance_owner>/sqllib/java12:/home/<db2_instance_owner>/sqllib/bin:/home/<db2_instance_owner>/sqllib/adm:/home/<db2_instance_owner>/sqllib/misc

Update the root environment file

The WebSphere Application Server will be run under root and will require access to the DB2 environment so that it can access the WebSphere administrative database. This requires that the root account's environment .profile be edited to add the following content at the end of the file.

```
# Set up DB2 environment for root user.
if [ -f /home/<db2_instance_owner>/sqllib/db2profile ] ; then
   . /home/<db2_instance_owner>/sqllib/db2profile
fi

# Force DB2 to use JDBC 2.0.
if [ -f /home/<db2_instance_owner>/sqllib/java12/usejdbc2 ] ; then
   . /home/<db2_instance_owner>/sqllib/java12/usejdbc2
fi
```

10.4.5 Set up the WebSphere administrative database

Next, set up a database in DB2 to use as the WebSphere administrative repository. The database will be populated with WebSphere schema and default values in a later task.

To set up the WebSphere database, complete the following steps:

1. Log in as <db2_instance_owner>.
2. Create the WebSphere database and configure its heap size to suit WebSphere requirements:

```
$ db2 create db was1
$ db2 update db cfg for was1 using applheapsz 256
```

> **Note:** Although the repository database created here is called “was1”, any valid DB2 database name can be used.

3. Check that the new database is known to DB2:

```
$ db2 list db directory
```

This should give output containing the following:

```
Database 1 entry:
 Database alias                 = WAS1
 Database name                  = WAS1
 Database drive                 = /home/db2inst1
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Indirect
 Catalog node number            = 0
```

4. In order to access the administrative database via TCP/IP, the DB2 node must first be cataloged:

```
$ db2 catalog tcpip node <node_name> remote <local_hostname> server
<service_name>
```

> **Important:** The <service_name> used to catalog the node must be the same as the database instance connection port name in the /etc/services file. The <node_name> chosen can be any valid DB2 node name.

5. The administrative database must now be cataloged as part of this TCP/IP node:

```
$ db2 catalog db was1 as was at node <node_hostname>
```

6. Check that the database TCP/IP alias is known to DB2:

```
$ db2 list db directory
```

This should give output containing the following:

```
Database 2 entry:
 Database alias                 = WAS
 Database name                  = WAS1
 Node name                      = <node_name>
 Database release level         = 9.00
 Comment                        =
```

```
 Directory entry type            = Remote
 Catalog node number             = -1
```

7. Verify connection to the local database via TCP/IP:

```
$ db2 connect to was user <db2_instance_owner> using <db2owner_passwd>
$ db2 disconnect current
```

   where <db2owner_passwd> is the DB2 instance owner password on the local DB2 server.

Tip: When the DB2 administration account is used to create a new database, it is automatically granted dba access rights. You only need to specifically grant access to another user if you plan to access the database from an account other than <db2_instance_owner>.

10.5 Install the database client

This section provides detailed instructions for installing, configuring, and verifying IBM DB2 Client for AIX.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install the DB2 client.
3. Verify the DB2 client installation.
4. Configure the DB2 client.
5. Set up access to the remote WebSphere administrative database.

10.5.1 Preinstallation tasks

Prior to installing the IBM DB2 client, the following checks and tasks need to be completed:

1. Check the paging space.
2. Check the filesystem free space.

Check the paging space

There must be a minimum of 128 MB of paging space. By default, the AIX configuration assistant will suggest that you create a paging file twice the size of the RAM in your system.

1. Log in as root.
2. Start a terminal session.

3. Display the paging space:

```
# lsps -a
```

4. If the total active paging space is not greater than 128 MB, increase the paging space.

Check the filesystem free space

The filesystems listed in Table 10-11 need to be checked to ensure you have sufficient free space available to install DB2 and WebSphere.

Table 10-11 Filesystem free space requirements

Filesystem mount point	Recommended blocks of free space...
/	100,000
/home	150,000
/usr	1,000,000

1. Before increasing a filesystem's size, view the existing free space by using the **df** command.

```
# df
```

2. Calculate the size required:

```
new_size = current_size + (desired_free_space - free_space)
```

3. Increase the size:

```
# chfs -a size='<new_size>' /<file_system_mount_point>
```

10.5.2 Install the DB2 client

In order to install the DB2 client for AIX, perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Load the IBM DB2 Universal Database, Enterprise Edition for AIX CD-ROM into the CD-ROM drive and mount the CD.

```
# mount -r -v cdrfs /dev/cd0 /mnt
```

4. Start the DB2 installer program:

```
# cd /mnt
# ./db2setup
```

5. In the Install DB2 window, select only the following option, then highlight **OK** and press Enter.

- DB2 Administration Client

6. In the Create DB2 Services window, select the **Create a DB2 Instance** option, highlight **OK** and press Enter.
7. In the DB2 instance authentication window, enter the following:
   - User Name: `<db2_instance_owner>`

     In the test environment we used a <db_instance_owner> of db2inst1.
   - User ID: `<use default UID>`
   - Group Name: `db2iadm1`
   - Group ID: `<use default GID>`
   - Home Directory: `/home/<db2_instance_owner>`
   - Password: `<user_password>`
   - Verify Password: `<user_password>`

   Important: The DB2 installer uses the above information to automatically perform the following operations:

   - Creates a group `db2iadm1`
   - Creates a user <db2_instance_owner> with primary group db2iadm1
   - Sets <db2_instance_owner> password to <user_password> value. The password used must meet DB2 requirements: eight characters or less and not containing the characters “<“ or “>”.
   - Changes the ownership (owner:group) of the /home/<db2_instance_owner> directory to be <db2_instance_owner>:db2iadm1
8. Highlight **OK** and press Enter.
9. Highlight **OK** and press Enter.
10. The Summary Report window is displayed, listing the product components to be installed. Highlight **Continue** and press Enter.
11. A window appears indicating that this is your last chance to stop. highlight **OK** and press Enter.
12. The db2setup program installs the selected components. Depending on the speed of your processor, this can take up to 15 minutes.
13. When the install completes, a notice window informs you whether the installation was successful. Highlight **OK** and press Enter.
14. Scan the Status Report to ensure that all components were installed successfully. Highlight **OK** and press Enter.

15. In the DB2 Installer window, highlight **Close** and press Enter.
16. A window appears asking `Do you want to exit the DB2 Installer?` Highlight **OK** and press Enter.
17. The DB2 installation is now complete.
18. Unmount the CD-ROM:

```
# cd /
# umount /mnt
```

10.5.3 Verify the DB2 client installation

To verify the DB2 client installation, complete the following tasks:

1. Check the home directory permissions.
2. Check the DB2 instance owner profile.
3. Check the DB2 instance symbolic links.
4. Check the DB2 release level.

Check the home directory permissions

Check that the home directory ownership has been correctly set up by the db2setup program, as listed in Table 10-12.

Table 10-12 DB2 home directory required ownership

Home directory path	Owner	Group	Permissions
/home/<db2_instance_owner>	<db2_instance_owner>	db2iadm1	drwxr-sr-x

Check the DB2 instance owner profile

The DB2 client installation should set up the .profile environment file of the <db2_instance_owner> so that the DB2 environment is set up when the user logs in.

1. The following content should have been added to .profile:

```
if [ -f /home/<db2_instance_owner>/sqllib/db2profile ]; then
   . /home/<db2_instance_owner>/sqllib/db2profile
fi
```

2. If not present, manually edit the <db2_instance_user> .profile file to add the above content.

Check the DB2 instance symbolic links

The DB2 client installation automatically creates a DB2 instance <db2_instance_owner> under the /home/<db2_instance_owner> directory. As part of the instance creation, db2setup should create symbolic links in the /home/<db2_instance_owner>/sqllib directory to files under /usr/lpp/db2_07_01.

Perform the following steps to check whether the symbolic links have been created:

1. Change the directory to /home/<db2_instance_owner>/sqllib.
2. Check whether a number of symbolic links exist pointing to files under /usr/lpp/db2_07_01.

   ```
   # ls -l
   ```

3. If not, issue the following commands:

   ```
   # cd /usr/lpp/db2_07_01/cfg
   # ./db2ln
   ```

Check the DB2 release level

Check that DB2 has the correct internal release level to meet WebSphere requirements:

1. Change to the user <db2_instance_owner>:

   ```
   # su - <db2_instance_owner>
   ```

2. Enter the following command:

   ```
   $ db2level
   ```

   This should generate output similar to the following:

   ```
   DB21085I  Instance "db2inst1" uses DB2 code release "SQL07021" with level
   identifier "03020105" and informational tokens "DB2 v7.1.0.43", "s010504"
   and "U475375a"
   ```

3. An internal release level of 7.1.0.43 should be indicated.

10.5.4 Configure the DB2 client

After the DB2 client installation, a number of configuration tasks must be performed so that WebSphere is able to use it as the repository for its administrative database:

1. Update the JDBC level.
2. Configure the TCP/IP communication mode.
3. Verify the DB2 environment.
4. Update the root environment file.

Update the JDBC level

IBM WebSphere Application Server V4.0 requires the use of JDBC2.0, whereas the default installation of IBM DB2 uses JDBC1.2. To update the DB2 JDBC level, complete the following steps:

1. Change to the user <db2_instance_owner>.

   ```
   # su - <db2_instance_owner>
   ```

2. Add the following content to the end of the <db2_instance_owner> .profile environment file:

   ```
   if [ -f /home/<db2_instance_owner>/sqllib/java12/usejdbc2 ] ; then
      . /home/<db2_instance_owner>/sqllib/java12/usejdbc2
   fi
   ```

Configure the TCP/IP communication mode

DB2 may need to be reconfigured to use TCP/IP as its primary communication method:

1. Change to the user <db2_instance_owner>:

   ```
   # su - <db2_instance_owner>
   ```

2. Check whether TCP/IP is the current DB2 communication method. The following command should return a value of tcpip:

   ```
   $ db2set DB2COMM
   ```

3. If not TCP/IP, reset the DB2COMM DB2 environment variable:

   ```
   $ db2set DB2COMM=tcpip
   ```

4. Exit back to root shell.

5. Edit the /etc/services file to add the following entry:

   ```
   db2cdb2inst1 50000/tcp # Connection port for remote DB2 instance db2inst1
   ```

Verify the DB2 environment

After the above configuration steps, we need to check that the environment being set up by the db2profile and usejdbc2 scripts is correct:

1. Change to the user <db2_instance_owner>:

   ```
   # su - <db2_instance_owner>
   ```

2. Issue the following command:

   ```
   # set | grep [Dd][Bb]2
   ```

3. Check that the environment variables in this output match the values in Table 10-10 on page 296.

Update the root environment file

The WebSphere Application Server will be run under root and will require access to the DB2 environment so that it can access the WebSphere administrative database. This requires that the root account's environment .profile be edited to add the following content at the end of the file.

```
# Set up DB2 environment for root user.
if [ -f /home/<db2_instance_owner>/sqllib/db2profile ] ; then
   . /home/<db2_instance_owner>/sqllib/db2profile
fi

# Force DB2 to use JDBC 2.0.
if [ -f /home/<db2_instance_owner>/sqllib/java12/usejdbc2 ] ; then
   . /home/<db2_instance_owner>/sqllib/java12/usejdbc2
fi
```

10.5.5 Set up access to the remote administrative database

The following steps set up access to the WebSphere Application Server repository database hosted on a remote DB2 server:

1. Change to the user <db2_instance_owner>:

   ```
   # su - <db2_instance_owner>
   ```

2. In order to access the administrative database via TCP/IP, the DB2 node must first be cataloged:

   ```
   $ db2 catalog tcpip node <node_name> remote <remote_hostname> server
   <service_name>
   ```

   Important: The <service_name> used to catalog the remote node must be the same as the database instance connection port name added to the /etc/services file. The <node_name> chosen can be any valid DB2 node name.

3. The administrative database must now be cataloged as part of this TCP/IP node:

   ```
   $ db2 catalog db was1 as was at node <node_hostname>
   ```

4. Check that the database TCP/IP alias is known to the DB2 client:

   ```
   $ db2 list db directory
   ```

   This should give output containing the following:

   ```
   Database 2 entry:
    Database alias                 = WAS
    Database name                  = WAS1
    Node name                      = <node_name>
   ```

```
Database release level          = 9.00
Comment                         =
Directory entry type            = Remote
Catalog node number             = -1
```

5. Verify the connection to the remote database via TCP/IP:

```
$ db2 connect to was user <dbw_instance_owner> using <db2owner_passwd>
$ db2 disconnect current
```

10.6 Install WebSphere Application Server

This section provides detailed instructions for installing, configuring, and verifying WebSphere Application Server V4.0, Advanced Edition for AIX.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install WebSphere.
3. Verify the WebSphere installation.

10.6.1 Preinstallation tasks

Prior to installing IBM WebSphere Application Server V4.0, the following checks and tasks need to be completed on the WebSphere server machine:

1. Check that IP ports are unused.
2. Stop the Web server processes.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing active services that use the following IP ports on the server:
   - 900 (bootstrap port)
   - 9000 (Location Service Daemon)
   - 9080 (default application server)

   We suggest using the following command for this task:

```
# netstat -an | grep LISTEN
```

Stop the Web server processes

The IBM HTTP Server process must be stopped while WebSphere is installed. The WebSphere installation changes the httpd.conf configuration file as part of the Web server plug-in component installation.

1. Log in as root on the IBM HTTP Server machine.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <http_server_install_path>/bin
# ./apachectl stop
```

10.6.2 Install WebSphere

To install IBM WebSphere Application Server V4.0, Advanced Edition using the GUI installer interface, complete the following steps on the WebSphere server machine:

Tip: The WebSphere installer (install.sh) also provides a non-GUI scripted or "silent" mode of operation. See 10.10, "Install WebSphere Application Server - silent mode" on page 324 for details.

1. Log in as root.
2. Start a terminal session.
3. Load the IBM WebSphere Application Server V4.0, Advanced Edition CD-ROM into the CD-ROM drive and mount the CD.

```
# mount -r -v cdrfs /dev/cd0 /mnt
```

4. Change the directory to the installation root.
5. Ensure the DISPLAY and TERM environment variables are properly set.
6. Run the install.sh installation script:

```
# ./install.sh
```

7. In the Welcome window, click **Next**.
8. In the Installation Options window, select **Custom Installation** and click **Next**.
9. In the Choose Application Server Components window, choose all options except IBM HTTP Server, then click **Next**.

Important: Although not listed in the Application Server Components window, the IBM JDK 1.3.0 is automatically installed under the WebSphere installation directory. There is no need to separately install a JDK for use by:

- WebSphere Application Server
- Web server plug-ins

10. In the Choose Webserver Plugin window, choose only the **IBM HTTP Server Plugin**, then click **Next**.
11. In the Database Options window, shown in Figure 10-2 on page 308, enter the following then click **Next**:
    - Database Type: DB2
    - Database Name (Database SID): <was database alias>

      Enter the DB2 TCP/IP alias of the WebSphere administrative database, as created in 10.4.5, “Set up the WebSphere administrative database” on page 296.
    - DB Home: /home/<db2_instance_owner>
    - Database User ID: <db2_instance_owner>
    - Database Password: <db2owner_password>
    - Remote DB: leave unselected

Note: Whether the DB2 database is local or remote, in our test environment we configure the DB2 client to access the database through a catalog alias. Under these conditions, the Remote DB setting should not be selected.

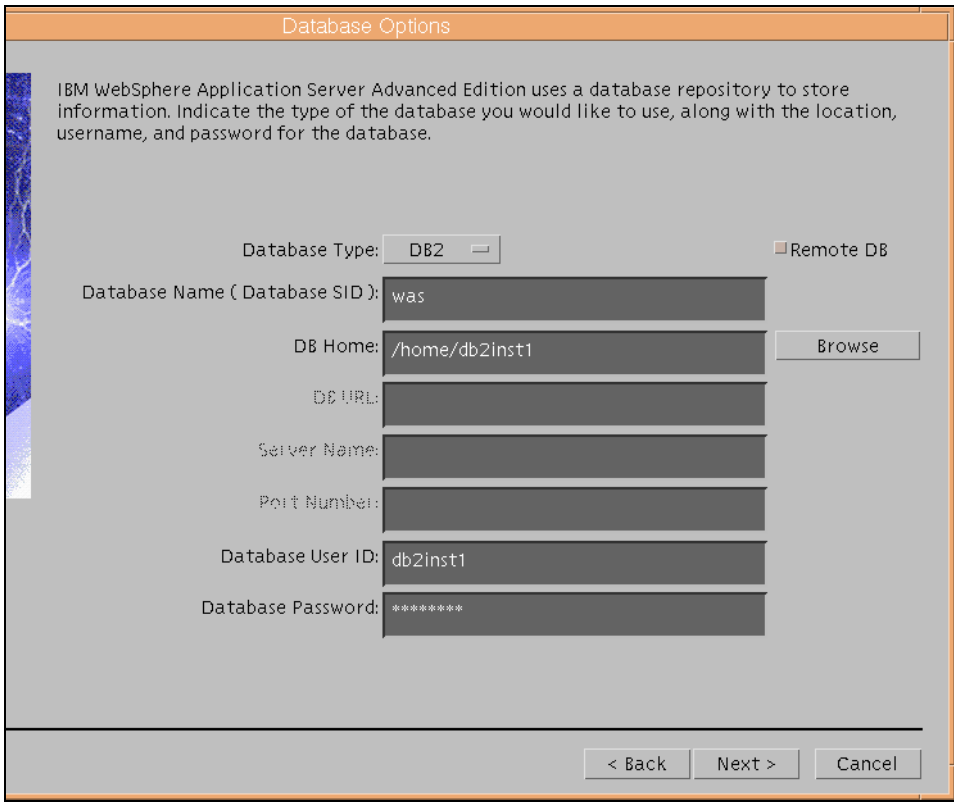


Figure 10-2 Specify DB2 database connection settings

12. In the Select Destination Directory window, shown in Figure 10-3, accept the default location for the WebSphere Application Server. Click **Next** to continue.

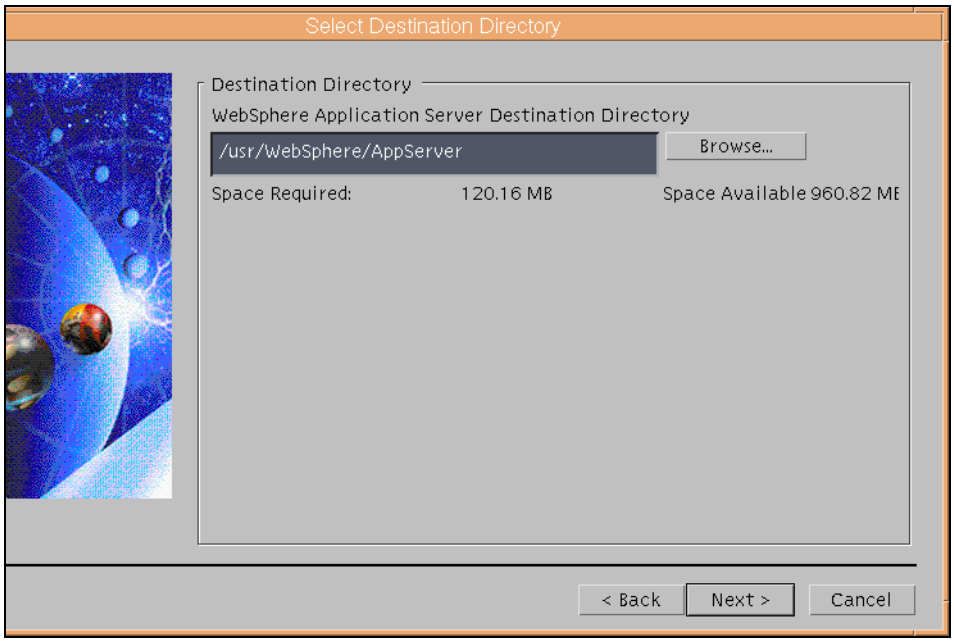


Figure 10-3 Specify the WebSphere destination directory

13. In the Install Options Selected window, check that the correct components have been selected. If yes, click **Install** to start the installation.
14. In the Location of Configuration Files window, enter the path to the IBM HTTP Server configuration file (httpd.conf), then click **Next**.
15. In the Setup Complete window, click **Finish**. The installation of the WebSphere Application Server is now complete.

10.6.3 Verify the WebSphere installation

In order to verify the installation of IBM WebSphere Application Server V4.0, Advanced Edition, the following tasks must be completed in order:

1. Check the installation log.
2. Check the admin.config settings.
3. Check the Web server configuration file changes.
4. Start the WebSphere administrative server processes.
5. Start the WebSphere Default Server.
6. Regenerate the Web server plug-in settings.
7. Restart the Web server processes.

8. Verify the Web server plug-in configuration.

Check the installation log

Check that the installation log <WAS_HOME>/logs/install.log does not contain any errors.

Check the admin.config settings

To check the admin.config settings, perform the following steps:

1. Check that the repository database settings are correct for the database type (DB2), instance (was) and user ID (<db2_instance_owner>) used in our test environment:

```
com.ibm.ejs.sm.adminServer.dbdataSourceClassName=COM.ibm.db2.jdbc.DB2Connec
tionPoolDataSource
com.ibm.ejs.sm.adminServer.dbserverName=
com.ibm.ejs.sm.adminServer.dbportNumber=
com.ibm.ejs.sm.adminServer.dbdatabaseName=<was database alias>
com.ibm.ejs.sm.adminServer.dbuser=<db2_instance_owner>
com.ibm.ejs.sm.adminServer.dbpassword=<db2owner_password>
com.ibm.ejs.sm.adminServer.dbdisable2Phase=true
```

2. Check the DB2 path-related parameter listed in Table 10-13.

Table 10-13 DB2-related paths required in admin.config

Parameter	Must contain path...
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs	<db2_install_path>/sqllib/java12/db2java.zip

3. Check that the WebSphere schema and initial configuration (for example, Default Server) will be written to the repository database on startup, as listed in Table 10-14.

Table 10-14 Required schema creation and initial configuration flags

Parameter	Required value...
com.ibm.ejs.sm.adminServer.createTables	true
install.initial.config	true

Check the Web server configuration file changes

To check the Web server configuration file changes, complete the following steps:

1. Check that the following required settings have been added to the IBM HTTP Server configuration file (httpd.conf) as a result of the WebSphere installation:

```
LoadModule ibm_app_server_http_module
/usr/WebSphere/AppServer/bin/mod_ibm_app_server_http.so
WebSpherePluginConfig /usr/WebSphere/AppServer/config/plugin-cfg.xml
AddModule mod_app_server_http.c
```

2. If not, manually add the above lines to the end of the httpd.conf file and save the changes.

Start up the WebSphere administrative server processes

The WebSphere administrative server needs to be started in order to test the installation as well as the connectivity between WebSphere and the WebSphere administrative database hosted in DB2:

1. Log in as root.
2. Start a terminal session.
3. Run the WebSphere administrative server by issuing the following commands:

```
# cd <WAS_HOME>/bin
# ./startupServer.sh
```

4. The startup of WebSphere administrative server is successful if the following conditions are met:
   a. There are no administrative server error logs in <WAS_HOME>/logs with names that start with __adminServer.
   b. The last line of the <WAS_HOME>/logs/tracefile file is similar to the following:

```
[01.07.05 11:28:01:591 EDT] 7dafee92 Server        A WSVR0023I: Server
__adminServer open for e-business
```

Start up the WebSphere Default Server

The WebSphere installation sets up a default application server (Default Server) in the WebSphere administrative domain. We use this application server and its Web applications to verify that the WebSphere installation is working correctly.

To start up the Default Server, perform the following steps:

1. Run the WebSphere Administrative Console by issuing the following commands:

```
# cd <WAS_HOME>/bin
# ./adminclient.sh
```

2. Right-click the **Default Server** under the <hostname> node and select **Start** from the pop-up menu.

3. The Default Server has been successfully started if the following conditions are met:

   a. The administrative console event messages pane (shown in Figure 10-4) shows the following messages:

      ```
      Transport http listening on port 9,080.
      Command "Default Server.start" completed successfuly.
      ```

   b. The last line of the <WAS_HOME>/logs/Default_Server_stdout.log file is similar to the following:

      ```
      [01.07.03 11:55:03:103 EDT] 7c6ff1d3 Server        A WSVR0023I: Server
      Default Server open for e-business
      ```

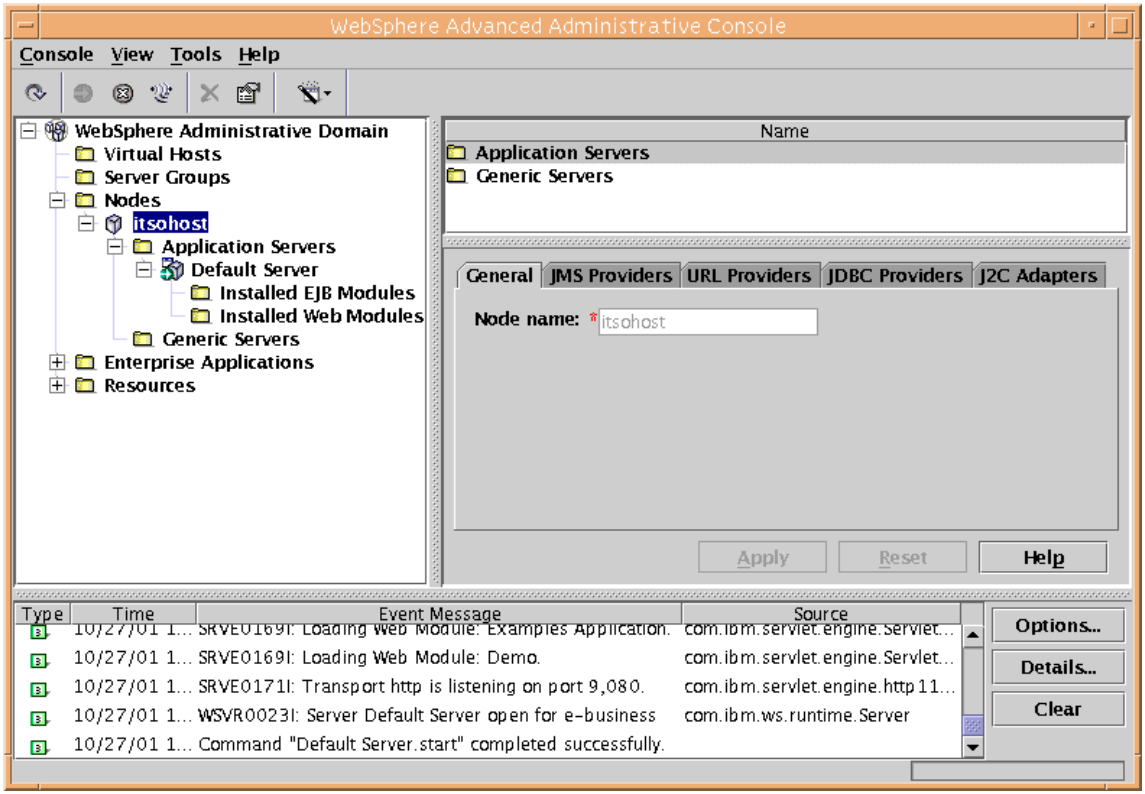


Figure 10-4 Successful startup of Default Server application server

4. Verify that the Default Server Web container has been properly installed and configured by accessing its servlets through the Web server "embedded" within the WebSphere V4.0 Web container:

   a. Using a Web browser, request the following URL:

      ```
      http://<hostname>:9080/servlet/snoop
      ```

A window similar to the one shown in Figure 10-5 should be displayed in your browser.

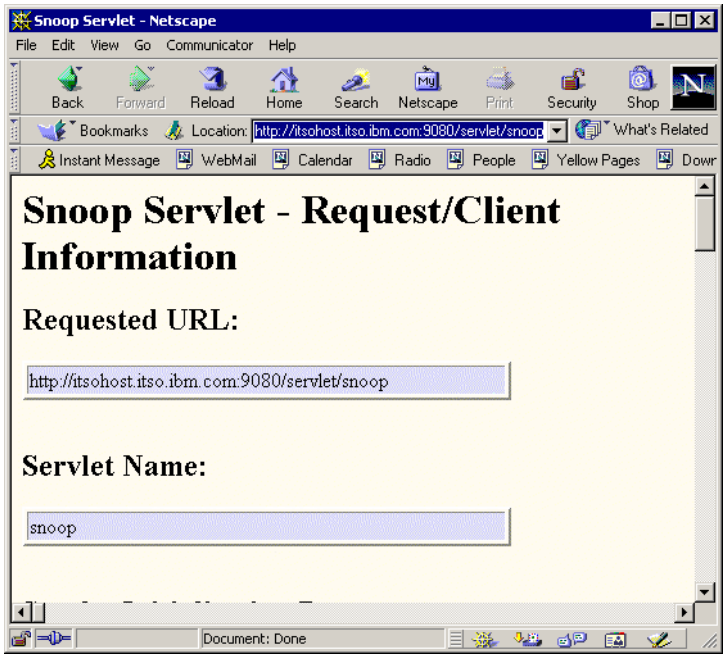


Figure 10-5 Snoop servlet accessed through embedded Web server

b. Using a Web browser, request the following URL:

`http://<hostname>:9080/webapp/examples/showCfg`

> **Note:** The embedded Web server is a new feature introduced with WebSphere V4.0. In previous releases, a stand-alone Web server was required in order to access any resource hosted in WebSphere.

5. Close the WebSphere Administrative Console.

Regenerate the Web server plug-in settings

Before the Default Server can be accessed from a stand-alone Web server (such as IBM HTTP Server) the Web server plug-in settings file <WAS_HOME>/config/plugin-cfg.xml must be regenerated to reflect the following settings used by the Web server plug-in:

- Virtual host settings
- Application Server transports

- Web container URIs

Perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Run the WebSphere Administrative Console by issuing the following command:

```
# cd <WAS_HOME>/bin
# ./adminclient.sh
```

4. Right-click the node <hostname> that contains the Default Server application server and select **Regen Webserver Plugin** from the pop-up menu, as shown in Figure 10-6.

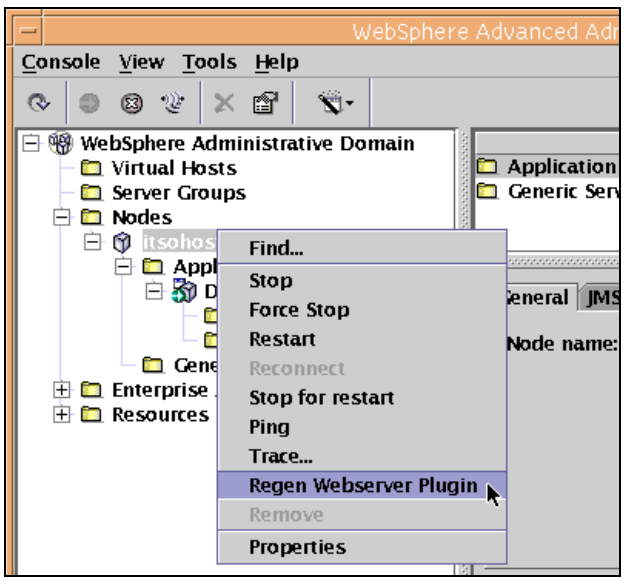


Figure 10-6 Regenerate Web server plug-in settings

Tip: WebSphere provides a command-line tool that can be used to regenerate the Web server plug-in configuration without having to run the WebSphere Administrative Console:

<WAS_HOME>/bin/GenPluginCfg.sh -adminNodeName <hostname>

5. Check that the content of the <WAS_HOME>/config/plugin-cfg.xml file has been updated to include the URIs of servlets contained within Default Server.

Tip: The plug-in regeneration command generates a <Server> element for the Default Server that contains a CloneID attribute:

```
<Server CloneID="stsul7n0" Name="Default Server">
   <Transport Hostname="itsohost" Port="9080" Protocol="http"/>
</Server>
```

In a non-cloned environment this attribute can be removed, resulting in performance improvements in the Web server plug-in.

Restart the Web server processes

The IBM HTTP Server process must be restarted before the Web server plug-in configuration can be tested.

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <http_server_install_path>/bin
# ./apachectl restart
```

Verify the Web server plug-in configuration

The Web server plug-in configuration can be verified by requesting a servlet through the Web server that has already been successfully requested through the Web container's embedded Web server:

1. Using a Web browser, request a servlet URL such as:

   ```
   http://<web_server_hostname>/servlet/snoop
   ```

   or:

   ```
   http://<web_server_hostname>/webapp/examples/showCfg
   ```

10.7 Install the WebSphere plug-in on a remote Web server

This section provides detailed instructions for installing, configuring, and verifying the WebSphere HTTP plug-in on a remote Web server.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install the WebSphere plug-in.

3. Verify the WebSphere plug-in installation.
4. Configure for a remote WebSphere plug-in.
5. Verify the remote plug-in installation.

10.7.1 Preinstallation tasks

The plug-in installation updates the httpd.conf configuration file, so the IBM HTTP Server processes on the Web server machine must be stopped before installing the plug-in.

To stop the IBM HTTP Server processes:

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <http_server_install_path>/bin
# ./apachectl stop
```

10.7.2 Install the WebSphere plug-in

To install the WebSphere plug-in, complete the following steps on the Web server machine:

1. Log in as root.
2. Start a terminal session.
3. Load the IBM WebSphere Application Server V4.0, Advanced Edition CD-ROM into the CD-ROM drive and mount the CD.

   ```
   # mount -r -v cdrfs /dev/cd0 /mnt
   ```

4. Change the directory to the installation root.
5. Ensure that the DISPLAY and TERM environment variables are correctly set.
6. Run the install.sh installation script:

   ```
   # ./install.sh
   ```

7. In the Welcome window, click **Next**.
8. In the Installation Options window, select **Custom Installation** and click **Next**.
9. In the Choose Application Server Components window, choose only the **WebServer Plugins** option, then click **Next**.

Important: Although not listed in the Application Server Components window, the IBM JDK 1.3.0 is automatically installed under the WebSphere installation directory. There is no need to separately install a JDK for use by the Web server plug-in.

10. In the Choose Webserver Plugin window, choose only the **IBM HTTP Server Plugin**, then click **Next**.
11. In the Database Options window, accept the defaults and click **Next**. None of the database settings are actually used by the Web server plug-in.
12. In the Select Destination Directory window, accept the default and click **Next**.

Note: Components installed under <plugin_install_path> by the plug-in installation include:

- Web server plug-in libraries.
- Plug-in configuration file (example).
- Plug-in certificate keystore (example).
- IBM JDK 1.3.0.

13. In the Install Options Selected window, check that the selected components are correct. If yes, click **Install** to start the installation.
14. In the Location of Configuration Files window, enter the path to the IBM HTTP Server configuration file (httpd.conf), then click **Next**.
15. In the Setup Complete window, click **Finish**. The installation of the WebSphere plug-in is now complete.

10.7.3 Verify the WebSphere plug-in installation

In order to verify the installation of the WebSphere HTTP plug-in, the following tasks must be completed on the Web server machine:

1. Check the Web server configuration file changes.

Check the Web server configuration file changes

To check that required settings have been added to the IBM HTTP Server configuration file (httpd.conf), follow the steps described in “Check the Web server configuration file changes” on page 310.

10.7.4 Configure for a remote WebSphere plug-in

In order to support requests from a WebSphere HTTP plug-in installed on a remote Web server, the following tasks must be performed:

1. Configure the WebSphere virtual host.
2. Regenerate the Web server plug-in settings.
3. Copy the plug-in settings to remote server.
4. Restart the Web server.

Configure the WebSphere virtual host

To map requests from the remote Web server to the required virtual host, complete the following steps:

1. Log in as root on the WebSphere server machine.
2. Start a terminal session.
3. Run the WebSphere Administrative Console by issuing the following commands:

   ```
   # cd <WAS_HOME>/bin
   # ./adminclient.sh
   ```

4. Select the **Virtual Hosts** folder in the tree pane of the administrative console.
5. In the details pane, select the **default_host** virtual host.
6. Add three new entries to the Host Aliases list of the default_host virtual host, as shown in Figure 10-7:

   ```
   <Web server hostname>:<port#>
   <Web server hostname.domain.com>:<port#>
   <Web server IP address>:<port#>
   ```

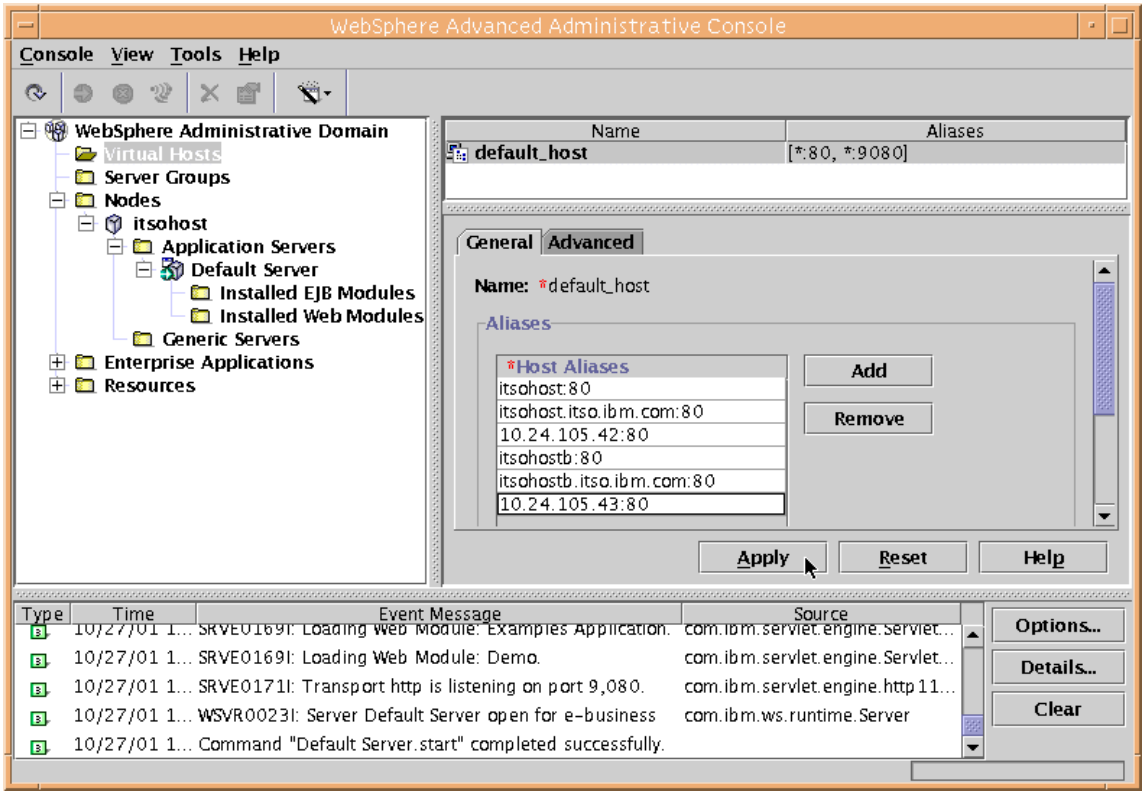


Figure 10-7 Add required aliases to virtual host

7. Click **Apply** to save changes.

> **Important:** Always remember to click the **Apply** button when changing settings with the administrative console. Failure to do so will result in all changes being lost, even if you just click into another resource in the console.

8. Restart each of the application servers that are associated with this virtual host.

Regenerate the Web server plug-in settings

To regenerate the Web server plug-in settings:

1. Follow the steps described in “Regenerate the Web server plug-in settings” on page 313.

2. Check that the content of the <WAS_HOME>/config/plugin-cfg.xml file has been updated to include the remote Web server's hostname amongst the virtual hosts settings.

Copy the plug-in settings to the remote server

Next the regenerated Web server plug-in settings must be copied to the remote Web server:

1. Copy the <WAS_HOME>/config/plugin-cfg.xml file from the WebSphere server machine across to the <WAS_HOME>/config directory on the remote Web server machine.

Important: WebSphere and the WebSphere HTTP plug-in can have different installation paths on the two servers. However, for simplicity and to reduce editing of the plugin-cfg.xml file, we recommend that the plug-in installation use the same path as the full WebSphere installation.

Restart the Web server

Restart the Web server if you want it to load the new plug-in settings immediately, or wait until the plug-in dynamically reloads.

1. Log in as root on the IBM HTTP Server machine.
2. Start a terminal session.
3. Restart the Web server by issuing the following commands:

```
# cd <http_server_install_path>/bin
# ./apachectl restart
```

10.7.5 Verify the remote plug-in configuration

In order to verify the configuration of the WebSphere HTTP plug-in installed on a remote Web server, the following tasks must be performed:

1. Check the plug-in logs.
2. Test the connection to WebSphere.

Check the plug-in logs

To check the plug-in logs, complete the following steps:

1. The location of the WebSphere HTTP plug-in's log is specified by the <Log> element of the plugin-cfg.xml configuration file. By default, this is set to the following:

```
<Log LogLevel="Warning" Name="<plugin_install_path>/logs/native.log"/>
```

2. Check the contents of this log file. If the plug-in has been correctly configured, then the following lines will be written to the end of the file:

```
[Tue Jul 03 11:45:03 2001] 0000634e 00000001 - PLUGIN: Plugins loaded.
[Tue Jul 03 11:45:03 2001] 0000634e 00000001 - PLUGIN:
--------------------System Information----------------------
[Tue Jul 03 11:45:03 2001] 0000634e 00000001 - PLUGIN: Bld date: Aug  9
2001, 05:40:32
[Tue Jul 03 11:45:03 2001] 0000634e 00000001 - PLUGIN: Webserver:
IBM_HTTP_SERVER/1.3.19  Apache/1.3.20 (Unix)
[Tue Jul 03 11:45:03 2001] 0000634e 00000001 - PLUGIN: Hostname = itsohost
[Tue Jul 03 11:45:03 2001] 0000634e 00000001 - PLUGIN: NOFILES = hard:
INFINITE, soft: 2000
[Tue Jul 03 11:45:03 2001] 0000634e 00000001 - PLUGIN: MAX COREFILE SZ =
hard: INFINITE, soft: 1073741312
[Tue Jul 03 11:45:03 2001] 0000634e 00000001 - PLUGIN: DATA = hard:
INFINITE, soft: 134217728
[Tue Jul 03 11:45:03 2001] 0000634e 00000001 - PLUGIN:
------------------------------------------------------------
```

Note: See Appendix C, “The plugin-cfg.xml file definitions” on page 1071 for more details.

Test the connection to WebSphere

To test the remote Web server connection to WebSphere, complete the following steps:

1. Using a Web browser, request the following URL:

   ```
   http://<Web server hostname>/servlet/snoop
   ```

2. If both the Web server and WebSphere have been correctly configured to support remote HTTP plug-in access, then a window similar to Figure 10-8 will be obtained.

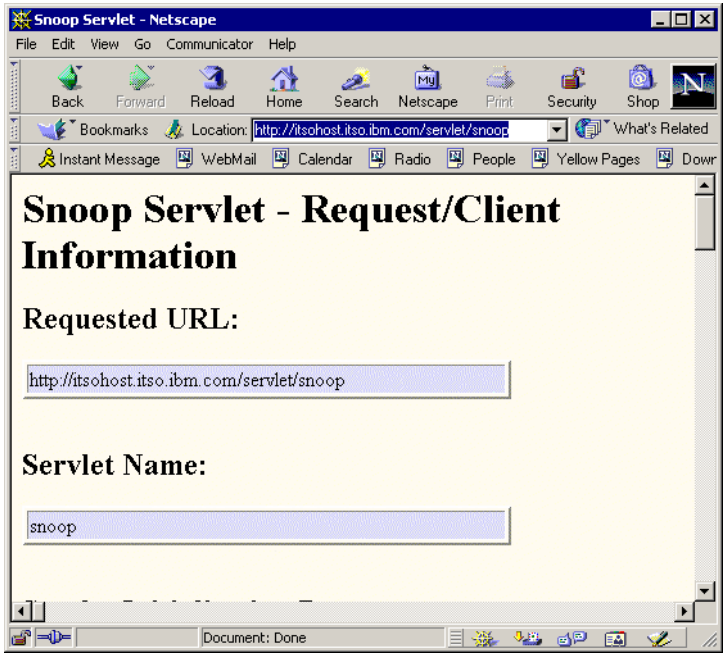


Figure 10-8 Request handled through remote HTTP plug-in

10.8 Install the new WebSphere node into an existing domain

This section provides detailed instructions for installing, configuring, and verifying a new WebSphere node into an existing WebSphere V4.0 administrative domain.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install the WebSphere.
3. Verify the WebSphere installation.

10.8.1 Preinstallation tasks

Prior to installing the new WebSphere node, complete the basic preinstallation tasks specified in 10.6.1, “Preinstallation tasks” on page 305.

Important: This section assumes that the IBM HTTP Server and DB2 client (in place of the DB2 server) are both already installed and configured on the server on which you will be installing this new WebSphere node.

10.8.2 Install WebSphere

To install the new WebSphere node into an existing WebSphere domain, complete the basic installation tasks specified in 10.6.2, “Install WebSphere” on page 306, except for the following:

1. In the Database Options window, specify:
   - Database Name: specify DB2 TCP/IP alias to the remote database.
   - Path: specify the installation path of the DB2 client instance (this should be automatically detected and presented as the default).
   - Remote Database: leave this unselected. The DB2 client will be configured to present a “local” alias to the remote DB2 database. See 10.5, “Install the database client” on page 298 for details.

10.8.3 Verify the WebSphere installation

In order to verify the new WebSphere node installed into an existing WebSphere domain, complete the basic verification tasks specified in 10.6.3, “Verify the WebSphere installation” on page 309, except for the following:

1. For each new node sharing an existing administrative database, ensure the database tables are not created and the default resources are not installed a second time. Edit the <WAS_HOME>/bin/admin.config file and set the values of the following settings as shown:

```
install.initial.config=false
com.ibm.ejs.adminServer.createTables=false
```

10.9 Configure the WebSphere HTTP transport for SSL

See 9.9, “Configure WebSphere HTTP transport for SSL” on page 245 for a description of how to configure a WebSphere HTTP transport to use SSL encryption. Apart from file system path changes, the method used on the each platform is the same.

Note: To execute the Key Management Utilities on the AIX platform, perform the following steps:

1. To start the WebSphere IBM Key Management Utility:
   a. Log in as root on the WebSphere server machine.
   b. Start a terminal session.
   c. Execute the following commands:

```
# cd <WAS_HOME>/bin
# ./ikeyman.sh
```

2. To start the Web server IBM Key Management Utility:
   a. Log in as root on the IBM HTTP Server machine.
   b. Start a terminal session.
   c. Execute the following commands:

```
# JAVA_HOME="/usr/jdk_base" ; export JAVA_HOME
# cd /usr/bin
# ./gsk5ikm
```

10.10 Install WebSphere Application Server - silent mode

This section describes how to install WebSphere Application Server V4.0, Advanced Edition using the non-interactive, or silent, mode. To complete a silent installation, you will use the default response file or create a customized one, then execute the installation script for WebSphere Application Server, supplying the response file as a command-line parameter.

These instructions assume the following:

- Your machine has sufficient memory and disk space for your installation.
- You have installed and configured a database.
- You do not have a previous version of WebSphere Application Server already installed on this machine. If you do have a previous version of WebSphere Application Server already installed, do not follow these instructions. Instead, see Chapter 25, “Migration” on page 1031.
- If you are using IBM HTTP Server as your Web server, you will install it at the same time and onto the same node as you install WebSphere Application Server. If you are using another supported Web server with WebSphere Application Server, you have already installed it onto the same node as WebSphere Application Server.

Note: IBM HTTP Server is supplied with WebSphere Application Server. If you plan to use a different Web server, you must purchase it and install it separately. It is recommended that the Web server be installed before WebSphere Application Server.

10.10.1 Using the default response file

A default response file, named install.script, is supplied with WebSphere Application Server. You can use this default response file to install WebSphere Application Server using the default options, or as a template for creating a customized response file.

If you use the default response file to install WebSphere Application Server using the default options, the following software and other resources are installed:

- IBM Java 2 Software Developer's Kit (SDK) 1.3.0
- IBM HTTP Server 1.3.19
- IBM WebSphere Application Server V4.0
- WebSphere Application Server application samples
- Documentation in U.S. English

Note: All products except IBM HTTP Server are installed into the directory /usr/WebSphere/AppServer. IBM HTTP Server is installed into the directory /usr/HTTPServer. In addition, WebSphere Application Server is configured for use with IBM HTTP Server when you use the default response file.

10.10.2 Using a customized response file

The default response file can also be used as a template for creating a customized response file. The default response file can be edited to enable the configuration of WebSphere Application Server with a different supported Web server or database, or to install the products in a different directory. Detailed comments within the default response file guide you through the installation and configuration options available for performing a silent installation.

10.10.3 Performing a silent installation

Perform the following steps to create a customized response file (if desired) to install WebSphere Application Server. These instructions assume that the installation is being performed from the product CD-ROM:

1. Ensure that you are logged into the machine with superuser (root) privileges.

2. If a Web server is running on your system, stop the Web server. If you plan to install IBM HTTP Server 1.3.19 as part of the WebSphere Application Server installation and you have a level of IBM HTTP Server prior to 1.3.19 on your system, you must uninstall it for the WebSphere Application Server installation program to install IBM HTTP Server 1.3.19.
3. Ensure that the DISPLAY and TERM environment variables set correctly.
4. Insert the IBM WebSphere Application Server V4.0, Advanced Edition CD-ROM into the CD-ROM drive. The following steps assume that you have already created and properly configured a CD-ROM mount point (for example, /mnt). If you have not already done so, refer to your AIX operating system documentation for more information.
5. Invoke SMIT for mounting a file system by entering the following command:

   ```
   # smitty mountfs
   ```

   a. With the cursor in the FILE SYSTEM name field, press F4, and then choose the appropriate CD-ROM file system that you want to mount.

   b. In the DIRECTORY over which to mount field, type the name of the mount point for the CD-ROM (these instructions assume you are using the /mnt mount point).

   c. With the cursor in the TYPE of file system field, press F4, and then choose the cdrfs option.

   d. Verify or change the entries in the remaining fields, depending on how you want to mount the CD-ROM, and then click Return. SMIT mounts the CD-ROM as a file system. When the process is complete, exit from SMIT.
6. Navigate to the /mnt directory.
7. Ensure that you are in the /mnt directory and create a copy of the default response file by using the **cp** command, as follows:

   ```
   # cp install.script <new_install.script>
   ```

   In this command, `<new_install.script>` represents the full path name of the copy of the default response file you are creating (for example, `/tmp/my_install.script`). The name of your response file must have a .script extension.
8. If you plan to install WebSphere Application Server by using the default options included in the default response file, proceed to the next step.

   If you plan to use the default response file as a template for creating a customized response file, perform the following steps:

   a. Use a text editor to open your copy of the default response file, <new_install.script>.

b. Use the detailed comments throughout the file to help you select the appropriate options for your WebSphere Application Server installation.

c. Save the changes that you have made to the customized response file.

9. Run the installation script by using the following commands. The install.sh script uses the response file to install the components and options that you have selected. The variable <new_install.script> represents the full path name of the copy of the default response file or the customized response file that you have created. (for example, /tmp/new_install.script).

```
# ./install.sh -silent -responseFile <new_install.script>
```

If you choose to install the plug-in for IBM HTTP Server, the installation process checks if you have the correct version of the Web server on your machine. If you do not have IBM HTTP Server installed on your machine, the installation process performs one of the following actions based on whether you have indicated in your response file to install IBM HTTP Server:

– If you indicated in your response file that you do want to have IBM HTTP Server installed, the installation process installs the plug-in for it.

– If you indicated in your response file that you do not want to have IBM HTTP Server installed, the script exits without installing the plug-in.

10. After installation is complete, refer to the log file named install.log located in the /tmp directory to determine if the silent installation was successful. A copy of this file also exists in the directory <WAS_HOME>/logs.

11. Unmount the CD-ROM before removing it from the CD-ROM drive by using the **umount** command, as follows:

```
# umount /mnt
```

12. If you installed IBM HTTP Server as part of the WebSphere Application Server silent installation, you might need to configure it. Perform the following steps to verify that the IBM HTTP Server is installed correctly:

a. Ensure that the Web server is running or start it by entering the following command:

```
# /usr/HTTPServer/bin/apachectl start
```

b. Start a Web browser and type the name of the host machine as the URL (http://host_machine). If you see the Welcome to the IBM HTTP Server Web page, the server has been installed correctly.

See the “Using silent installation on AIX “ article in the InfoCenter for further details.

11

Solaris installation steps

This chapter provides detailed procedures for installing, configuring, and verifying a number of the scenarios described in Chapter 8, “Installation approach” on page 127, for an environment consisting of the following components:

- Operating system - Solaris 2.7
- Database server - Oracle 8i Server
- Web server - Netscape iPlanet Web Server
- Application server - WebSphere Application Server V4.0, Advanced Edition

The procedures described are intended to be used as working examples in conjunction with the product installation guides for all the possible values that may be unique within your runtime environment. This chapter is organized into the following sections:

- Planning
- Install Solaris
- Install the Web server
- Install the database server
- Install the database client
- Install WebSphere Application Server V4.0, Advanced Edition
- Install the WebSphere plug-in on the remote Web server

- Install a new WebSphere node into an existing domain
- Install WebSphere Application Server - silent mode

11.1 Planning

This section defines the hardware and software used within the Solaris environment to test a number of different WebSphere Application Server V4.0 configuration scenarios.

11.1.1 Hardware and software prerequisites

WebSphere Application Server V4.0, Advanced Edition has the following hardware and software requirements.

Hardware

- 250 MB diskspace (minimum) for WebSphere Application Server
- 1160 MB diskspace (minimum) for Oracle 8i Server Enterprise Edition
- 350 MB diskspace (minimum) for Oracle 8i Client
- 130 MB diskspace (minimum) for Netscape iPlanet Web Server
- At least 400MB swap space, or twice the physical RAM size, whichever is greater.

Software

- Solaris 2.7 or 2.8 + prerequisites (see 11.2, “Install Solaris” on page 337 for Solaris 2.7).
- IBM WebSphere Application Server V4.0, Advanced Edition
- Oracle 8i Release 3 (8.1.7) for Solaris
- Netscape iPlanet Web Server 4.1
- Netscape Communicator V4.61 or higher

Note: For further details on requirements, see the following documentation:

1. WebSphere -

   http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

2. Oracle 8i - refer to the *Oracle 8i Installation Guide*, Release 3 (8.1.7) for Sun SPARC Solaris, Part No. A85471.

3. Netscape iPlanet -

   http://docs.iplanet.com/docs/manuals/enterprise/50/ig/contents.htm.

11.1.2 Software used in our test environment

We used the following software in our test environment:

- Solaris 2.7 + prerequisites
- IBM WebSphere Application Server V4.0, Advanced Edition
- Oracle 8i Release 3 (8.1.7) Enterprise Edition for Solaris
- Netscape iPlanet Web Server, Enterprise Edition V4.1 SP7 (4.1.7)
- Netscape Communicator 4.76 for Solaris
- Sun JDK 1.3.0.3 (included in WebSphere Application Server package)

Note: Other Web servers and database software may be used, as documented in the *Product Installation Guide*.

Product installation roots

The variables listed in Table 11-1 are used frequently throughout this documentation to represent the root installation directories of the software components.

Table 11-1 Product installation roots

Variable	Default value	Component
<WAS_HOME>	/opt/WebSphere/AppServer	WebSphere
<plugin_install_path>	/opt/WebSphere/AppServer	WebSphere HTTP plug-in
<oracle_home>	/opt/oracle8/u01/app/oracle/product/8.1.7	Oracle 8i Server
<http_server_install_path>	/opt/netscape/server4	Netscape iPlanet

Installation variables

The variables listed in Table 11-2 are used frequently throughout this documentation to represent settings used during the installation of software components.

Table 11-2 Common installation settings

Variable	Default value	Component
listener_name	LISTENER	Oracle 8i Server
listener_port	1521	Oracle 8i Server
database_SID	was	Oracle 8i server

11.1.3 Hardware used in our test environment

This section describes the hardware used within our test WebSphere Application Server V4.0 environment on Solaris.

- Server 1
  - Sun SPARC Ultra 60 (600-6486-01)
  - 450 MHZ UltraSPARC II CPU
  - 1 GB RAM
  - 17 GB hard disk
  - 1 Ethernet adapter (built-in)
- Server 2
  - Sun SPARC Ultra 60 (600-6486-01)
  - 450 MHZ UltraSPARC II CPU
  - 1 GB RAM
  - 17 GB hard disk
  - 1 Ethernet adapter (built-in)

11.1.4 Example scenarios

Within this test environment, we describe the tasks necessary to install and configure the following scenarios that are described in Chapter 8, “Installation approach” on page 127:

- Scenario A - basic one-server configuration (server 1), but using the WebSphere embedded Web server in place of the usual stand-alone Web server.

- Scenario B - basic one-server configuration (server 1), but using a stand-alone Web server and the WebSphere HTTP plug-in.
- Scenario D - basic two-server configuration, one server hosting WebSphere and the database server (server 1), the other hosting the Web server and WebSphere HTTP plug-in (server 2). Communication between plug-in and WebSphere is HTTP (unencrypted).
- Scenario H - basic two-server configuration using a single WebSphere administrative domain database.

Important: We were unable to test any scenarios within this test environment that required SSL support by the iPlanet Web Server plug-in.

The IBM GSKit (the SSL toolkit used by the plug-in) requires that all applications be compiled with Sun Workshop 5 in order for SSL to work on Solaris. iPlanet on Solaris is compiled with Sun Workshop 4.2 causing exceptions not to be handled correctly in the GSK code.

This issue has since been corrected by E-fix PQ52815 which allows SSL from iPlanet plug-in to WebSphere Application Server V4.0.1. This fix will be included in WebSphere V4.0.2.

Installation and configuration tasks

Table 11-3 provides a detailed summary of the tasks and the order in which they must be performed that are necessary to install and configure each of the example scenarios described above.

To install and configure an example, perform the tasks listed for the example in the specified order. Only those tasks that are numbered (non-blank) are to be performed for a particular example. Detailed descriptions of each of these tasks can be found later in this chapter.

Table 11-3 Installation and configuration tasks broken down by scenario

Server	Task	Example scenario			
		A	B	D	H
1	Operating System: Installation of Solaris operating system and any required patches: 1. Install operating system 2. Apply any patches and packages as detailed in the WebSphere platform specific prerequisites 3. Set up file systems See 11.2, “Install Solaris” on page 337.	1	1	1	1
1	Web server: Installation and configuration of iPlanet Web Server: 1. Perform preinstallation tasks 2. Install iPlanet Web Server 3. Configure iPlanet Web Server 4. Verify iPlanet Web Server See 11.3, “Install the Web server” on page 340.	-	2	-	2
1	Database server: Installation and configuration of Oracle 8i server necessary to support usage by WebSphere: 1. Perform preinstallation tasks 2. Install Oracle 8i server 3. Configure Oracle 8i server 4. Set up WebSphere administrative database See 11.4, “Install the database server” on page 349.	2	3	2	3

Server	Task	Example scenario			
		A	B	D	H
1	WebSphere Application Server: Installation and basic configuration of WebSphere necessary to handle requests through the “embedded” Web server: 1. Perform preinstallation tasks 2. Install WebSphere 3. Verify WebSphere installation See 11.6, “Install WebSphere Application Server” on page 370.	3	4	3	4
2	Web server: Installation and configuration remote iPlanet Web Server: 1. Perform preinstallation tasks 2. Install iPlanet Web Server 3. Configure iPlanet Web Server 4. Verify iPlanet Web Server See 11.3, “Install the Web server” on page 340.	-	-	4	5
2	Database client: Installation and configuration of Oracle 8i client necessary to support remote usage by WebSphere: 1. Perform preinstallation tasks 2. Install Oracle 8i client 3. Configure Oracle 8i client 4. Set up access to remote WebSphere administrative database See 11.5, “Install the database client” on page 365.	-	-	-	6

Server	Task	Example scenario			
		A	B	D	H
2	WebSphere Application Server: Installation and basic configuration of WebSphere necessary to handle requests through the “embedded” Web server: 1. Perform preinstallation tasks 2. Install WebSphere 3. Verify WebSphere installation See 11.6, “Install WebSphere Application Server” on page 370.	-	-	-	7
2	WebSphere HTTP plug-in: Basic installation and configuration of WebSphere HTTP plug-in on remote Web server: 1. Perform preinstallation tasks 2. Install WebSphere plug-in 3. Verify WebSphere plug-in installation 4. Configure for remote WebSphere plug-in 5. Verify remote WebSphere plug-in See 11.7, “Install the WebSphere plug-in on a remote Web server” on page 386.	-	-	5	-
1	Web server: Test that Web server and plug-in can access the current configuration of the WebSphere Default Server: 1. Restart Web server 2. Verify configuration – `http://<server1 hostname>/webapp/examples/showCfg`	-	5	-	8
2	Web server: Test that Web server and plug-in can access the current configuration of the WebSphere Default Server: 1. Restart Web server 2. Verify configuration – `http://<server2 hostname>/webapp/examples/showCfg`	-	-	6	9

11.2 Install Solaris

Prior to installing any of the WebSphere components, the proper level of the operating system must be installed.

- File systems
- Solaris patches
- Solaris packages

11.2.1 File systems

During the interactive phase of the Solaris 2.7 operating system installation, you will need to customize the file systems and allocate the necessary space for the software components installed in subsequent steps.

Table 11-4 provides general guidelines for the space required by each component.

Table 11-4 Component disk space requirements

Component	File system	Disk space required (MB)
iPlanet Web Server 4.1.7	/opt	130
WebSphere Application Server V4.0	/opt	250
Oracle 8i client	/opt/oracle8/u01	350
Oracle 8i server	/opt/oracle8/u01	1160
Oracle databases - WAS	/opt/oracle8/u02	300 (each)

Based on the disk space requirements listed in Table 11-4, we recommend the file systems listed in Table 11-5 for this test environment.

Table 11-5 Recommended file systems

File system	Description	Usage	Size (MB)
/	root		500
/swap	swap file	1-2 times the size of physical memory	512
/opt	Program files	WebSphere iPlanet Web Server Oracle 8i server Oracle 8i client	3000

File system	Description	Usage	Size (MB)
/export/home	User home directory	Home directories	1000
/tmp	Temporary files		200

11.2.2 Solaris patches

Prior to installing any of the software components used on a particular Solaris server within the test environment, a check must be performed to determine whether all required patches (and versions) are installed on that server.

Check the installed patches

To check that the required Solaris patches are installed:

1. Ensure the Solaris 7 patches listed in Table 11-6 are installed at the indicated level or higher.

Table 11-6 Solaris 7 patches required by test environment software component

Solaris 7 patch ID	Description	Required by components...
106980-10	Libthread patch	WebSphere, iPlanet Web Server
107636-03	SunOS 5.7: X Input and Output Method patch	WebSphere, Oracle 8i
108376-07	OpenWindows 3.6.1: Xsun patch	WebSphere
106541-12	SunOS 5.7: kernel update patch	WebSphere, iPlanet Web Server
107544-03	SunOS 5.7: /usr/lib/fs/ufs/fsck patch	WebSphere
106950-09	SunOS 5.7: Linker patch	WebSphere
106327-08	SunOS 5.7: Shared library patch for C++	WebSphere
106300-09	SunOS 5.7: Shared library patch for 64bit C++	WebSphere
107081-20	Motif 1.2.7 and 2.1.1: Runtime library patch for Solaris 2.7	WebSphere
105181-15	Kernel patch	Oracle 8i

2. To verify whether a particular patch is installed, issue the following command:

```
# showrev -p | grep <patch_id>
```

Where <patch_id> is a patch ID from the required patches table above.

For example:

```
# showrev -p | grep 107081
```

Install new or updated patches

If any of the required patches are not installed, the simplest fix is to download and install the latest recommended Solaris 2.7 patch cluster from the following site:

http://sunsolve.sun.com/

To apply a recommended patch cluster to the Solaris 2.7 operating system to bring patch levels up to required versions, do the following:

1. Log in as root and start a terminal session.
2. Download the recommended Solaris 2.7 patch cluster archive 7_Recommended.zip (dated 16-JUL-2001) from http://sunsolve.sun.com/.
3. Change the operating system to single user state:

   ```
   # /usr/sbin/shutdown -iS -y -g0
   ```

4. Copy archive to a temporary directory with more than 50 MB diskspace free:

   ```
   # cp 7_Recommended.zip /tmp
   ```

5. Unzip 7_Recommended.zip archive in place:

   ```
   # cd /tmp
   # /usr/bin/unzip 7_Recommended.zip
   ```

6. Run the patch cluster installer:

   ```
   # ./install_cluster
   ```

   Application of the patch cluster takes 15 to 20 minutes.

7. Review the contents of the /var/sadm/install_data/Solaris_7_Recommended_log file to determine whether any problems or errors occurred.

   Note: The log will report a number and error for each of those patches in the cluster that are already installed on the server with the same version.

8. Reboot the system:

   ```
   # /usr/sbin/shutdown -i6 -y -g0
   ```

11.2.3 Solaris packages

Prior to installing any of the software components used on a particular Solaris server within the test environment, a check must be performed to determine whether all required packages are installed on that server.

Check the installed packages

To check that the required Solaris packages are installed:

1. Ensure the Solaris 7 packages listed in Table 11-7 are installed.

Table 11-7 Solaris 7 packages required by Oracle8i

Solaris 7 package name	Description
SUNWarc	Archive libraries
SUNWbtool	CCS tools bundled with SunOS
SUNWhea	SunOS header files
SUNWlibm	Sun workshop bundled libm
SUNWlibms	Sun workshop bundled shared libm
SUNWsprot	Solaris bundled tools
SUNWtoo	Programming tools

2. To verify whether a particular patch is installed, issue the following command:

```
# pkginfo -i <package_name>
```

   Where <package_name> is a package name from the required packages table above.

   For example:

```
# pkginfo -i SUNWarc
```

Install new or updated packages

For details regarding the installation of new or updated Solaris packages, see your Solaris Administration documentation.

11.3 Install the Web server

This section provides detailed instructions for installing, configuring, and verifying Netscape iPlanet Web Server V4.1.7 for Solaris.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install iPlanet Web Server.
3. Configure iPlanet Web Server.
4. Verify iPlanet Web Server.

Note: Whether or not a particular task needs to be performed, and the order in which tasks must be performed, is identified during the planning stage and is dependent upon the needs of the particular topology or scenario being installed. See 11.1, “Planning” on page 330 for details on those tasks to be performed for each example.

11.3.1 Preinstallation tasks

Prior to installing Netscape iPlanet Web Server, Enterprise Edition V4.1 SP7 (4.1.7), the following checks and tasks must be completed on the Web server machine:

1. Determine the server host name and IP address.
2. Check that IP ports are unused.
3. Install Netscape Communicator.
4. Create a runtime UNIX account.

Note: For more detailed information on iPlanet Web Server refer to the following URL:

`http://developer.iplanet.com/docs/manuals/enterprise.html`

Determine the server host name and IP address

Use the hostname and nslookup commands to determine the host name and IP address of the server onto which the iPlanet Web Server will be installed.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing services on the server that use the following IP ports:
   - 80 (standard HTTP port)
   - 443 (standard HTTPS port)
   - 8888 (iPlanet administration server)

We suggest using the following command for this task:

```
# netstat -a | grep LISTEN
```

Install Netscape Communicator

The iPlanet Web Server requires Netscape Communicator to be used for accessing the iPlanet administration server.

Complete the following steps to install Netscape Communicator:

1. Start a Solaris Console window, and enter the following:

   ```
   # cd /opt
   # mkdir ns476
   ```

2. To download Netscape Communicator V4.76, enter the following URL:

   http://www.sun.com/solaris/netscape/getnetscape476.html

   Download the file to the /opt/ns476 directory.

3. Uncompress the download file by typing the following:

   Check the name of the downloaded Netscape Communicator file.

   – If the Netscape file does not have a .Z suffix, your browser has uncompressed the Netscape file automatically. Use the following command to extract the files.

     ```
     # tar -xf <filename>
     ```

   – If the Netscape file has a .Z suffix, your browser did not uncompress the Netscape file automatically. Use the following command to uncompress and extract the files.

     ```
     # zcat <filename>.Z | tar -xvf -
     ```

4. Change to the directory where the compressed version of Netscape Communicator was extracted.

   For example:

   ```
   # cd /opt/ns476
   ```

5. Install Netscape Communicator using **pkgadd**:

   ```
   # pkgadd -d `pwd` NSCPcom
   ```

6. Follow install instruction prompts.

7. To start Netscape Communicator do one of the following:

   – From the command line:

     ```
     # cd /opt/NSCPcom
     # ./netscape
     ```

   – Click the Internet icon (Globe) from the Solaris 7 CDE Front Panel

8. Add an entry to your PATH for the directory in which Netscape Communicator is installed. For example, if Netscape Communicator were installed in the default directory, you would add /opt/NSCPcom to your PATH statement. The path may be set in the .dtprofile, .login, or .cshrc file.

   For example, we added the following path to the /etc/.dtprofile file:

   ```
   export PATH=$PATH:/opt/NSCPcom
   ```

9. Configure Netscape Communicator by clicking **Edit** > **Preferences** from the menu bar. We configured the SOCKS server, fonts, and home page.

Note: Netscape Communicator V4.76 installation instructions can be found at:

http://www.sun.com/software/solaris/netscape/476_install.html

Create a runtime UNIX account

Create a UNIX user account to be used by the iPlanet Web Server. This account should have restricted access for security reasons. To create a user:

1. In this example, we create a user and group both named “www”.

   ```
   # groupadd www
   # useradd -g www -d /export/home/www -m -s /bin/ksh www
   # passwd www
   ```

After the last command above, you will be prompted for a password for the new user, and then asked to confirm it.

11.3.2 Install iPlanet Web Server

To install iPlanet Web Server V4.1.7, complete the following steps on the Web server machine:

1. Log in as root
2. Start a terminal session.
3. Insert iPlanet Web Server CD-ROM in the CD-ROM drive. The Volume Management daemon will automatically mount the CD under /cdrom/cdrom0.
4. Copy the iPlanet Web Server tar file to a temp directory on your system.

   For example:

   ```
   # mkdir -p /opt/ip417
   # cd /<root_iplanet_cdrom>/solaris/enterprise
   # cp enterprise.tar /opt/ip417/
   ```

5. Uncompress the tar file by typing the following commands:

   ```
   # cd /opt/ip417
   # tar -xvf enterprise.tar
   ```

6. Start the iPlanet Web Server install by typing the following command:

   ```
   # ./setup
   ```

7. When you are prompted with the message `Would you like to continue with the installation [Yes]:` press Enter (for Yes as default).

Tips: The following tips are used for navigation during the install:

- Press Enter to choose the default and go to the next window.
- Type Control-B to back to the previous window.
- Type Control-C to cancel the installation program.
- You can enter multiple items using commas to separate them, for example 1,2,3.

8. When you are prompted with the message `Do you agree to license terms? [No]:` type Yes, and then press Enter.
9. When you are prompted with the message `Choose the installation type [2]:` press Enter to accept the default (Typical installation).
10. When you are prompted with the message `Install location [/usr/netscape/server4]:` enter the path `/opt/netscape/server4`, then press Enter.
11. When you are prompted with the message `Specify the components you wish to install [All]:` press Enter.
12. When you are prompted with the message `Specify the components you wish to install [1,2,3,4,5,6,8]: type 1,2,3,4,5` and then press Enter.

Note: The following components are required by WebSphere Application Server:

- 1 - Server Core.
- 2 - Java Runtime Environment.
- 3 - Java Support.
- 4 - SSJS Support.
- 5 - SSJS Database Support.

13. When you are prompted with the message `Computer name [your_hostname.domain]:` press Enter.
14. When you are prompted with the message `System User [nobody]:` type the user created in “Create a runtime UNIX account” on page 343, for example `www`, and then press Enter.

15. When you are prompted with the message `System Group [nobody]:` type the group created in “Create a runtime UNIX account” on page 343, for example `www`, and then press Enter.
16. When you are prompted with the message `Run iWS Administration Server as [root]:` press Enter.
17. When you are prompted with the message `iWS Administration Server User Name [admin]:` press Enter. You will be prompt for a Password: <password>
18. When you are prompted with the message `iWS Admin Server Port [8888]:` press Enter.
19. When you are prompted with the message `Web Server Port [80]:` press Enter.
20. When you are prompted with the message `Do you want to register this with an existing Directory Server [No]:` press Enter.
21. When you are prompted with the message Web Server Content Root `[/opt/netscape/server4/docs]:` press Enter.
22. When you are prompted with the message Do you want to use your own JDK [No]: press Enter.
23. Review messages to make sure everything was extracted and installed successfully. Press Enter to continue and return to the command prompt.
24. The iPlanet Web Server installation is complete.

11.3.3 Configure iPlanet Web Server

After installation of iPlanet Web Server, the following configuration tasks must be completed on the iPlanet Web Server machine:

1. Create iPlanet system startup script.

Create the iPlanet system startup script

Create a script named “iplanet” in the /etc/init.d directory to be used to cleanly start up and shut down the iPlanet Web Server processes on system startup and shutdown, respectively.

1. Change to the /etc/init.d directory.

```
# cd /etc/init.d
```

2. Create a script called “iplanet”.

   Example 11-1 provides a sample iPlanet startup script.

Example 11-1 Sample iPlanet startup script

```
#!/bin/ksh
#
```

```
# Filename: iplanet
# Purpose: Cleanly startup/shutdown the iPlanet Web Server on system
# startup/shutdown respectively.

IPLANET_HOME=/opt/netscape/server4
INSTANCES="itsohost.itso.ibm.com admserv"

case "$1" in
'start')
   for i in ${INSTANCES}
   do
      ${IPLANET_HOME}/https-${i}/start
   done
   ;;
'end')
   for i in ${INSTANCES}
   do
      ${IPLANET_HOME}/https-${i}/stop
   done
   ;;
esac
```

11.3.4 Verify iPlanet Web Server

In order to verify the iPlanet Web Server V4.1.7 installation, perform the following checks on the iPlanet Web Server machine:

1. Check the server status.
2. Check the process status.
3. Check request handling.

Check the server status

To check the iPlanet Web Server status, perform the following steps:

1. Start the iPlanet Administration Server process and load its Console:

   ```
   # cd <http_server_install_path>
   # ./startconsole
   ```

2. In the Select a Server pull-down, select the Web server, for example itsohost.itso.ibm.com, as shown in Figure 11-1.

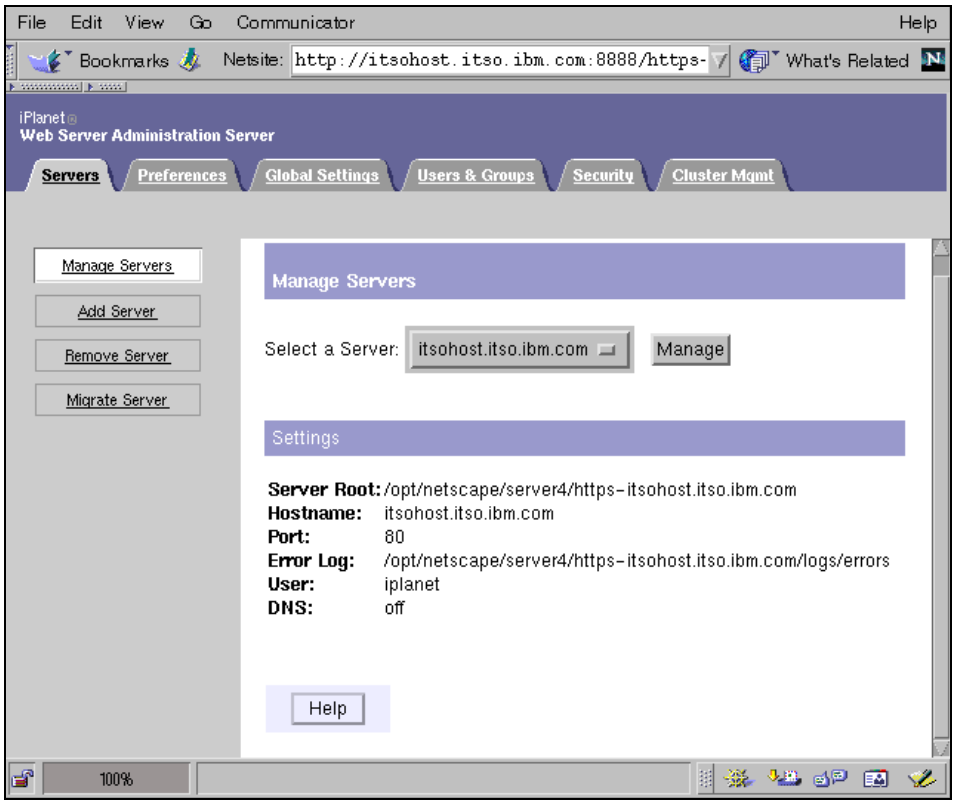


Figure 11-1 Server status

3. Click the **Manage** button.
4. Ensure that the Web server is started. The server status will be displayed, as shown in Figure 11-2. If the server is not running, click **Server On** to start the server.

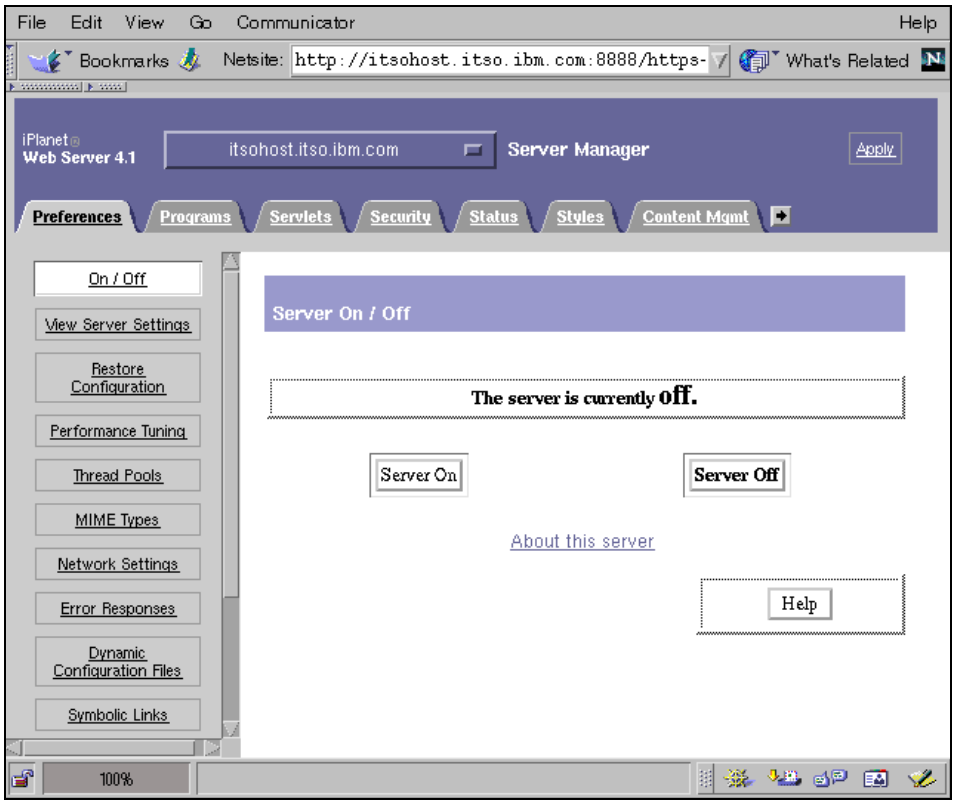


Figure 11-2 Set server status to On

5. If the server starts successfully you should see the message:

   ```
   The server is currently on.
   ```

Check the process status

To check the iPlanet Web Server process status, perform the following steps:

1. Check that the iPlanet Web Server processes are running by issuing the following command:

   ```
   ps -ef | grep ns-httpd
   ```

   The output should list one process for the iPlanet Web Server and one process for the iPlanet administration server.

2. Check that the Web server is registered to listen on port 80 and is therefore ready to handle requests:

   ```
   netstat -a | grep LISTEN | grep 80
   ```

Check request handling

To check iPlanet Web Server request handling, perform the following steps:

1. Using a Web browser, request the following URL representing the iPlanet Web Server home page:

   `http://<hostname.domain.com>/`

   The window shown in Figure 11-3 will be displayed if the iPlanet Web Server has been installed and configured correctly.

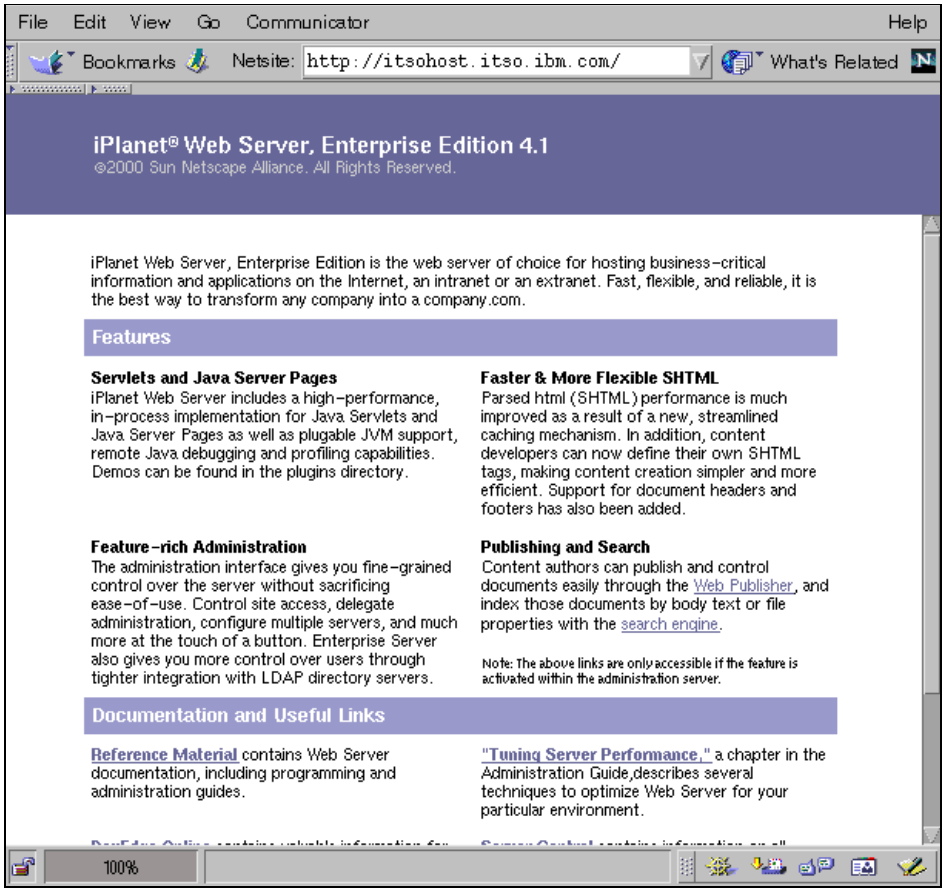


Figure 11-3 Home page request handled by Web server

11.4 Install the database server

This section provides detailed instructions for installing, configuring, and verifying Oracle 8i Release 3 (8.1.7), Enterprise Edition for Solaris.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install the Oracle 8i server.
3. Configure the Oracle 8i server.
4. Set up the WebSphere administrative database.

11.4.1 Preinstallation tasks

Prior to installing Oracle 8i Release 3 (8.1.7), Enterprise Edition for Solaris, the following checks and tasks need to be completed:

1. Create UNIX groups for Oracle8i.
2. Create a UNIX user account for Oracle8i.
3. Set environment variables.
4. Create mount points.
5. Verify system executables.
6. Configure the UNIX kernel for Oracle 8i.

Create UNIX groups for Oracle8i

Depending on the size and makeup of your organization, you may have several roles defined for database administration. The *Oracle 8i Installation Guide, Release 3 (8.1.7) for Sun SPARC, A85471* recommends that you create groups for the roles, as listed in Table 11-8

Table 11-8 Groups required by Oracle8i

Role of Oracle8i group	Suggested group name
Database administrator	dba
Oracle installer group	oinstall

In our example, we created a group called dba and oinstall as follows:

1. Log in as root and start a terminal session.
2. Create the groups by typing the following commands:

```
# groupadd dba
# groupadd oinstall
```

Create a UNIX user account for Oracle8i

To create a user named “oracle” and make the user a member of the dba group, complete the following steps:

1. Log in as root and start a terminal session.
2. Create the user oracle by typing the following command:

```
# useradd -d /export/home/oracle -m -g oinstall -G dba -s /bin/ksh oracle
```

Note: Syntax of the **useradd** command:

```
# useradd -d <path_home_dir> -m -g <primary_group> -G
<secondary_group> -s <path_shell> <username>
```

Where:

`-d <path_home_dir>` is the home directory of the user

`-m` creates the directory of the user and .profile

`-g` is the primary group to add user to

`-G` is the secondary group to add user to

`-s <path_shell>` is the path of the shell used by this user

`<username>` is the user name to create

3. Create a password for the user oracle by typing the following:

```
# passwd oracle
```

This will prompt the user for the password.

4. Verify login with user as follows:

```
# su - oracle
$ su - oracle
```

You will be prompted for your password.

Set environment variables

To set environment variables as part of the .profile of the oracle user shell, complete the following steps:

1. Log in as user oracle

```
# su - oracle
```

2. Add the following environment variables to the oracle user .profile:

```
umask 022
export ORACLE_BASE = /opt/oracle8/u01
export ORACLE_HOME = $ORACLE_BASE/app/oracle/product/8.1.7
export ORACLE_SID = <your_default_SID (created in subsequent step)>
export PATH=/usr/ccs/bin:$PATH:$ORACLE_HOME/bin:/etc:/usr/openwin/bin

# The following export is required for XWindows used by Oracle8i install.
export DISPLAY=<hostname>:0.0
```

Create mount points

To create mount points, complete the following steps:

1. Log in as root and start a terminal session.
2. Create the following directories:

   This directory is used for the Oracle8i software:

   ```
   # mkdir -p /opt/oracle8/u01
   ```

   The following directories are intended to be used for databases:

   ```
   # mkdir -p /opt/oracle8/u02
   # mkdir -p /opt/oracle8/u03
   # mkdir -p /opt/oracle8/u04
   ```

3. Change the owner of the directories to user oracle and group dba by typing the following command:

   ```
   # chown -R oracle:dba /opt/oracle8
   ```

4. Change permissions as follows to enable group write access:

   ```
   # chmod -R g+w /opt/oracle8
   ```

Verify system executables

This step verifies that the correct version of the system executables is being used.

1. Type the following commands to verify system executables:

   ```
   $ /usr/bin/which make
   $ /usr/bin/which ar
   $ /usr/bin/which ld
   $ /usr/bin/which rm
   ```

   The **which** command returns the executable path.

2. The path search order is determined by the location of the path in the PATH environment variable of the shell. Check that the directory returned for each executable is /usr/ccs/bin.
3. If this is not already in your path or not in the beginning of the path, you will need to add /usr/ccs/bin to the beginning of the PATH of the current shell.

Configure the UNIX kernel for Oracle8i

To ensure that Oracle8i has enough system memory we need to configure the UNIX kernel.

To configure the UNIX kernel for Oracle8i, complete the following steps:

1. Start a Solaris Console window, and log in as user root.
2. Back up the system file:

```
# cd /etc
# cp system system.bak
```

3. Add the following lines to bottom of the system file:

```
set shmsys:shminfo_shmmax=4294967295
set shmsys:shminfo_shmmin=1
set shmsys:shminfo_shmmni=100
set shmsys:shminfo_shmseg=10
set semsys:seminfo_semmni=100
set semsys:seminfo_semmsl=1000
set semsys:seminfo_semmns=2000
set semsys:seminfo_semopm=100
set semsys:seminfo_semvmx=32767
```

4. Type the following command to shut down and reboot:

```
# /usr/sbin/shutdown -i6 -g0 -y
```

11.4.2 Install Oracle 8i server

In order to install Oracle 8i Release 3 (8.1.7), Enterprise Edition for Solaris, perform the following steps:

1. Log in as root and start a terminal session.
2. Disable access control for X-Windows display by typing the following:

```
# xhost +
```

3. Change to user oracle:

```
# su - oracle
```

4. Insert the Oracle8i Enterprise Edition Release 3 (8.1.7) for Sun SPARC Solaris CD-ROM in the database server CD-ROM drive. The Volume Management daemon automatically mounts the CD under /cdrom/cdrom0.
5. Start the Oracle8i install type the following:

```
# cd /cdrom/cdrom0
# ./runInstaller
```

Base installation

6. In the Oracle8i Welcome window, click **Next**.
7. In the File Locations window, verify that the paths are as follows and then click **Next**:
   - Source: /cdrom/oracle8i/stage/products.jar
   - Destination: /opt/oracle8/u01/app/oracle/product/8.1.7

Note: The destination should be the path specified as the $ORACLE_HOME environment variable.

8. In the UNIX Group Name window, enter the group oinstall defined in “Create UNIX groups for Oracle8i” on page 350, and then click **Next**.
9. In the Oracle Universal Installer window, you will be prompted to run a script as user root in another Solaris Console window.
   a. Start a Solaris Console window and log in as root.
   b. Execute the script by typing the command:

      ```
      # /opt/oracle8/u01/app/oracle/product/8.1.7/orainstRoot.sh
      ```

   c. You should see a message that states the following:

      ```
      Creating Oracle Inventory pointer file (/var/opt/oracle/oraInst.loc).
      Changing groupname of /opt/oracle8/u01/app/oracle/oraInventory to
      oinstall.
      ```

10. Click the Oracle Universal Install window to bring it back to the foreground, and then click **Retry**.
11. In the Available Products window, select **Oracle8i Enterprise Edition 8.1.7.0.0**, and then click **Next**.
12. In the Installation Types window, select **Custom** and then click **Next**.
13. In the Available Product Components window, do the following:
   – Under Oracle Product Options 8.1.7.0.0, deselect **Legato Storage Manager**
   – Deselect **Oracle Product Options 8.1.7.0.0**
   – Deselect **Development Tools 8.1.7.0.0**
   – Deselect **Oracle HTTP Server 1.3.12.0.1a**
   – Click **Oracle Java Products 8.1.7.0.0 -> Oracle JDBC drivers** and select all the available drivers:
      • Select **Oracle JDBC/OCI Driver for JDK 1.1 8 1.7.0.0**
      • Select **Oracle JDBC/OCI Driver for JDK 1.2 8.1.7.0.0**
      • Select **Oracle JDBC Thin Driver for JDK 1.1 8 1.7.0.0**
      • Select **Oracle JDBC Thin Driver for JDK 1.2 8.1.7.0.0**
   – Click **Next**.

   You should see the status bar indicating that the components are loading.
14. In the Component Locations window, accept the defaults and click **Next**.

You should see the status bar indicating that the components are loading.

15. In the Privileged Operating System Groups window, accept the defaults and click **Next**:
    - Database Administrator (OSDBA) Group = dba
    - Database Operator (OSOPER) Group = dba

    You should see the status bar indicating that the components are loading.

16. In the Create Database window, select **No** and then click **Next**.

> **Note:** We will create a database for WebSphere in subsequent steps via the Database Configuration Assistant.

17. In the Summary window, click **Install** to start the installation.

    The installation will take approximately 30 minutes to complete.

> **Tip:** After 85% of the installation has been completed, the installer program will prompt you to load CD #2 without exiting the installation:
>
> a. Start a terminal session as root.
>
> b. Unmount the CD-ROM:
>
> ```
> # cd /
> # umount /cdrom/cdrom0
> ```
>
> c. Eject the CD-ROM using the Solaris CDE File Manager utility.
>
> d. Load CD #2 into CD-ROM drive.
>
> e. The Solaris Volume Management daemon automatically mounts the CD-ROM under /cdrom/cdrom0.

18. In the Setup Privileges window, a message will be displayed to run the following script:
    a. Start a Solaris Console window, log in as root.
    b. Execute the script by typing:

    ```
    # /opt/oracle8/u01/app/oracle/product/8.1.7/root.sh
    ```

    c. You will be prompted to enter the full path to the local bin directory (we entered `/usr/bin`) then press Enter.
    d. Click the Setup Privileges window to bring it to the foreground, then click **OK**.

Net8 configuration

19. In the Net8 Configuration Assistant Welcome window, click **Next**.
20. In the Net8 Configuration Assistant: Directory Service Access window, click **No, I want to defer directory service access configuration to another time**, then click **Next**.
21. In the Net8 Configuration Assistant: Listener Configuration Listener Name window, accept the default (LISTENER) and click **Next**.
22. In the Net8 Configuration Assistant: Listener Configuration Select Protocols window, accept the default (TCP in selected protocols) and click **Next**.
23. In the Net8 Configuration Assistant: Listener Configuration TCP/IP Protocol window, accept the default (use the standard port number of 1521) and click **Next**.
24. In the Net8 Configuration Assistant: Listener Configuration More Listeners window, accept the default (No) and click **Next**.
25. In the Net8 Configuration Assistant: Listener Configuration Done window, click **Next**.
26. In the Net8 Configuration Assistant: Naming Methods Configuration window, select **No, I do not want to change the naming methods configured**, then click **Next**.
27. In the Net8 Configuration Assistant Done window, click **Finish**.
28. In the End of Installation window, click **Exit**.
29. The Oracle8i Enterprise Edition server installation is now complete.

11.4.3 Configure Oracle 8i server

After the Oracle8i installation, we need to perform the following configuration steps:

1. Update the oracle user .profile.
2. Net8 configuration.

Update the oracle user .profile

In this section we will add Oracle8i required environment variables to the oracle user .profile that we created by completing the following steps:

1. Start a Solaris Console window, and log in as the oracle user:

```
# su - oracle
```

2. Add the following entries to the end of the oracle user .profile:

```
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
```

```
export
CLASSPATH=$ORACLE_HOME/JRE:$ORACLE_HOME/jlib:$ORACLE_HOME/product/jlib

ORACLE_SID=was
ORAENV_ASK=NO
. /usr/bin/oraenv
```

Net8 configuration

This section describes the required configuration for Net8 after the Oracle8i installation.

1. Start a Solaris Console window, and log in as root.
2. Add the listener to the /etc/services file. The listener name and port were defined during the Oracle8i installation (Net8).

   We add the following line to our services file:

   ```
   LISTENER 1521/tcp
   ```

3. Save the services file.

   Note: The /etc/services file is symbolically linked to /etc/inet/services. When saving the file with vi use the **`vi w!`** command to override default write permissions.

4. Check the status of the listener.

   ```
   # su - oracle
   $ lsnrctl status LISTENER
   ```

   Where `LISTENER` is the name of your listener. The output should indicate that the listener is on PORT=1521, PROTOCOL = TCP.

11.4.4 Set up the WebSphere administrative database

Next, set up a database in Oracle to use as the WebSphere administrative repository. The database will be populated with WebSphere schema and default values in a later task.

To set up the WebSphere database, complete the following steps:

This section is organized as follows:

1. Create the WebSphere database.
2. Verify the WebSphere database.
3. Configure the WebSphere database to autostart.
4. Tune the WebSphere database.

5. Create the WebSphere Oracle user ID and tablespace.
6. Verify user creation.
7. Configure Net8 on the Oracle 8i server.

Create the WebSphere database

We will create the following Oracle8i database in preparation for the WebSphere installation. To create an Oracle8i database, complete the following steps on the Oracle8i Enterprise Edition Server system:

1. Log in as root and start a terminal session.
2. Change to user oracle:

   ```
   # su - oracle
   ```

3. Run the Oracle8i Database Configuration Assistant:

   ```
   $ dbassist &
   ```

4. In the Oracle Database Configuration Assistant Welcome window, select **Create a Database**, then click **Next**.
5. Select **Custom** as the database type, then click **Next**.
6. Select **Multipurpose** as the application type, then click **Next**.
7. Enter the number of concurrently connected users. We accepted the default (15), then clicked **Next**.
8. Select **Dedicated Server Mode** as the server mode, then click **Next**.
9. In the Select Options window, deselect all options except **SQL*Plus help** (optional), and then click **Next**.
10. In the Database Information window, enter the following:
    - Global Database Name: was
    - SID: was
    - Initialization Filename: /opt/oracle8/u01/app/oracle/admin/was/pfile/initwas.ora
    - Compatible Parameter: 8.1.0
    - Click **Change Character Set**:
      i. Select the **Character Set** pull-down and select **UTF8**.
      ii. Select the **National Character Set** pull-down and select **UTF8**.
      iii. Click **OK**.

> **Note:** WebSphere does not require that the character set be changed to UTF8. We select UTF8 for its globalization support.

– Click **Next**.

11. In the Review Control File Parameter Info window, enter the paths listed in Table 11-9, then click **Next**.

Table 11-9 Required paths for WAS database control files

Control file	Required path...
Control file 1	/opt/oracle8/u02/oradata/was/control01.ctl
Control file 2	/opt/oracle8/u02/oradata/was/control01.ctl
Control file 3	/opt/oracle8/u02/oradata/was/control01.ctl

12. In the Review Tablespace window, there are a number of tablespaces on each of the tabbed sections of the window. Enter the paths listed in Table 11-10, then click **Next**.

Table 11-10 Required paths for WAS database tablespaces

Tablespace	Required path...
SYSTEM	/opt/oracle8/u02/oradata/was/system01.dbf
TOOLS	/opt/oracle8/u02/oradata/was/tools01.dbf
USERS	/opt/oracle8/u02/oradata/was/users01.dbf
ROLLBACK	/opt/oracle8/u02/oradata/was/rbso1.dbf
INDEX	/opt/oracle8/u02/oradata/was/indx01.dbf
TEMP	/opt/oracle8/u02/oradata/was/temp01.dbf

13. In the Redo Log File Parameter Information window, enter the details listed in Table 11-11, then click **Next**.

Table 11-11 Required paths for WAS database redo files

Control file	Required path...
Redo file 1	/opt/oracle8/u02/oradata/was/redo01.ctl
Redo file 2	/opt/oracle8/u02/oradata/was/redo02.ctl
Redo file 3	/opt/oracle8/u02/oradata/was/redo03.ctl

14. In the Logging Parameter Information window, accept the defaults and click **Next**.

15. In the SGA Parameter Information window, accept the defaults and click **Next** (tuning will be done in a later task).

16. In the Trace Files Directory Path window, accept the defaults and click **Next**.

17. Select **Create database now**, and click **Finish**.

18. In the Oracle Database Configuration Assistant alert, asking if you want to proceed, click **Yes**.

    The database creation progress indicator should be visible. This process takes approximately 20 minutes.

Important: If an error alert occurs during the database creation process, review the log located in the $ORACLE_BASE/admin/was/create directory to determine the cause of the error.

A common error is “unable to open communication channel”, often indicating that the Solaris kernel parameters have been incorrectly configured to support the database settings selected in dbassist. For example, if the number of processes is too high, you may need to increase the SEMMSL and SEMMNS semaphore settings in /etc/system.

Verify the WebSphere database

After the installation, we recommend that you take some steps to verify that the Oracle8i Enterprise Edition server is installed correctly before attempting to connect to it from WebSphere.

To verify the Oracle8i Enterprise Edition, complete the following steps:

1. Start a Solaris Console window, and log in as user oracle.

   ```
   # su - oracle
   ```

2. To open an Oracle SQL*Plus window, type the following:

   ```
   $ sqlplus system/manager@was
   SQL> quit
   ```

3. If the previous step succeeds, then the Oracle 8i database is correctly configured to support TCP access.

Configure the WebSphere database to autostart

The following procedure will automate the startup and shutdown of the Oracle8i database server.

1. Edit the /var/opt/oracle/oratab file.

   Database entries in the oratab file appear in the following format:

   ```
   ORACLE_SID:ORACLE_HOME:{Y|N}
   ```

   Where Y or N specifies whether you want the dbstart and dbshut scripts to start up and shut down the database. Find the entries for all the databases that you want to start up. They are identified by the `SID` in the first field. Change the last field for each to Y to enable the database to start when the **`dbstart`** command is run.

For example:

```
was:/opt/oracle8/u01/app/oracle/product/8.1.7:Y
```

2. Create the system autostart script.

   Refer to *Oracle 8i Installation Guide, Release 3 (8.1.7) for Sun SPARC, A85471* for more detailed instructions concerning configuration of database autostart.

   Create a script named dbora in the /etc/init.d directory:

   a. Change to the /etc/init.d directory.

      ```
      # cd /etc/init.d
      ```

   b. Create a script called dbora.

      Example 11-2 shows a sample dbora script.

Example 11-2 Sample dbora script

```
#!/bin/sh

ORA_HOME=/opt/oracle8/u01/app/oracle/product/8.1.7
ORA_OWNER=oracle

if [ ! -f $ORA_HOME/bin/dbstart ]
then
   echo "Oracle startup: cannot find dbstart file... cannot start"
   exit
fi

case "$1" in
'start')
   # startup the ORA listener daemon
   su - $ORA_OWNER -c '/opt/oracle8/u01/app/oracle/product/8.1.7/bin/lsnrctl
start'

   # startup each ORA instance listed in /var/opt/oracle/oratab
   su - $ORA_OWNER -c $ORA_HOME/bin/dbstart
   ;;

'stop')
   # shutdown each ORA instance listed in /var/opt/oracle/oratab
   su - $ORA_OWNER -c $ORA_HOME/bin/dbshut

   # shutdown the ORA listener daemon
   su - $ORA_OWNER -c '/opt/oracle8/u01/app/oracle/product/8.1.7/bin/lsnrctl
stop'
   ;;
esac
```

c. Link the dbora script as follows:

```
# ln -s /etc/init.d/dbora /etc/rc0.d/K10dbora
# ln -s /etc/init.d/dbora /etc/rc2.d/S99dbora
```

Tune the WebSphere database

The following procedure will tune the Oracle8i database to suite WebSphere.

1. Log in as root and start a terminal session.
2. Change to user oracle:

   ```
   # su - oracle
   ```

3. Modify the parameters listed in Table 11-12 of the $ORACLE_HOME/dbs/init<your_SID>.ora database initialization file.

Table 11-12 Oracle8i database tuning parameters

Parameters	Required value
open_cursors	300
db_block_buffers	2500
shared_pool_size	20971520
processes	100

4. Stop and start the Oracle8i server for the database tuning changes to take effect.

   ```
   $ dbshut
   $ dbstart
   ```

   Note: The Oracle8i dbshut and dbstart scripts will stop and start all databases. This behavior is desired in this example, but may not be appropriate in production situations.

   Using **svrmgrl** is another option for stopping and starting Oracle databases.

5. If your Oracle8i Server database instances do not start, check the tuning parameters entered for errors. Changing the tuning values beyond the physical memory available on the system will result in the database instance not starting.

Create WebSphere Oracle user ID and tablespace

To create a WebSphere Oracle user ID and tablespace, complete the following steps:

1. Start a Solaris Console window on the database server system, and log in as user oracle (Oracle DBA).
2. Start a SQL*Plus session by typing the following command:

   ```
   $ sqlplus system/<system_password>@<was_SID>
   ```

   For example:

   ```
   $ sqlplus system/manager@was
   ```

3. Create the WebSphere tablespace, by typing the following command:

   ```
   SQL> CREATE TABLESPACE was
   DATAFILE '%ORACLE_HOME%/dbs/initwas.ora'
   SIZE 4M
   REUSE
   AUTOEXTEND ON NEXT 2M
   MAXSIZE UNLIMITED;
   ```

4. Create the WebSphere user ID ejsadmin by typing the following command:

   ```
   SQL> CREATE USER ejsadmin
   IDENTIFIED BY ejsadmin
   DEFAULT TABLESPACE was
   QUOTA UNLIMITED ON was;
   ```

5. Grant privileges to the WebSphere Oracle user ejsadmin by typing the following commands:

   ```
   SQL> GRANT dba TO ejsadmin;
   SQL> ALTER USER ejsadmin TEMPORARY TABLESPACE temp;
   ```

   Tip: If you don't want ejsadmin to have dba authority, issue the following commands instead:

   ```
   SQL> GRANT connect,resource TO ejsadmin;
   SQL> ALTER USER ejsadmin TEMPORARY TABLESPACE temp;
   ```

6. Create the WebSphere user ID ejb by typing the following command:

   ```
   SQL> CREATE USER ejb
   IDENTIFIED BY ejb
   DEFAULT TABLESPACE was
   QUOTA UNLIMITED ON was;
   ```

7. Grant privileges to the WebSphere Oracle user ejb by typing the following commands:

   ```
   SQL> GRANT dba TO ejb;
   SQL> ALTER USER ejb TEMPORARY TABLESPACE temp;
   ```

Tip: If you don't want ejb to have dba authority, issue the following commands instead:

```
SQL> GRANT connect,resource TO ejb;
SQL> ALTER USER ejb TEMPORARY TABLESPACE temp;
```

Verify user creation

To verify the users have been created correctly, log on to SQL*Plus with each of the user IDs as follows:

1. Log in as user oracle:

   ```
   # su - oracle
   ```

2. Start SQL*Plus using ejsadmin:

   ```
   $ sqlplus ejsadmin/ejsadmin@was
   SQL> quit
   ```

3. Start SQL*Plus using ejb:

   ```
   $ sqlplus ejb/ejb@was
   SQL> quit
   ```

Configure Net8 on Oracle8i server

To configure Net8 on the Oracle8i Enterprise Edition database server, complete the following steps:

1. Log in as root and start a terminal session.
2. Disable access control for X-Windows display by typing the following:

   ```
   # xhost +
   ```

3. Change to user oracle:

   ```
   # su - oracle
   ```

4. Start Net8 Assistant:

   ```
   # netasst &
   ```

5. Verify that a Service Name entry exists for the following:
   - Service Name: was
   - Protocol: TCP/IP
   - Host Name: <database server hostname>
   - Port Number: <listener_port>
6. Check the status of the Oracle listener:

   ```
   $ lsnrctl status <listener_name>
   ```

For example:

```
$ lsnrctl status LISTENER
```

Note: The status of all listeners will be displayed if the listener name is omitted from the above command.

7. If the listener is not started, start it as follows:

   ```
   $ lsnrctl start <listener_name>
   ```

   For example:

   ```
   $ lsnrctl start LISTENER
   ```

   Note: All listeners will be started if the listener name is omitted.

11.5 Install the database client

This section provides detailed instructions for installing, configuring, and verifying Oracle 8i Release 3 (8.1.7) Client.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install Oracle 8i client.
3. Configure Oracle 8i client.
4. Set up access to remote WebSphere administrative database.

11.5.1 Preinstallation tasks

Prior to installing the Oracle 8i client, the following tasks must be performed:

1. Create UNIX groups for Oracle 8i.
2. Create UNIX user account for Oracle 8i.
3. Set up environment variables.

See 11.4.1, “Preinstallation tasks” on page 350 for a description of each of these tasks.

11.5.2 Install the Oracle 8i client

To install the Oracle 8i client, perform the following steps:

1. Log in as root and start a terminal session.

2. Disable access control for X-Windows display by typing the following:

```
# xhost +
```

3. Change to user oracle:

```
# su - oracle
```

4. Insert the Oracle8i Enterprise Edition Release 3 (8.1.7) for Sun SPARC Solaris CD-ROM in the database server CD-ROM drive. The Volume Management daemon automatically mounts the CD under /cdrom/cdrom0.
5. Start the Oracle8i install type the following:

```
# cd /cdrom/cdrom0
# ./runInstaller
```

6. In the Oracle8i Welcome window, click **Next**.

Base installation

7. In the File Locations window, verify that the paths are as follows, then click **Next**.
   - Source: /cdrom/oracle8i/stage/products.jar
   - Destination: /opt/oracle8/u01/app/oracle/product/8.1.7

   Note: The destination should be the path specified by the ORACLE_HOME.environment variable.

8. In the UNIX Group Name window, enter the group defined in 11.5.1, “Preinstallation tasks” on page 365. For example, enter `oinstall` and then click **Next**.
9. In the Oracle Universal Installer window, you will be prompted to run a script as user root in another Solaris Console window.
   a. Start a Solaris Console window and log in as root.
   b. Execute the script by typing the following:

      ```
      # /opt/oracle8/u01/app/oracle/product/8.1.7/orainstRoot.sh
      ```

   c. You should see a message that states the following:

      ```
      Creating Oracle Inventory pointer file (/var/opt/oracle/oraInst.loc).
      Changing groupname of /opt/oracle8/u01/oraInventory to oinstall.
      ```

10. Click the Oracle Universal Install window to bring it back to the foreground, then click **Retry**.
11. In the Available Products window, select **Oracle8i Client 8.1.7.0.0**, then click **Next**.
12. In the Installation Types window, select **Custom**, then click **Next**.

13. When the Available Product Components window appears, do the following:
    - Deselect all options except the following:
      - Oracle Java Products 8.1.7.0.0
      - Oracle Utilities 8.1.7.0.0
      - Net8 Products 8.1.7.0.0
    - Under Oracle Java Products 8.1.7.0.0 > Oracle JDBC drivers, select all the available drivers:
      - Select **Oracle JDBC/OCI Driver for JDK 1.1 8 1.7.0.0**
      - Select **Oracle JDBC/OCI Driver for JDK 1.2 8.1.7.0.0**
      - Select **Oracle JDBC Thin Driver for JDK 1.1 8 1.7.0.0**
      - Select **Oracle JDBC Thin Driver for JDK 1.2 8.1.7.0.0**
    - Click **Next**.

    You should see the status bar indicating that the components are loading.
14. In the Component Locations window, accept the default and click **Next**.

    You should see the status bar indicating that the components are loading.
15. In the Oracle Product Support window, choose only the default protocol (TCP) for database access, then click **Next**.
16. When the Summary window appears, review the summary information, and then click **Install**.

    The installation will take approximately 10 minutes to complete.
17. When the Setup Privileges window appears, a message will be displayed to run the following script:
    a. Start a Solaris Console window and log in as root.
    b. Execute the script by typing:

       ```
       # /opt/oracle8/u01/app/oracle/product/8.1.7/root.sh
       ```
    c. You will be prompted to enter the full path to the local bin directory (we entered `/usr/bin`) and then press Enter.
    d. Click the Setup Privileges window to bring it to the foreground, and then click **OK**.

Note: The base installation is now complete. The Oracle8i Client install will automatically continue in the next section.

Net8 configuration

18. In the Net8 Configuration Assistant Welcome window, click **Next**.

19. In the Net8 Configuration Assistant: Directory Service Access window, click **No, I want to defer directory service access configuration to another type**, then click **Next**.
20. When the Net8 Configuration Assistant: Naming Methods Configuration window appears, accept the default (local) and then click **Next**.
21. In the Net8 Configuration Assistant: Net Service Name Configuration, Database Version window, accept the default (Oracle8i database or service) and then click **Next**.
22. In the Net8 Configuration Assistant: Net Service Name Configuration, Service Name window, enter the following:
    - Service Name: <global_database_name> (defined in the Oracle8i Enterprise Edition Database Server install).

      For example, we entered: was
    - Click **Next**.
23. In the Net8 Configuration Assistant: Net Service Name Configuration, open the Protocols window, select **TCP** and then click **Next**.
24. In the Net8 Configuration Assistant: Net Service Name Configuration, TCP/IP Protocol window, do the following:
    - Host Name: <database_server_fully_qualified_hostname>

      For example, we entered: itsohost.itso.ibm.com
    - Select **Use the standard port number of 1521**

      Note: If you specified a different port from the default during the Oracle8i server install, use the port you defined.

    - Click **Next**.
25. In the Net8 Configuration Assistant: Net Service Name Configuration, Test window, select **Yes, perform a test**.
26. In the Net8 Configuration Assistant: Net Service Name Configuration, Connecting window, you will see an error message stating that the test did not succeed. A default user ID of *Scott* is defined.
    a. Click **Change Login**, and enter the following:
       - Username: system
       - Password: manager

    b. Click **OK**.

27. The test will occur automatically after clicking OK to change the login. You should see a message stating the following:

    ```
    Connecting... Test successful.
    ```

    When done, click **Next**.

28. In the Net8 Configuration Assistant: Net Service Name Configuration, Net Service Name window, enter the following:

    – Net Service Name: <global_database_name>.<domain>

      For example, was.itso.ibm.com

    – Click **Next**.

29. In the Net8 Configuration Assistant: Net Service Name Configuration, Another Net Service Name window, select **Yes**, then click **Next**.

30. In the Net8 Configuration Assistant: Net Service Name Configuration Done window, click **Next**.

31. In the Net8 Configuration Assistant: Naming Methods Configuration Done window, click **Next**.

32. In the Net8 Configuration Assistant Done window, click **Finish**.

33. In the End of Installation window, click **Exit**.

34. The Oracle8i Client install is now complete.

11.5.3 Configure the Oracle 8i client

After the Oracle8i Client installation, we need to perform the following configuration steps:

1. Update the oracle user .profile.

Update the oracle user .profile

Add Oracle8i required environment variables to the oracle user .profile by completing the following steps:

1. Log in as root and start a terminal session.
2. Change to user oracle:

   ```
   # su - oracle
   ```

3. Add the following entries to the end of the oracle account's .profile file:

   ```
   export LD_LIBRARY_PATH=$ORACLE_HOME/lib
   export CLASSPATH=$ORACLE_HOME/JRE:$ORACLE_HOME/jlib:$ORACLE_HOME/product/jlib
   ```

11.5.4 Verify access to the remote WebSphere administrative database

After the installation, we recommend that you take some steps to verify that the Oracle8i client is installed correctly. To verify the Oracle8i client, complete the following steps:

1. Log in as root and start a terminal session.
1. Change to user oracle:

   ```
   # su - oracle
   ```

2. To start a SQL*Plus command window, type the following:

   ```
   $ sqlplus system/manager@was.itso.ral.ibm.com
   SQL> quit
   ```

3. If the previous step succeeds, then the Oracle 8i client is correctly configured for TCP access of remote Oracle 8i servers.

11.6 Install WebSphere Application Server

This section provides detailed instructions for installing, configuring, and verifying WebSphere Application Server V4.0, Advanced Edition for Solaris.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install WebSphere.
3. Verify the WebSphere installation.

11.6.1 Preinstallation tasks

Prior to installing IBM WebSphere Application Server V4.0, the following checks and tasks need to be completed on the WebSphere server machine:

1. Check that IP ports are unused.
2. Stop Web server processes.
3. Start the Oracle 8i server and listener.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing active services that use the following IP ports on the server:

– 900 (bootstrap port)
– 9000 (Location Service Daemon)
– 9080 (default application server)

We suggest using the following command for this task:

```
$ netstat -a | grep LISTEN
```

Stop Web server processes

The iPlanet Web Server process must be stopped while WebSphere is installed. The WebSphere installation changes the iPlanet configuration file as part of the Web server plug-in component installation.

1. Log in as root on the iPlanet Web Server machine.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <http_server_install_path>/https-<ServerName>
# ./stop
```

For example:

```
# cd /opt/netscape/server4/https-itsohost.itso.ibm.com
# ./stop
```

Start Oracle 8i server and listener

The Oracle 8i server and listener processes must be started before proceeding with the WebSphere Application Server installation.

See 11.4, “Install the database server” on page 349 for details.

11.6.2 Install WebSphere

To install IBM WebSphere Application Server V4.0, Advanced Edition using the GUI installer interface, complete the following steps on the WebSphere server machine:

Tip: The WebSphere installer (install.sh) also provides a non-GUI scripted or “silent” mode of operation. See 11.9, “Install the WebSphere Application Server - silent mode” on page 395 for details.

1. Log in as root.
2. Start a terminal session.

3. Load the IBM WebSphere Application Server V4.0 CD-ROM into the CD-ROM drive. The CDE will automatically mount the CD under /cdrom/cdrom0.
4. Change the directory to the installation root:

```
# cd /cdrom/cdrom0
```

5. Ensure the DISPLAY and TERM environment variables are properly set.
6. Run the install.sh installation script:

```
# ./install.sh
```

7. In the Welcome window, click **Next**.

Important: If prerequisite checking is enabled for UNIX, then an alert may be displayed indicating that some of the installation prerequisites have not been met. This can even occur if newer patches or packages than those listed in prereq.properties are installed. If such an alert is displayed, recheck all prerequisites, and if they are met or exceeded, perform the following steps:

- Exit the installation.
- Disable prerequisite checking.
- Restart the installation from the beginning.

8. In the Installation Options window, select **Custom Installation** and click **Next**.
9. In the Choose Application Server Components window, shown in Figure 11-4, choose all options except IBM HTTP Server, and click **Next**.

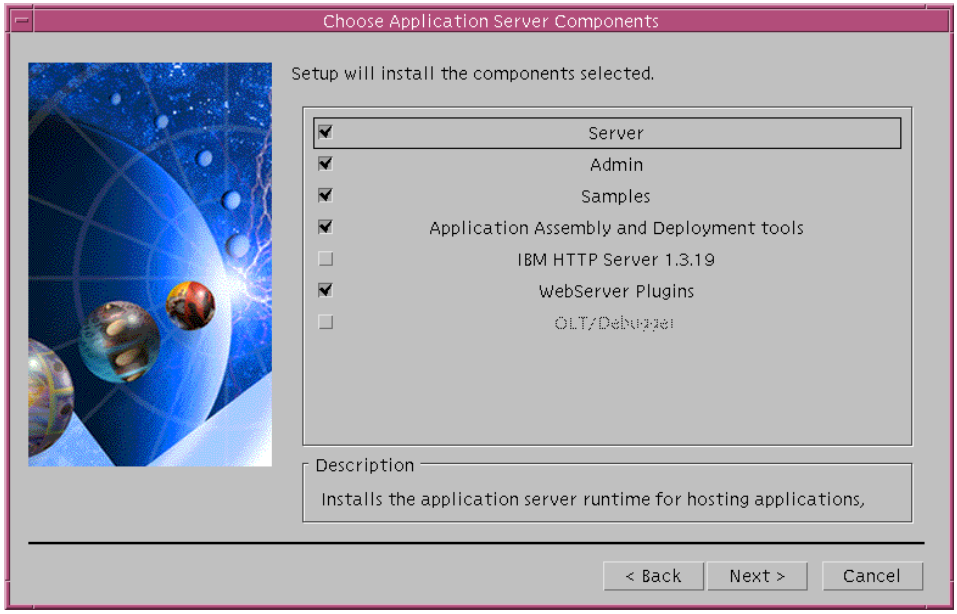


Figure 11-4 Select components to install

Important: Although not listed in the Application Server Components window, the Sun JDK 1.3.0 is automatically installed under the WebSphere installation directory. There is no need to separately install a JDK for use by:

- WebSphere Application Server
- Web server plug-ins

10. In the Choose Web server plug-in window, shown in Figure 11-5, choose only the **IBM HTTP Server plug-in**, and click **Next**.

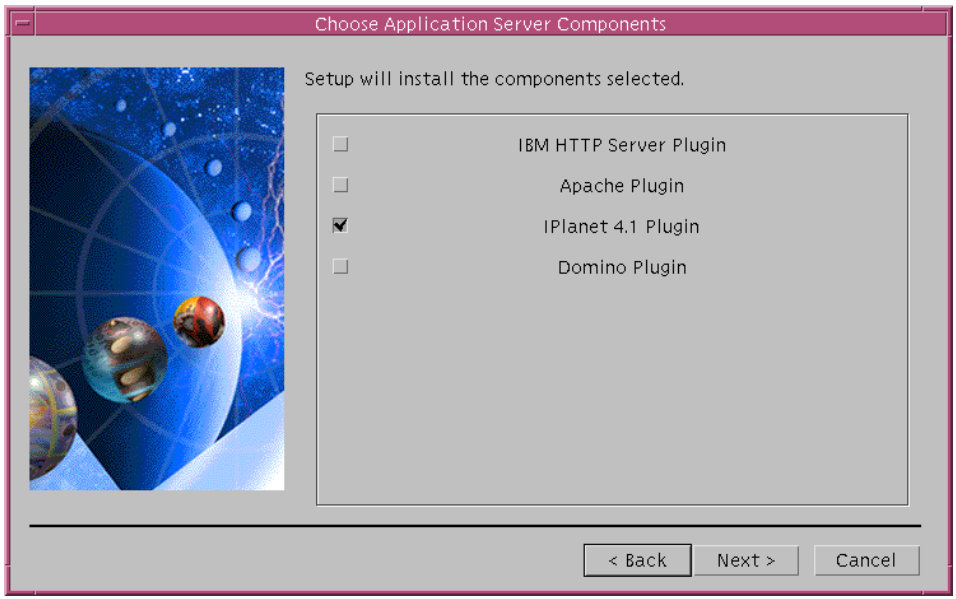


Figure 11-5 Select iPlanet Web Server plug-in

11. In the Database Options window, shown in Figure 11-6, enter the following then click **Next**:
    - Database Type: Oracle
    - Database Name (Database SID): <database_SID>

      Enter the Oracle SID of the WebSphere administrative database, as created in 11.4.4, “Set up the WebSphere administrative database” on page 357.
    - DB Home: <value of $ORACLE_HOME>

      In our test environment we used a value of /opt/oracle8/u01/app/oracle/product/8.1.7
    - DB URL: `jdbc:oracle:thin:@<hostname.domain.com>:<listener_port>:<database_SID>`

      Note: You should not have to edit the DB URL field. The value is dynamically built up from the values entered into the other fields.
    - Server Name: <hostname.domain.com>
    - Port Number: <listener_port>

In our test environment we used a value of 1521

- Database User ID: ejsadmin
- Database Password: ejsadmin
- Remote DB: leave unselected

Note: Whether the Oracle database is local or remote, in our test environment we configure the Oracle client to access the database through a TNS alias. Under these conditions, the Remote DB setting should not be selected.

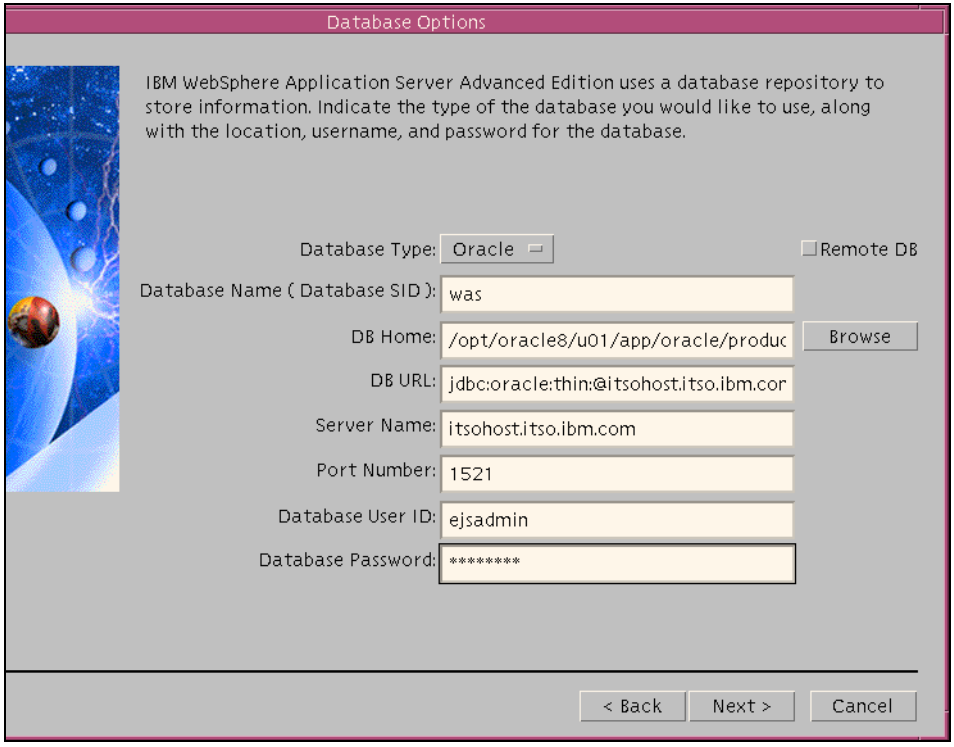


Figure 11-6 Specify Oracle 8i database connection settings

12. In the Select Destination Directory window, accept the default location for the WebSphere Application Server (/opt/WebSphere/AppServer). Click **Next** to continue.
13. In the Install Options Selected window, shown in Figure 11-7, check that the correct components have been selected. If yes, click **Install** to start the installation.

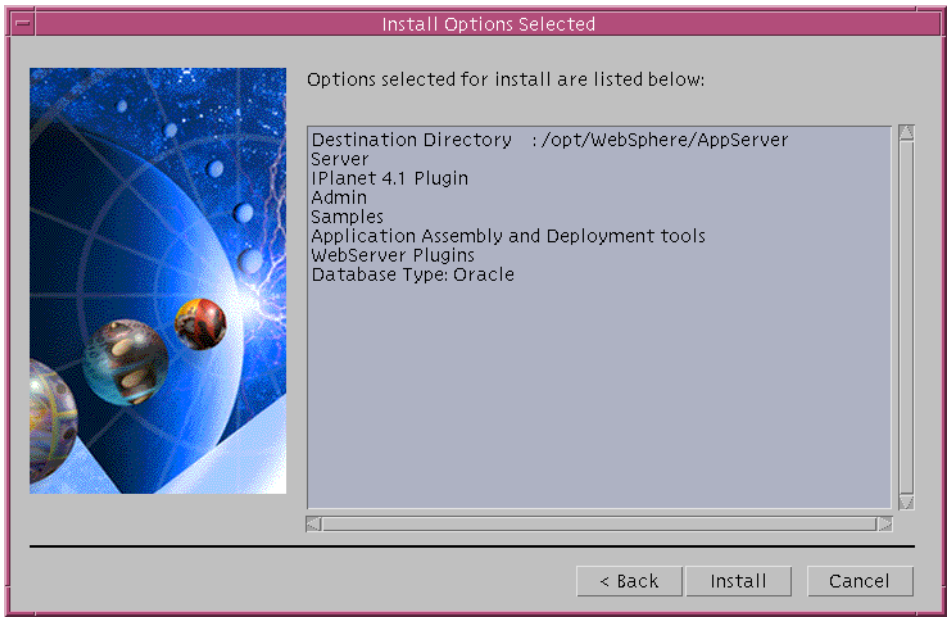


Figure 11-7 Components required for WebSphere installation

14. In the Location of Configuration Files window, shown in Figure 11-8, enter the path to the iPlanet Web Server configuration file (obj.conf), then click **Next**.

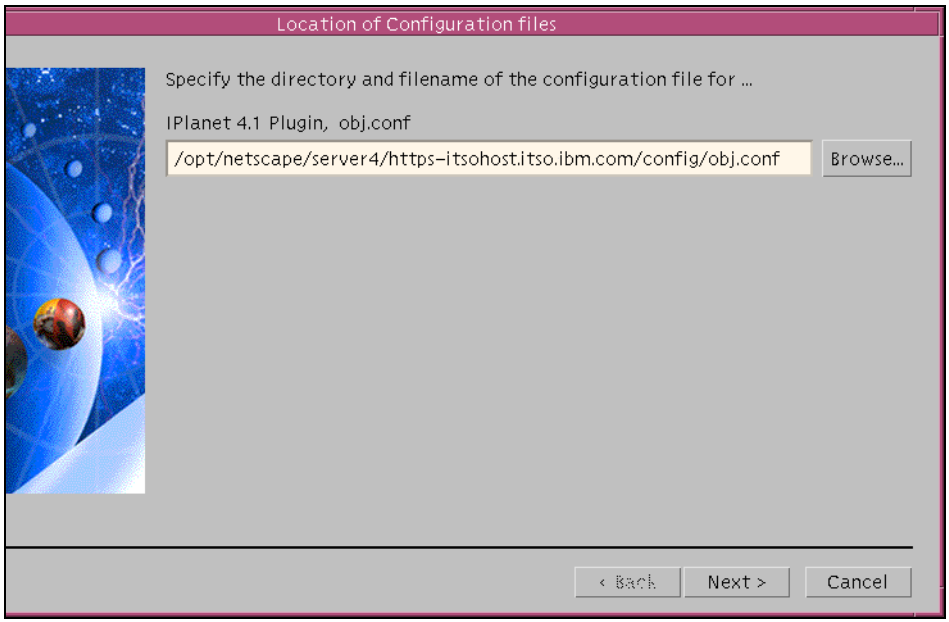


Figure 11-8 Specify iPlanet Web Server configuration file location

15. In the Setup Complete window, click **Finish**. The installation of the WebSphere Application Server is now complete.

11.6.3 Verify the WebSphere installation

In order to verify the installation of IBM WebSphere Application Server V4.0, Advanced Edition, the following tasks must be completed in order:

1. Check the installation log.
2. Check the admin.config settings.
3. Check the Web server configuration file changes.
4. Start the WebSphere administrative server processes.
5. Start the WebSphere Default Server.
6. Regenerate the Web server plug-in settings.
7. Restart the Web server processes.
8. Verify the Web server plug-in configuration.
9. Create the WebSphere system startup script.

Check the installation log

Check that the installation log <WAS_HOME>/logs/install.log does not contain any errors.

Check the admin.config settings

To check the admin.config settings, perform the following steps:

1. Check that the repository database settings are correct for the database type (Oracle), instance (<database_SID>) and user ID (ejsadmin) used in our test environment:

```
com.ibm.ejs.sm.adminServer.dbdataSourceClassName=oracle.jdbc.pool.OracleCon
nectionPoolDataSource
com.ibm.ejs.sm.adminServer.dbserverName=<hostname.domain.com>
com.ibm.ejs.sm.adminServer.dbportNumber=<listener_port>
com.ibm.ejs.sm.adminServer.dbdatabaseName=<database_SID>
com.ibm.ejs.sm.adminServer.dbuser=ejsadmin
com.ibm.ejs.sm.adminServer.dbpassword=ejsadmin
com.ibm.ejs.sm.adminServer.dbdisable2Phase=true
com.ibm.ejs.sm.adminServer.dbURL=jdbc:oracle:thin:@<hostname.domain.com>:<l
istener_port>:<database_SID>
```

Important: If a user ID other than EJSADMIN is used to access the WebSphere administrative database hosted in Oracle, then ensure that the user ID and schema lines reflect your actual database user ID and password:

```
com.ibm.ejs.sm.adminServer.dbuser=your_user_name
com.ibm.ejs.sm.adminServer.dbSchema=your_user_name
```

2. Check the path-related parameter shown in Table 11-13.

Table 11-13 Oracle-related path required in admin.config

Parameter	Must contain paths...
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs	<oracle_home>/jdbc/lib

3. Check that the WebSphere schema and initial configuration (for example, Default Server) will be written to the repository database on startup, as listed in Table 11-14.

Table 11-14 Required schema creation and initial configuration flags

Parameter	Required value...
com.ibm.ejs.sm.adminServer.createTables	true
install.initial.config	true

Check the Web server configuration file changes

To check the Web server configuration file changes, complete the following steps:

1. Check that the following two lines have been added to httpsobj.conf above the <Object name=default> entry:

   ```
   Init fn="load-modules" funcs="as_init,as_handler,as_term"
   shlib="/opt/WebSphere/AppServer/bin/libns41_http.so"
   Init fn="as_init"
   bootstrap.properties=/opt/WebSphere/AppServer/config/plugin-cfg.xml
   ```

   You can find obj.conf in the following directory:

   ```
   <http_server_install_path>/https-<ServerName>/config
   ```

   For example:

   ```
   /opt/netscape/server4/https-itsohost.itso.ibm.com/config
   ```

2. If not, manually edit the file to include the required lines and save the changes.

Start the WebSphere administrative server processes

The WebSphere administrative server needs to be started in order to test the installation as well as the connectivity between WebSphere and the WebSphere administrative database hosted in Oracle:

1. Log in as root.
2. Start a terminal session.

1. Run the WebSphere administrative server by issuing the following commands:

   ```
   # cd <WAS_HOME>/bin
   # ./startupServer.sh
   ```

2. The startup of WebSphere administrative server is successful if the following conditions are met:

   a. There are no administrative server error logs in <WAS_HOME>/logs with names that start with __adminServer.

   b. The last line of the <WAS_HOME>/logs/tracefile file is similar to the following:

   ```
   [01.07.05 11:28:01:591 EDT]   147496 Server        I WSVR0023I: Server
   __adminServer open for e-business
   ```

Tip: To view the output sent to tracefile as it happens, use the following commands:

```
# cd /opt/WebSphere/AppServer/logs
# tail -f tracefile
```

Start the WebSphere Default Server

The WebSphere installation sets up a default application server (Default Server) in the WebSphere administrative domain. We use this application server and its Web applications to verify that the WebSphere installation is working correctly.

To start up the Default Server, perform the following steps:

1. Run the WebSphere Administrative Console by issuing the following commands:

   ```
   # cd <WAS_HOME>/bin
   # ./adminclient.sh
   ```

2. Right-click the **Default Server** under the <hostname> node and select **Start** from the pop-up menu.

3. The Default Server has been successfully started if the following conditions are met:

   a. The administrative console event messages pane shows the following messages:

      ```
      Transport http listening on port 9,080.
      Command "Default Server.start" completed successfuly.
      ```

      as shown in Figure 11-9.

   b. The last line of the <WAS_HOME>/logs/Default_Server_stdout.log file is similar to the following:

      ```
      [01.07.03 11:55:03:103 EDT]   6ff4af Server        I WSVR0023I: Server
      Default Server open for e-business
      ```

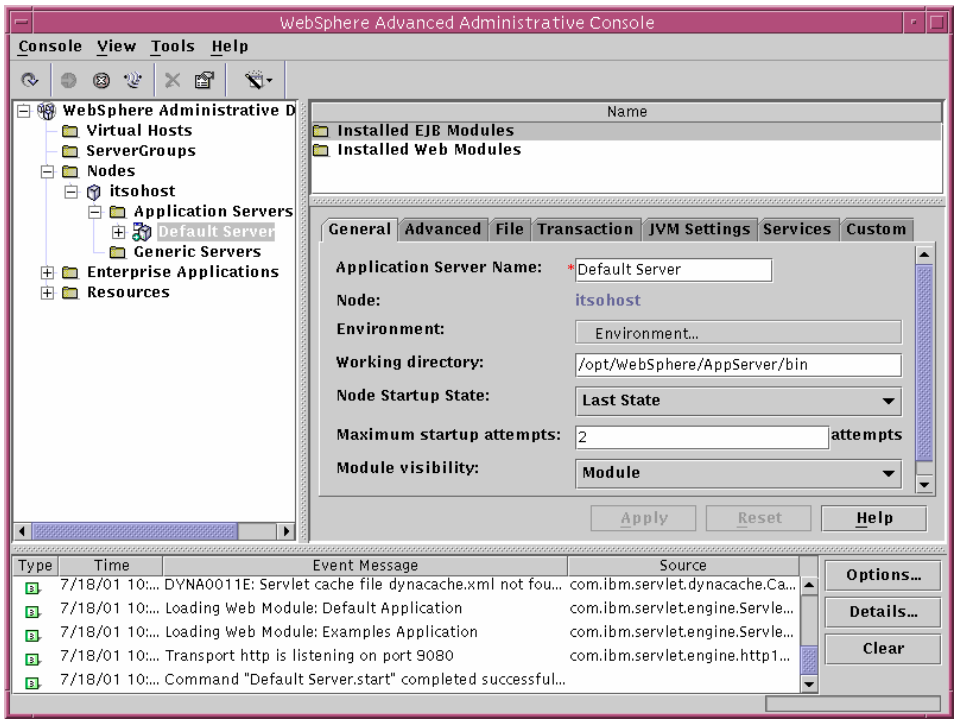


Figure 11-9 Successful startup of Default Server application server

4. Verify that the Default Server Web container has been properly installed and configured by accessing its servlets through the Web server “embedded” within the WebSphere V4.0 Web container:

   a. Using a Web browser, request the following URL:

      `http://<hostname>:9080/servlet/snoop`

      A window similar to the one shown in Figure 11-10 should be displayed in your browser.

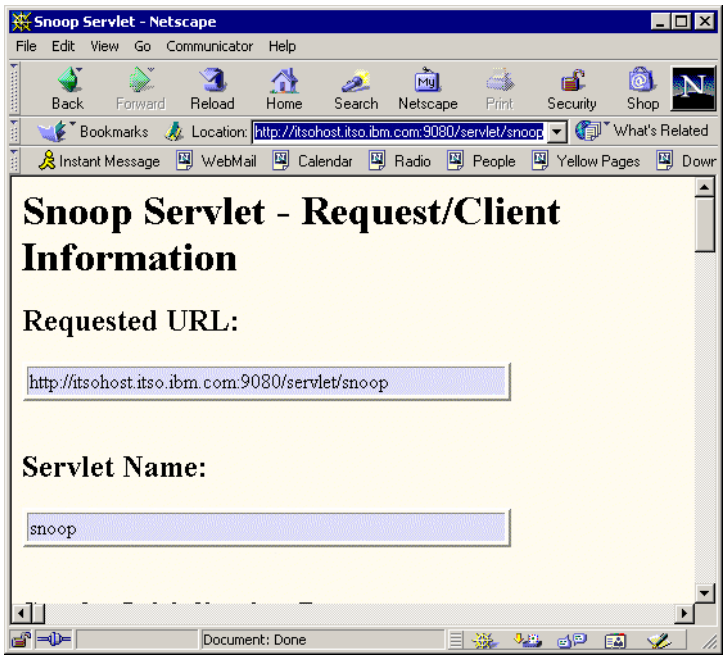


Figure 11-10 Snoop servlet accessed through embedded Web server

b. Using a Web browser, request the following URL:

```
http://<hostname>:9080/webapp/examples/showCfg
```

Note: The embedded Web server is a new feature introduced with WebSphere V4.0. In previous releases, a stand-alone Web server was required in order to access any resource hosted in WebSphere.

5. Close the WebSphere Administrative Console.

Regenerate the Web server plug-in settings

Before the Default Server can be accessed from a stand-alone Web server (such as iPlanet) the Web server plug-in settings file <WAS_HOME>/config/plugin-cfg.xml must be regenerated to reflect the following settings used by the Web server plug-in:

- Virtual host settings
- Application server transports
- Web container URIs

Perform the following steps:

1. If not already running the WebSphere Administrative Console, issue the following commands:

```
# cd <WAS_HOME>/bin
# ./adminclient.sh
```

2. Right-click the node <hostname> that contains the Default Server application server and select **Regen Webserver Plugin** from the pop-up menu, as shown in Figure 11-11.

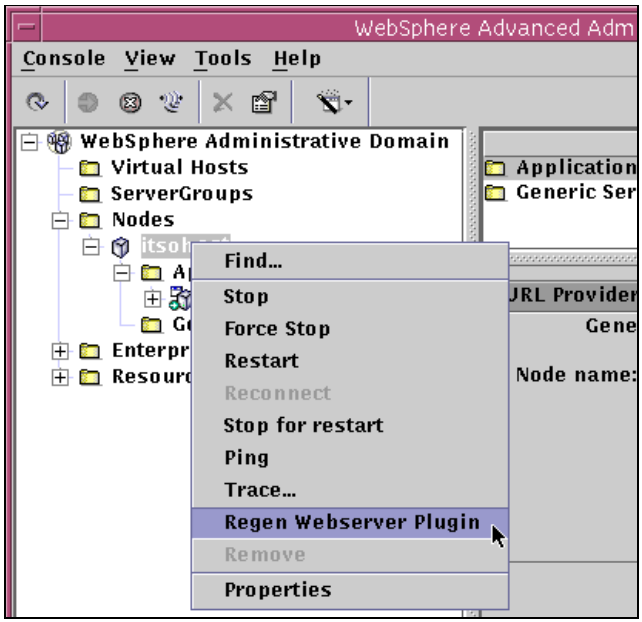


Figure 11-11 Regenerate Web server plug-in settings

Tip: WebSphere provides a command-line tool that can be used to regenerate the Web server plug-in configuration without having to run the WebSphere Administrative Console:

<WAS_HOME>/bin/GenPluginCfg.sh -adminNodeName <hostname>

3. Check that the content of the <WAS_HOME>/config/plugin-cfg.xml file has been updated to include the URIs of servlets contained within Default Server.

Tip: The plug-in regeneration command generates a <Server> element for the Default Server that contains a CloneID attribute:

```
<Server CloneID="stsul7n0" Name="Default Server">
   <Transport Hostname="m23wpn18" Port="9080" Protocol="http"/>
</Server>
```

In a non-cloned environment this attribute can be removed, resulting in performance improvements in the Web server plug-in.

Restart the Web server processes

The iPlanet Web Server process must be restarted before the Web server plug-in configuration can be tested.

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands:

   ```
   # cd <http_server_install_path>/https-<ServerName>
   # ./start
   ```

   For example:

   ```
   # cd /opt/netscape/server4/https-itsohost.itso.ibm.com
   # ./start
   ```

Verify the Web server plug-in configuration

The Web server plug-in configuration can be verified by requesting a servlet through the Web server that has already been successfully requested through the Web container's embedded Web server:

1. Using a Web browser, request a servlet URL, such as:

   ```
   http://<web_server_hostname>/snoop/servlet
   ```

   or:

   ```
   http://<web_server_hostname>/webapp/examples/showCfg
   ```

Create the WebSphere system startup script

Create a script named "websphere" in the /etc/init.d directory to be used to cleanly start up and shut down the WebSphere Application Server processes on system startup and shutdown, respectively.

1. Change to the /etc/init.d directory.

   ```
   # cd /etc/init.d
   ```

2. Create a file called websphere.

Example 11-3 provides a sample WebSphere startup script.

Example 11-3 Sample WebSphere startup script

```
#!/bin/ksh
#
# Filename: websphere
# Purpose: Cleanly startup/shutdown the iPlanet Web Server on system
# startup/shutdown respectively.

LOCAL_WAS_ROOT="/opt/WebSphere/AppServer"

case "$1" in

'start')
   cd ${LOCAL_WAS_ROOT}/bin
   ./startupServer.sh
   ;;

'stop')
   LOCAL_NODE_NAME=`uname -n`
   LOCAL_WORK_DIR=`pwd`
   LOCAL_XML_FILE=${LOCAL_WORK_DIR}/startstop-node.xml

   cat > ${LOCAL_XML_FILE} <<__EOF__
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"\$XMLConfigDTDLocation\$\$dsep\$xmlconfig.dtd" >
<websphere-sa-config>
<node name="${LOCAL_NODE_NAME}" action="stop">
</node>
</websphere-sa-config>
__EOF__

   cd ${LOCAL_WAS_ROOT}/bin
   ./XMLConfig.sh -import ${LOCAL_XML_FILE} -adminNodeName ${LOCAL_NODE_NAME}
   rm ${LOCAL_XML_FILE}
   cd ${LOCAL_WORK_DIR}
   ;;
esac
```

Note: WebSphere does not provide a shutdown equivalent of the startupServer.sh script. The script shown in Example 11-3 uses XMLConfig in order to cleanly shut down the WebSphere node. You could also use the WSCP tool. See Chapter 23, “Command-line administration and scripting” on page 881 for more information on WSCP and XMLConfig.

11.7 Install the WebSphere plug-in on a remote Web server

This section provides detailed instructions for installing, configuring, and verifying the WebSphere HTTP plug-in on a remote Web server.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install the WebSphere plug-in.
3. Verify the WebSphere plug-in installation.
4. Configure for a remote WebSphere plug-in.
5. Verify the remote plug-in installation.

11.7.1 Preinstallation tasks

The plug-in installation updates the Web server configuration, so the iPlanet Web Server processes on the Web server machine must be stopped before installing the plug-in.

To stop the iPlanet Web Server processes:

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <http_server_install_path>/https-<ServerName>
# ./stop
```

For example:

```
# cd /opt/netscape/server4/https-itsohost.itso.ibm.com
# ./stop
```

11.7.2 Install WebSphere plug-in

To install the WebSphere plug-in, complete the following steps on the iPlanet Web Server machine:

1. Log in as root and start a terminal session.
2. Load the WebSphere Application Server V4.0, Advanced Edition CD-ROM into the CD-ROM drive. The Volume Management daemon will automatically mount the CD under /cdrom/cdrom0.
3. Change the directory to the installation root:

```
# cd /cdrom/cdrom0
```

4. Ensure that the DISPLAY and TERM environment variables are correctly set.
5. Run the install.sh installation script:

```
# ./install.sh
```

6. In the Welcome window, click **Next**.

Important: If prerequisite checking is enabled for UNIX, then an alert may be displayed indicating that some of the installation prerequisites have not been met. This can even occur if newer patches/packages than those listed in prereq.properties are installed. If such an alert is displayed, recheck all prerequisites, and if they are met or exceeded, perform the following steps:

- Exit the installation.
- Disable prerequisite checking.
- Restart the installation from the beginning.

7. In the Installation Options window, select **Custom Installation** and click **Next**.
8. In the Choose Application Server Components window, shown in Figure 11-12, choose only the **WebServer Plugins** option, then click **Next**.

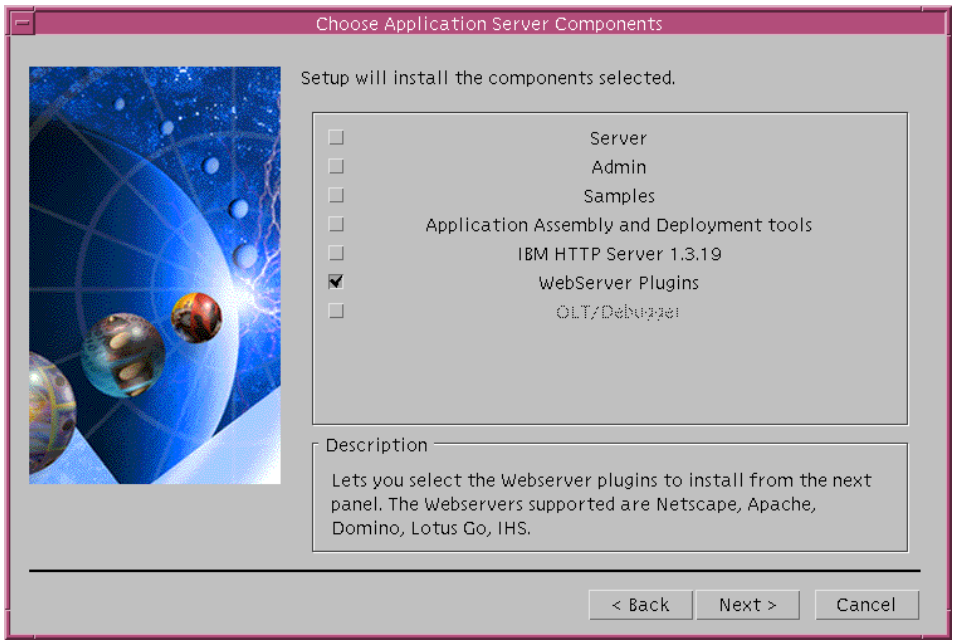


Figure 11-12 Select components required for plug-in installation

Important: Although not listed in the Application Server Components window, the Sun JDK 1.3.0 is automatically installed under the WebSphere installation directory. There is no need to separately install a JDK for use by the Web server plug-in.

9. In the Choose Webserver Plugin window, choose only the **iPlanet Web Server** plug-in, then click **Next**.
10. In the Database Options window, accept the defaults and click **Next**. None of the database settings are actually used by the Web server plug-in.
11. In the Select Destination Directory window, accept the default (/opt/WebSphere/AppServer) and click **Next**.

Note: Components installed under <plugin_install_path> by the plug-in installation include:

- Web server plug-in libraries.
- Plug-in configuration file (example).
- Plug-in certificate keystore (example).
- Sun JDK 1.3.0.

12. In the Install Options Selected window, check that the selected components are correct. If yes, click **Install** to start the installation.
13. In the Location of Configuration Files window, enter the path to the iPlanet Web Server configuration file (obj.conf), then click **Next**.
14. In the Setup Complete window, click **Finish**. The installation of the WebSphere plug-in is now complete.

11.7.3 Verify the WebSphere plug-in installation

In order to verify the installation of the WebSphere HTTP plug-in, the following task must be completed on the iPlanet Web Server machine:

1. Check Web server configuration file changes.

Check the Web server configuration file changes

To check that required settings have been added to the iPlanet Web Server configuration file (obj.conf), follow the steps described in “Check the Web server configuration file changes” on page 379.

11.7.4 Configure for a remote WebSphere plug-in

In order to support requests from a WebSphere HTTP plug-in installed on a remote Web server, the following tasks must be performed:

1. Configure the WebSphere virtual host.
2. Regenerate the Web server plug-in settings.
3. Copy the plug-in settings to the remote server.
4. Restart the Web server.

Configure the WebSphere virtual host

To map requests from the remote Web server to the required virtual host, complete the following steps:

1. Log in as root on the WebSphere server machine.
2. Start a terminal session.
3. Run the WebSphere Administrative Console by issuing the following commands:

```
# cd <WAS_HOME>/bin
# ./adminclient.sh
```

4. Select the **Virtual Hosts** folder in the tree pane of the administrative console.
5. In the details pane, select the **default_host** virtual host.
6. Add three new entries to the Host Aliases list of the default_host virtual host, as shown Figure 11-13.

```
<Web server hostname>:<port#>
<Web server hostname.domain.com>:<port#>
<Web server IP address>:<port#>
```

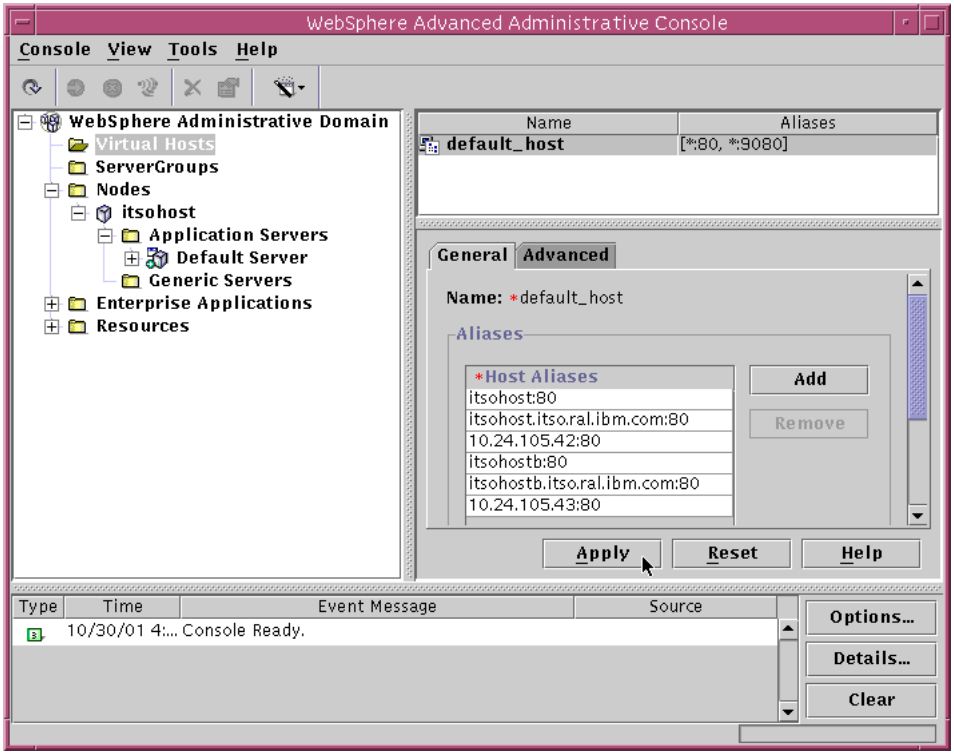


Figure 11-13 Add required aliases to virtual host

7. Click **Apply** to save changes.

Important: Always remember to click the **Apply** button when changing settings with the administrative console. Failure to do so will result in all changes being lost, even if you just click into another resource in the Console.

8. Restart each of the application servers that are associated with this virtual host.

Regenerate the Web server plug-in settings

To regenerate the Web server plug-in settings:

1. Follow the steps described in “Regenerate the Web server plug-in settings” on page 382.
2. Check that the content of the <WAS_HOME>/config/plugin-cfg.xml file has been updated to include the remote Web server’s host name among the virtual hosts settings.

Copy the plug-in settings to the remote server

Next the regenerated Web server plug-in settings must be copied to the remote Web server:

1. Copy the <WAS_HOME>/config/plugin-cfg.xml file from the WebSphere server machine across to the <WAS_HOME>/config directory on the remote Web server machine.

Important: WebSphere and the WebSphere HTTP plug-in can have different installation paths on the two servers. However, for simplicity and to reduce editing of the plugin-cfg.xml file, we recommend that the plug-in installation use the same path as the full WebSphere installation.

Restart the Web server

Restart the Web server if you want it to load the new plug-in settings immediately, or wait until the plug-in dynamically reloads.

1. Log in as root on the iPlanet Web Server machine.
2. Start a terminal session.
3. Restart the Web server by issuing the following commands:

```
# cd <http_server_install_path>/https-<ServerName>
# ./start
```

For example:

```
# cd /opt/netscape/server4/https-itsohost.itso.ibm.com
# ./start
```

11.7.5 Verify the remote plug-in configuration

In order to verify the configuration of the WebSphere HTTP plug-in installed on a remote Web server, the following tasks must be performed:

1. Check the plug-in logs.
2. Test the connection to WebSphere.

Check the plug-in logs

To check the plug-in logs, complete the following steps:

1. The location of the WebSphere HTTP plug-in's log is specified by the <Log> element of the plugin-cfg.xml configuration file. By default, this is set to the following:

   ```
   <Log LogLevel="Warning" Name="<plugin_install_path>/logs/native.log"/>
   ```

2. Check the contents of this log file. If the plug-in has been correctly configured, then the following lines will be written to the end of the file:

   ```
   [Sun Jul 29 14:02:37 2001] 00001df5 00000001 - PLUGIN: Plugins loaded.
   [Sun Jul 29 14:02:37 2001] 00001df5 00000001 - PLUGIN:
   --------------------System Information----------------------
   [Sun Jul 29 14:02:37 2001] 00001df5 00000001 - PLUGIN: Bld date: Jul 11
   2001, 21:08:29
   [Sun Jul 29 14:02:37 2001] 00001df5 00000001 - PLUGIN: Webserver:
   iPlanet-WebServer-Enterprise/4.1
   [Sun Jul 29 14:02:37 2001] 00001df5 00000001 - PLUGIN: Hostname = itsohost
   [Sun Jul 29 14:02:37 2001] 00001df5 00000001 - PLUGIN: NOFILES = hard:
   1024, soft: 1024
   [Sun Jul 29 14:02:37 2001] 00001df5 00000001 - PLUGIN: MAX COREFILE SZ =
   hard: INFINITE, soft: INFINITE
   [Sun Jul 29 14:02:37 2001] 00001df5 00000001 - PLUGIN: DATA = hard:
   INFINITE, soft: INFINITE
   [Sun Jul 29 14:02:37 2001] 00001df5 00000001 - PLUGIN:
   ------------------------------------------------------------
   ```

Note: See Appendix C, "The plugin-cfg.xml file definitions" on page 1071 for more details.

Test the connection to WebSphere

To test the remote Web server connection to WebSphere, complete the following steps:

1. Using a Web browser, request the following URL:

   ```
   http://<web_server_hostname>/servlet/snoop
   ```

2. If both the Web server and WebSphere have been correctly configured to support remote HTTP plug-in access, then a window similar to Figure 11-14 will be obtained.

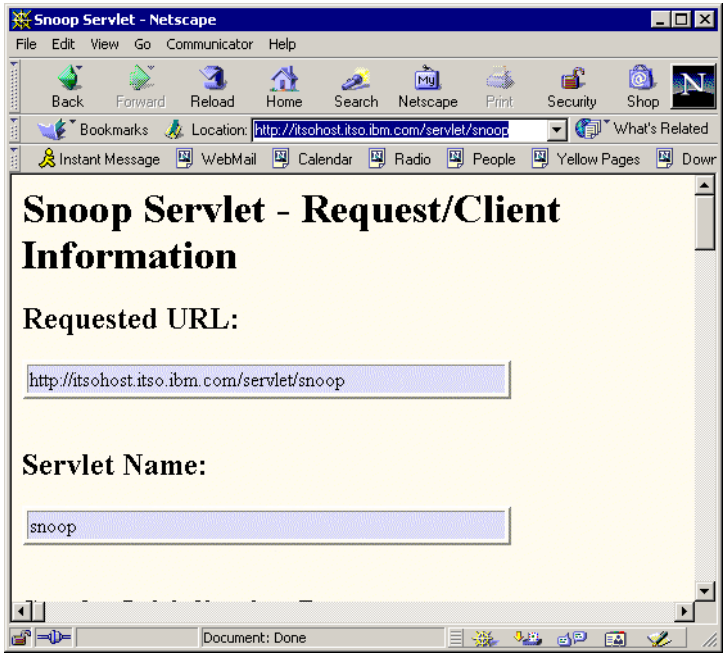


Figure 11-14 Request handled through remote HTTP plug-in

11.8 Install a new WebSphere node into an existing domain

This section provides detailed instructions for installing, configuring, and verifying a new WebSphere node into an existing WebSphere V4.0 administrative domain.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install WebSphere.
3. Verify the WebSphere installation.

11.8.1 Preinstallation tasks

Prior to installing the new WebSphere node, complete the basic preinstallation tasks specified in 11.6.1, “Preinstallation tasks” on page 370.

Important: This section assumes that the iPlanet Web Server and Oracle 8i client (in place of the Oracle 8i server) are both already installed and configured on the server on which you will be installing this new WebSphere node.

11.8.2 Install WebSphere

To install the new WebSphere node into an existing WebSphere domain, complete the basic installation tasks specified in 11.6.2, “Install WebSphere” on page 371, except for the following:

1. In the Database Options window, specify:
   - Database Name: specify Oracle 8i service name for WebSphere database instance.
   - Path: specify the value of <oracle_home> for the Oracle 8i client.
   - Remote Database: leave this unselected.

11.8.3 Verify the WebSphere installation

In order to verify the new WebSphere node installed into an existing WebSphere domain, complete the basic verification tasks specified in 11.6.3, “Verify the WebSphere installation” on page 377, except for the following:

1. For each new node sharing an existing administrative database, ensure the database tables are not created and the default resources are not installed a second time. Edit the <WAS_HOME>/bin/admin.config file and set the values of the following settings as shown:

```
install.initial.config=false
com.ibm.ejs.adminServer.createTables=false
```

11.9 Install the WebSphere Application Server - silent mode

This section describes how to install WebSphere Application Server V4.0, Advanced Edition using the noninteractive, or silent, mode. To complete a silent installation, you will use the default response file or create a customized one, then execute the installation script for WebSphere Application Server, supplying the response file as a command-line parameter.

These instructions assume the following:

- Your machine has sufficient memory and disk space for your installation.
- You have installed and configured a database.
- You do not have a previous version of WebSphere Application Server already installed on this machine. If you do have a previous version of WebSphere Application Server already installed, do not follow these instructions. Instead, see Chapter 25, “Migration” on page 1031.
- If you are using IBM HTTP Server as your Web server, you will install it at the same time and onto the same node as you install WebSphere Application Server. If you are using another supported Web server with WebSphere Application Server, you have already installed it onto the same node as WebSphere Application Server.

Note: IBM HTTP Server is supplied with WebSphere Application Server. If you plan to use a different Web server, you must purchase it and install it separately. It is recommended that the Web server be installed before WebSphere Application Server.

11.9.1 Using the default response file

A default response file, named install.script, is supplied with WebSphere Application Server. You can use this default response file to install WebSphere Application Server using the default options, or as a template for creating a customized response file.

If you use the default response file to install WebSphere Application Server using the default options, the following software and other resources are installed:

- IBM Java 2 Software Developer's Kit (SDK) 1.3.0
- IBM HTTP Server V1.3.19
- IBM WebSphere Application Server V4.0
- WebSphere Application Server application samples

- Documentation in U.S. English

Note: All files except IBM HTTP Server are installed into the directory /opt/WebSphere/AppServer. IBM HTTP Server is installed into the directory /opt/IBMHTTPD. In addition, WebSphere Application Server is configured for use with IBM HTTP Server.

11.9.2 Using a customized response file

You can also use the default response file as a template for creating a customized response file. You can edit the default response file to enable the configuration of WebSphere Application Server with a different supported Web server or database or to install the products into a different directory. Detailed comments within the default response file guide you through the installation and configuration options available for performing a silent installation.

11.9.3 Performing a silent installation

Perform the following steps to create a customized response file (if needed) and install WebSphere Application Server. These instructions assume that the installation is being performed from the product CD-ROM:

1. Ensure that you are logged into the machine with superuser (root) privileges.
2. If a preexisting Web server on your system is running, stop the Web server. If you plan to install IBM HTTP Server V1.3.19 as part of the WebSphere Application Server installation and a version prior to V1.3.19 is already installed on your system, you must uninstall it in order for the WebSphere Application Server installation program to successfully install V1.3.19.
3. Insert the IBM WebSphere Application Server V4.0, Advanced Edition CD-ROM in the CD-ROM drive. The Solaris Volume Management daemon will automatically mount the CD under /cdrom/cdrom0.
4. Navigate to the /cdrom/cdrom0/sun directory by entering the following command:

   ```
   # cd /cdrom/cdrom0/sun
   ```

5. Ensure that the directory /usr/ucb exists in the PATH environment variable for the root login. If it does not, you must edit the install.sh script. To edit this script, do the following:

   a. Copy the install.sh script from the /cdrom/cdrom0/sun directory to the /tmp directory on the machine on which you will install WebSphere Application Server.

   b. Open this script in a text editor and find the line:

```
USERNAME=`/usr/ucb/whoami`.
```

c. Add the following line before the line USERNAME='/usr/ucb/whoami':

```
export PATH = $PATH:/usr/ucb
```

d. Save the edited install.sh script.

6. Ensure that you are in the directory /cdrom/cdrom0/sun and create a copy of the default response file by using the **cp** command, as follows:

```
# cp install.script <new_install.script>
```

In this command, <new_install.script> represents the full path name of the copy of the default response file you are creating (for example, /tmp/new_install.script). The name of your response file must have a .script extension.

7. If you plan to install WebSphere Application Server by using the default options included in the default response file, proceed to the next step.

If you plan to use the default response file as a template for creating a customized response file, perform the following steps:

a. Use a text editor to open your copy of the default response file, <new_install.script>.

b. Use the detailed comments throughout the file to help you select the appropriate options for your WebSphere Application Server installation.

c. Save the changes that you have made to the customized response file.

8. Run the installation script in one of the following ways depending on the actions you have taken in Step 5. (Because the shell script performs some pre- and post-processing steps on the installation, do not use the Solaris admintool utility to run the script.) The install.sh shell script uses the response file to install the components and options that you have selected. In each case, the value <new_install.script> represents the full path name of the copy of the default response file or the customized response.

– If you have edited the install.sh script as detailed in Step 5, run the installation script file by entering the following commands:

```
# cd /cdrom/cdrom0/sun
# /tmp/install.sh -silent -responseFile new_install.script
```

– If you have not edited the install.sh script as detailed in Step 5, run the installation script file by entering the following command:

```
# /cdrom/cdrom0/sun/install.sh -silent \
-responseFile <new_install.script>
```

If you choose to install the plug-in for IBM HTTP Server, the installation process checks if you have the correct version of the Web server on your machine. If you do not have IBM HTTP Server installed on your machine, the installation process performs one of the following actions, based on whether you have indicated in your response file to install IBM HTTP Server:

- If you indicated in your response file that you do want to have IBM HTTP Server installed, the installation process installs the plug-in for it.
- If you indicated in your response file that you do not want to have IBM HTTP Server installed, the script exits without installing the plug-in.

9. After installation is complete, refer to the log file named install.log located in the /tmp directory to determine whether the silent installation was successful. A copy of this file also exists in the directory <WAS_HOME>/logs.
10. Unmount the CD-ROM before removing it from the CD-ROM drive by using the **umount** command, as follows:

    ```
    # umount /cdrom/cdrom0
    ```

    You can then eject the CD-ROM.
11. If you are using a Web server other than IBM HTTP Server, start the server. If you installed IBM HTTP Server as part of the WebSphere Application Server silent installation, you might need to configure it. Perform the following steps to verify that the IBM HTTP Server is installed correctly:

    a. Ensure that the Web server is running or start it by entering the following command:

       ```
       # /opt/IBMHTTPD/bin/apachectl start
       ```

    b. Start a Web browser window and type the name of the host machine as the URL (`http://host_machine`). If you see the Welcome to the IBM HTTP Server window, the server has been installed correctly.

See the “Using silent installation on Solaris“ article in the InfoCenter for further details.

12

Linux installation steps

This chapter provides detailed procedures for installing, configuring, and verifying a number of the scenarios described in Chapter 8, “Installation approach” on page 127, for an environment consisting of the following components:

- Operating system - Red Hat Linux 7.1
- Database server - IBM DB2 UDB Enterprise Edition
- Web server - IBM HTTP Server
- Application server - WebSphere Application Server V4.0, Advanced Edition

The procedures described are intended to be used as working examples in conjunction with the product installation guides for all the possible values that may be unique within your runtime environment. This chapter is organized into the following sections:

- Planning
- Install Linux
- Install the Web server
- Install the database server
- Install WebSphere Application Server V4.0, Advanced Edition
- Configure WebSphere HTTP transport for SSL
- Install WebSphere Application Server - silent mode

Note: For more information on WebSphere for Linux, see *WebSphere Application Server V4 for Linux, Implementation and Deployment Guide*, REDP0405, found at http://www.ibm.com/redbooks. This Redpaper provides detailed procedures for implementing multi-tier runtime environments for WebSphere Application Server V4.0, Advanced Edition for Linux. It also describes how to deploy an enterprise application to a WebSphere for Linux runtime environment.

12.1 Planning

This section defines the hardware and software used within the Linux environment to test a number of different WebSphere Application Server V4.0 configuration scenarios.

12.1.1 Hardware and software prerequisites

WebSphere Application Server V4.0, Advanced Edition has the following hardware and software requirements.

Hardware

- 150 MB diskspace (minimum) for WebSphere Application Server
- 300 MB diskspace (minimum) for IBM DB2 Enterprise Edition
- 50 MB diskspace (minimum) for IBM HTTP Server

Software

- Red Hat Linux 7.1 Standard Edition + prerequisites (see 12.2, “Install Linux” on page 405)
- IBM WebSphere Application Server V4.0, Advanced Edition
- IBM DB2 Universal Database V7.2 FP4, Enterprise Edition for Linux
- IBM HTTP Server 1.3.19
- IBM GSKit 5.0.3.52

Note: For further details on requirements, see the following documentation:

1. DB2 -

   http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winox2unix/support/v7pubs.d2w/en_main

2. WebSphere -

   http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

12.1.2 Software used in our test environment

We used the following software in our test environment:

- Red Hat Linux 7.1 Standard Edition + prerequisites
- IBM WebSphere Application Server V4.0, Advanced Edition
- IBM DB2 Universal Database V7.2 FP4, Enterprise Edition for Linux
- IBM HTTP Server 1.3.19
- IBM GSKit 5.0.3.52 (included in IBM HTTP Server package)
- IBM JDK 1.3.0 (included in WebSphere Application Server package)

Note: Other Web servers and database software may be used, as documented in the *Product Installation Guide*.

Product installation roots

The variables listed in Table 12-1 are used frequently throughout this documentation to represent the root installation directories of the software components.

Table 12-1 Product installation roots

Variable	Default value	Component
<WAS_HOME>	/opt/WebSphere/AppServer	WebSphere
<db2_install_path>	/home/db2inst1	DB2 server
<http_server_install_path>	/opt/IBMHTTPServer	IBM HTTP Server

Installation variables

The variable listed in Table 12-2 are used frequently throughout this documentation to represent settings used during the installation of software components.

Table 12-2 Common installation settings

Variable	Default value	Component
db2_instance_owner	db2inst1	DB2 server

12.1.3 Hardware used in our test environment

This section describes the hardware used within our test WebSphere Application Server V4.0 environment on Linux.

- Server 1
  - IBM NetFinity 3000
  - 1 350 MHz CPU
  - 512 MB RAM
  - 4.3 GB hard disk
  - 1 Intel 82558 PCI Ethernet Adapter
  - 1 Adaptec AIC-78xx PCI SCSI Adapter
  - 1 S3 Trio 3D Video Adapter

12.1.4 Example scenarios

Within this test environment, we describe the tasks necessary to install and configure the following scenarios that are described in Chapter 8, “Installation approach” on page 127:

- Scenario A - basic one-server configuration, but using the WebSphere embedded Web server in place of the usual standalone Web server.
- Scenario B - basic one-server configuration, but using a standalone Web server and the WebSphere HTTP plug-in.
- Scenario C - extension of the basic one-server configuration of Scenario B, with the HTTP communication between the plug-in and WebSphere encrypted using SSL and client authentication mode enabled.

Installation and configuration tasks

Table 12-3 provides a detailed summary of the tasks and the order in which they must be performed that are necessary to install and configure each of the example scenarios described above.

To install and configure an example, perform the tasks listed for the example in the specified order. Only those tasks that are numbered (non-blank) are to be performed for a particular example. Detailed descriptions of each of these tasks can be found later in this chapter.

Table 12-3 Installation and configuration tasks broken down by scenario

Server	Task	Example scenario		
		A	B	C
1	Operating system: Installation of Linux operating system and any required patches: 1. Install Linux 2. Apply any paches and packages as detailed in the WebSphere platform specific prerequisites See 12.2, “Install Linux” on page 405.	1	1	1
1	Web server: Installation and configuration of IBM HTTP Server: 1. Perform preinstallation tasks 2. Install IBM HTTP Server 3. Configure IBM HTTP Server 4. Verify IBM HTTP Server See 12.3, “Install the Web server” on page 407.	-	2	2
1	Database server: Installation and configuration of IBM DB2 server necessary to support usage by WebSphere: 1. Perform preinstallation tasks 2. Install DB2 server 3. Verify DB2 server installation 4. Configure DB2 server 5. Set up WebSphere administrative database See 12.4, “Install the database server” on page 413.	2	3	3

Server	Task	Example scenario		
		A	B	C
1	WebSphere Application Server: Installation and configuration of WebSphere necessary to handle requests through the "embedded" Web server: 1. Perform preinstallation tasks 2. Install WebSphere 3. Verify WebSphere installation See 12.5, "Install WebSphere Application Server" on page 425.	3	4	4
1	WebSphere Application Server: Creation and configuration of an HTTPS transport on WebSphere default Web container: 1. Create new server certificate keystore 2. Create new server self-signed certificate used for SSL handshaking 3. Create new HTTPS transport for WebSphere Default Server See 12.6, "Configure WebSphere HTTP transport for SSL" on page 434.	-	-	5
1	WebSphere HTTP plug-in: Configuration of plug-in necessary to communicate with HTTPS (client authentication disabled) transport in WebSphere: 1. Create new plug-in certificate keystore 2. Import WebSphere self-signed server certificate as a trusted CA 3. Update Web server plug-in configuration file to support HTTPS transport and certificate keystore. See 12.6, "Configure WebSphere HTTP transport for SSL" on page 434.	-	-	6

Server	Task	Example scenario		
		A	B	C
1	WebSphere HTTP plug-in: Extra configuration of plug-in necessary to support HTTPS transport with client authentication enabled: 1. Create new self-signed client certificate for use to authenticate plug-in to server during SSL handshaking See 12.6, “Configure WebSphere HTTP transport for SSL” on page 434.	-	-	7
1	WebSphere Application Server: Extra configuration of WebSphere HTTPS transport necessary to support client authentication: 1. Import plug-in self-signed certificate as a trusted CA 2. Configure HTTPS transport to use client authentication See 12.6, “Configure WebSphere HTTP transport for SSL” on page 434.	-	-	8
1	WebSphere Application Server: Start up the WebSphere Default Server ready to handle requests: 1. Regenerate Web server plug-in configuration file See 12.6, “Configure WebSphere HTTP transport for SSL” on page 434.	-	-	9
1	Web server: Test that Web server and plug-in can access the current configuration of the WebSphere Default Server: 1. Restart Web server 2. Verify configuration – `http://<hostname>/webapp/examples/showCfg`	-	5	10

12.2 Install Linux

Prior to installing any of the WebSphere components, the proper level of the operating system must be installed.

- Linux kernel
- File systems
- Linux packages

12.2.1 Linux kernel

The WebSphere V4.0 minimum supported level of Red Hat Linux is Release 7.1, running the 2.4 kernel.

12.2.2 File systems

During the interactive phase of the Red Hat Linux 7.1 operating system installation, you will need to customize the file systems and allocate the necessary space for the software components installed in subsequent steps.

Table 12-4 provides general guidelines for the space required by each component.

Table 12-4 Component disk space requirements

Component	File system	Disk space required (MB)
IBM HTTP Server	/opt	50
WebSphere Application Server V4.0, Advanced Edition	/opt	250
IBM DB2 Server, Enterprise Edition	/usr /home	300 MB 100 MB

Based on the disk space requirements listed in Table 12-4, we recommend the file systems listed in Table 12-5 for this test environment.

Table 12-5 Recommended file systems

File system	Description	Usage	Size (MB)
/	root		400
/tmp	Temporary files		200
/home	Home directories	DB2 instance	600
/opt	Program files	WebSphere, IBM HTTP Server	1200
/usr	Program files	DB2 Server	1200
<swap>	Swap space		512

12.2.3 Linux packages

There are a number of prerequisite packages (of specific versions) required for the installation of WebSphere Application Server V4.0. The required packages and their version numbers are listed in Table 12-6.

Table 12-6 Red Hat Linux packages required by WebSphere V4.0

Package	Require this version or newer...
ncurses4	5.0-2
pdksh	5.2.14-12 (installed when the ncurses4 package gets installed).

1. Determine whether each of these packages are currently installed by using the **rpm** (Red Hat Package Management) command:

   ```
   # rpm --verify <package name>
   ```

2. If the output does not include the required version of the fileset (or newer), then the package must be upgraded before continuing. Use the following command to install or upgrade a package:

   ```
   # rpm -U --nodeps <package name>-<version>.i386.rpm
   ```

   For example:

   ```
   # rpm -U --nodeps IBM_ADMIN_Server-1.3.19-0.i386.rpm
   ```

Note: The ncurses4 package is supplied on the Red Hat Linux 7.1 CD-ROM, but is not installed by default.

Red Hat Linux packages can be obtained from the following FTP site:

ftp://ftp.redhat.com/

12.3 Install the Web server

This section provides detailed instructions for installing, configuring, and verifying IBM HTTP Server V1.3.19 for Linux.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install the IBM HTTP Server.
3. Configure the IBM HTTP Server.
4. Verify the IBM HTTP Server.

Note: Whether or not a particular task needs to be performed, and the order in which tasks must be performed, is identified during the planning stage and is dependent upon the needs of the particular topology or scenario being installed. See 12.1, “Planning” on page 400 for details on those tasks to be performed for each example.

12.3.1 Preinstallation tasks

Prior to installing IBM HTTP Server V1.3.19, the following check must be completed:

1. Check that IP ports are unused.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing services on the server that use the following IP ports:
   - 80 (standard HTTP port)
   - 443 (standard HTTPS port)
   - 8008 (IBM HTTP Server Administration port)

   We suggest using the following command for this task:

   ```
   # netstat -a | grep LISTEN
   ```

12.3.2 Install the IBM HTTP Server

To install the IBM HTTP Server V1.3.19, complete the following steps:

1. Log in as root.
2. Start a terminal session.
3. Insert the IBM WebSphere Application Server V4.0 CD-ROM into the CD-ROM drive. This CD also contains the IBM HTTP Server.
4. Mount the CD-ROM:

   ```
   # mount -t iso9660 -r /dev/cdrom /cdrom
   ```

5. Change to the ihs_128 subdirectory of the root installation directory on the CD.

6. Create a text file rpm.list listing each of packages given in Table 12-7.

Table 12-7 Required IBM HTTP Server packages

Package	Description
gsk5bas-5.0-3.61.i386.rpm	GSK certificate and security libraries
IBM_ADMIN_EN-1.3.19-0.i386.rpm	IBM Administration Server documentation
IBM_ADMIN_Server-1.3.19-0.i386.rpm	IBM Administration Server program files
IBM_FastCGI-1.3.19-0.i386.rpm	Implementation of FastCGI standard - increases CGI performance
IBM_HTTP_Server-1.3.19-0.i386.rpm	IBM HTTP Server program files
IBM_MAN_ENU-1.3.19-0.i386.rpm	IBM HTTP Server manual (man) pages
IBM_MSG_EN-1.3.19-0.i386.rpm	Language message files for IBM HTTP Server
IBM_SSL_128-1.3.19-0.i386.rpm	IBM SSL (Secure Sockets Layer) 128-bit library
IBM_SSL_Base-1.3.19-0.i386.rpm	IBM SSL (Secure Sockets Layer) program files
IBM_SSL_EN-1.3.19-0.i386.rpm	Language message files for IBM SSL

Important: Double-check that you have selected the correct packages. Due to the number of selections, it is easy to make a mistake that will result in the IBM HTTP Server not working properly.

7. Install the packages using the **rpm** (Red Hat Package Management) tool:

```
# for i in ‘cat rpm.list‘ ; do rpm -U --nodeps $i ; done
```

8. Unmount the CD-ROM:

```
# cd /
# umount /cdrom
```

12.3.3 Configure the IBM HTTP Server

After installation of IBM HTTP Server V1.3.19, the following configuration tasks must be completed on the IBM HTTP Server machine:

1. Create the IBM HTTP Server admin account.
2. Create a UNIX runtime account.
3. Update httpd.conf.

4. Restart IBM HTTP Server.

Create the HTTP server admin account

The administration account is used to access the HTTP Administration Server Configuration GUI. To create the account, perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Change the directory to the <http_server_install_path>/bin directory.
4. Create the administration user by typing the following commands:

```
# ./htpasswd -m ../conf/admin.passwd admin
New password: <your_password>
Re-type new password: <your_password>
```

   Where `admin` is the IBM HTTP Server administration user ID.

Create a UNIX runtime account

Although the HTTP Server process is started under the root account, it must be configured to, then switched to, run under another account. A UNIX account needs to be created specifically for the purpose.

Perform the following steps to create the UNIX account and configure the IBM HTTP Server:

1. Log in as root.
2. Start a terminal session.
3. Change the directory to the <http_server_install_path>/bin directory.
4. Run the setupadm script:

```
# ./setupadm
```

5. Answer the prompts as follows:
   a. Please supply a user ID to run the Administration Server. For example:

      User ID: `www`

      Press Enter.

   b. Please Supply a GROUP NAME to run the Administration Server. For example:

      Group Name: `www`

      Press Enter.

   c. Please supply the Directory containing the files for which a change in the permissions is necessary. The default is /opt/IBMHTTPServer/conf.

Press Enter to accept the default.

d. To perform the change, enter 1. To quit with no changes, enter 2 (default).

   Enter 1 to perform changes.

   Press Enter.

e. Configuration file: '/opt/IBMHTTPServer/conf/admin.conf' will be saved. Do you wish to update the Administration Server Configuration file? To perform the change, enter 1. To exit with no change, enter 2 (default).

   Enter 1 to update configuration file.

   Press Enter.

f. Do you wish to run the Admin Server and IHS Server in a language other than English? For a language other than English, enter 1. For English, enter 2 (default).

   Press Enter to accept the default (English).

6. The setupadm program returns to the system prompt.

Update httpd.conf

The HTTP Server configuration file httpd.conf must be updated to reflect the following:

- The fully qualified host name of the server.
- The UNIX account to run under.

1. Edit the <http_server_install_path>/conf/httpd.conf file and update the settings listed in Table 12-8.

Table 12-8 httpd.conf required settings

Setting	Required value...
User	www
Group	www
ServerName	<hostname.domain.com>

2. Save the changes and exit.

Restart IBM HTTP Server

The IBM HTTP Server must be restarted in order for the above configuration changes to take effect:

1. Log in as root.
2. Start a terminal session.

3. Issue the following commands:

```
# cd <http_server_install_path>/bin
# ./apachectl restart
```

12.3.4 Verify IBM HTTP Server

In order to verify the IBM HTTP Server V1.3.19 installation, perform the following checks on the IBM HTTP Server machine:

1. Check the process status.
2. Check request handling.

Check the process status

To check the IBM HTTP Server process status, perform the following steps:

1. Check that the HTTP Server processes are running by issuing the following command:

   ```
   ps -ef | grep httpd
   ```

   The output should list a number of processes.

2. Check that the HTTP Server is registered to listen on port 80 and is therefore ready to handle requests:

   ```
   netstat -a | grep LISTEN | grep www
   ```

Check request handling

To check IBM HTTP Server request handling, perform the following steps:

1. Using a Web browser, request the following URL representing the IBM HTTP Server home page:

   ```
   http://<hostname.domain.com>/
   ```

   The window shown in Figure 12-1 will be displayed if the IBM HTTP Server has been installed and configured correctly.

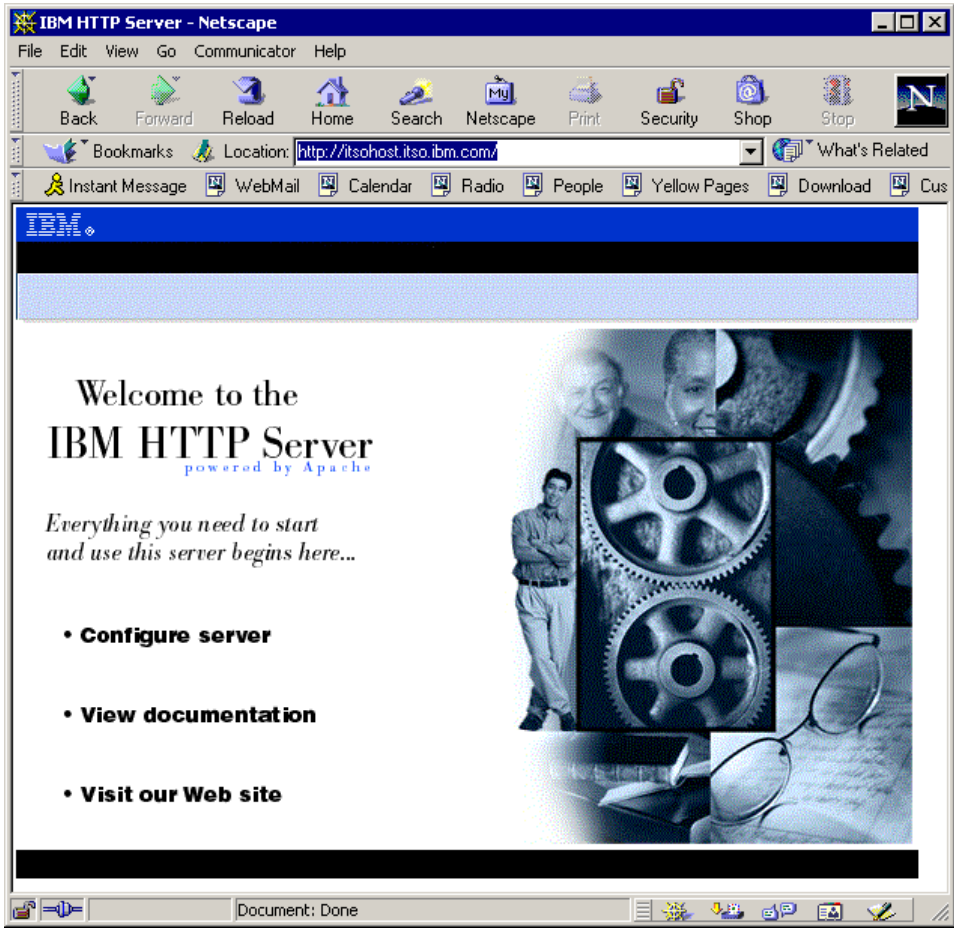


Figure 12-1 Home page request handled by Web server

12.4 Install the database server

This section provides detailed instructions for installing, configuring, and verifying IBM DB2 Universal Database, Enterprise Edition for Linux.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install the DB2 server.
3. Verify the DB2 server installation.
4. Configure the DB2 server.

5. Set up the WebSphere administrative database.

12.4.1 Preinstallation tasks

Prior to installing IBM DB2 Universal Database, Enterprise Edition for Linux, the following task needs to be completed:

1. Check IP ports are unused.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing active services that use the following IP ports on the server:
   - 523 (DB2 server)
   - 50000 (DB2 instance connection port)
   - 50001 (DB2 instance interrupt port)
   - 50002 (DB2 Control Server)

   We suggest using the following command for this task:

   ```
   # netstat -a | grep LISTEN
   ```

12.4.2 Install the DB2 server

In order to install IBM DB2 Universal Database, Enterprise Edition for Linux, perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Load the IBM DB2 Universal Database, Enterprise Edition for Linux CD-ROM into the CD-ROM drive and mount the CD.

   ```
   # mount -t iso9660 -r /dev/cdrom /cdrom
   ```

4. Start the DB2 installer program:

   ```
   # cd /cdrom
   # ./db2setup
   ```

5. In the Install DB2 window, select only the following option:
   - **DB2 UDB Enterprise Edition**

Navigation tips:

- Press Tab to move between available options and fields.
- Highlight and press Enter to select an option.

6. Highlight the **DB2 Product Library** option and press Enter.
7. In the DB2 Product Library window, highlight the appropriate option for your locale under the DB2 Product Library (HTML) section, then highlight **OK** and press Enter.
8. Highlight **OK** and press Enter.
9. In the Create DB2 Services window, select the **Create a DB2 Instance** option, highlight **OK** and press Enter.
10. In the DB2 instance authentication window, enter the following:
    - User Name: <db2_instance_owner>
    - User ID: use default UID
    - Group Name: db2iadm1
    - Group ID: use default GID
    - Home Directory: /home/<db2_instance_owner>
    - Password: <user_password>
    - Verify Password: <user_password>

    Important: The DB2 installer uses the above information to automatically perform the following operations:

    - Create a group db2iadm1.
    - Create a user <db2_instance_owner> with primary group db2iadm1.
    - Sets <db2_instance_owner> password to <user_password> value. The password used must meet DB2 requirements: 8 characters or less and not containing the characters “<“ or “>”.
    - Changes the ownership (owner:group) of the /home/<db2_instance_owner> directory to be <db2_instance_owner>:db2iadm1

11. Highlight **OK** and press Enter.
12. In the Fence User window, enter the following:
    - User Name: db2fenc1

- User ID: use default UID
- Group Name: db2fadm1
- Group ID: use default GID
- Home Directory: /home/db2fenc1
- Password: <db2fenc1_password>
- Verify password: <db2fenc1_password>

Important: The DB2 installer uses the above information to automatically perform the following operations:

- Create a group db2fadm1
- Create a user db2fenc1 with primary group db2fadm1
- Sets db2fenc1 password to <db2fenc1_password> value. The password used must meet DB2 requirements: 8 characters or less and not containing the characters “<“ or “>”.
- Changes the ownership (owner:group) of the /home/db2fenc1 directory to be db2fenc1:db2fadm1

13. Highlight **OK,** and press Enter.
14. In the DB2 Warehouse Control Database window, select the **Do not set up DB2 Warehouse Control Database** option, then highlight **OK,** and press Enter.
15. In the Create DB2 Services window, highlight the **Create Administration Server** option, then enter the following:
    - User Name: db2as
    - User ID: use default UID
    - Group Name: db2asgrp
    - Group ID: use default GID
    - Home Directory: /home/db2asgrp
    - Password: <db2asgrp_password>
    - Verify password: <db2asgrp_password>

Important: The DB2 installer uses the above information to automatically perform the following operations:

- Create a group db2asgrp
- Create a user db2as with primary group db2asgrp
- Sets db2as password to <db2as_password> value. The password used must meet DB2 requirements: eight characters or less and not containing the characters “<“ or “>”.
- Changes the ownership (owner:group) of the /home/db2as directory to be db2as:db2asgrp

16. Highlight **OK** and press Enter.
17. A message window appears indicating that DB2SYSTEM will be set to <hostname>. Highlight **OK** and press Enter.
18. Back in the Create DB2 Services window, highlight **OK** and press Enter.
19. The Summary Report window is displayed, listing the product components to be installed. Highlight **Continue** and press Enter.
20. A warning window appears indicating this is your last chance to stop. Highlight **OK** and press Enter.
21. The db2setup program installs the selected components. Depending on the speed of your processor, this can take up to 15 minutes.
22. You may be prompted to register the product. Complete the registration, then exit back to the install window.
23. When the install completes, a notice window informs you whether the installation was successful. Highlight **OK** and press Enter.
24. Scan the Status Report to ensure that all components were installed successfully. Highlight **OK** and press Enter.
25. In the DB2 Installer window, highlight **Close** and press Enter.
26. A window appears asking `Do you want to exit the DB2 Installer?`. Highlight **OK** and press Enter.
27. The DB2 installation is now complete.
28. Unmount the CD-ROM:

```
# cd /
# umount /cdrom
```

12.4.3 Verify the DB2 server installation

To verify the DB2 Server installation, complete the following tasks:

1. Check the home directory permissions.
2. Check the DB2 instance owner profile.
3. Check the DB2 instance symbolic links.
4. Check the DB2 release level.
5. Check the DB2 service name.
6. Check the database manager configuration.
7. Create a sample database.

Check the home directory permissions

Check that the home directory ownership has been correctly set up by the db2setup program, as listed in Table 12-9.

Table 12-9 DB2 home directory required permissions

Home directory path	Owner	Group	Permissions
/home/<db2_instance_owner>	<db2_instance_owner>	db2iadm1	drwxr-sr-x
/home/db2fenc1	db2fenc1	db2fadm1	drwxr-xr-x
/home/db2as	db2as	db2asgrp	drwxr-xr-x

If a DB2-related home directory has not been correctly configured, perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Issue the command, substituting values from the table above:

```
# chown -fR <owner>:<group> <home_directory_path>
```

Check the DB2 instance owner profile

The DB2 Server installation should set up the .bashrc environment file of the <db2_instance_owner> so that the DB2 environment is set up when the user logs in.

1. The following content should have been added to the file:

```
if [ -f /home/<db2_instance_owner>/sqllib/db2profile ] ; then
   . /home/<db2_instance_owner>/sqllib/db2profile
fi
```

2. If not present, manually edit the file to add the above content.

Check the DB2 instance symbolic links

The DB2 server installation automatically creates a DB2 instance <db2_instance_owner> under the /home/<db2_instance_owner> directory. As part of the instance creation, db2setup should create symbolic links in the /home/<db2_instance_owner>/sqllib directory to files under /usr/IBMdb2/V7.1.

Perform the following steps to check whether the symbolic links have been created:

1. Log in as root.
2. Start a terminal session.
1. Change the directory to /home/<db2_instance_owner>/sqllib.
2. Check whether a number of symbolic links exist pointing to files under /usr/IBMdb2/V7.1.
3. If not, issue the following commands:

```
# cd /usr/IBMdb2/V7.1/cfg
# ./db2ln
```

Check the DB2 release level

Check that DB2 has the correct internal release level to meet WebSphere requirements:

1. Change to user <db2_instance_owner>:

```
# su - <db2_instance_owner>
```

2. Enter the following command:

```
$ db2level
```

   This should generate output similar to the following:

```
DB21085I  Instance "db2inst1" uses DB2 code release "SQL07021" with level
identifier "03020105" and informational tokens "DB2 v7.1.0.43", "s010504"
and "U475381a"
```

3. An internal release level of 7.1.0.43 should be indicated.

Check the DB2 service name

Check the service name recorded in the TCP/IP services file:

1. Open the /etc/services file, and locate the entries that have comments referring to the DB2 instance connection port.
2. Locate the service name in the first column that corresponds to the lower port number. For example, if the following services were displayed:

```
db2cdb2inst1 50000/tcp #Connection port for DB2 instance db2inst1
db2idb2inst1 50001/tcp #Interrupt port for DB2 instance db2inst1
```

3. Record the db2cdb2inst1 service name for later use.

Check the database manager configuration

Check that the service name is recorded in the database manager configuration:

1. Change to user <db2_instance_owner>:

   ```
   # su - <db2_instance_owner>
   ```

2. Enter the following command:

   ```
   $ db2 get dbm cfg | grep SVCENAME
   ```

3. Verify that the SVCENAME value matches the service name recorded above from the services file. For example, something similar to the following should be displayed:

   ```
   TCP/IP Service name (SVCENAME)=db2cdb2inst1
   ```

4. If the value does not match, update the database manager configuration using the following commands:

   ```
   $ db2 update dbm cfg using svcename <service_name>
   $ db2stop
   $ db2start
   ```

   Where <service_name> must be replaced with the service name.

Create a sample database

The DB2 installation can be tested by creating and connecting to the sample database supplied with DB2 specifically for this purpose:

1. Change to user <db2_instance_owner>:

   ```
   # su - <db2_instance_owner>
   ```

2. Create the sample database:

   ```
   $ db2sampl
   ```

3. Check that DB2 knows about the new database:

   ```
   $ db2 list db directory
   ```

   This should give output containing the following:

   ```
   Database 1 entry:
    Database alias                 = SAMPLE
    Database name                  = SAMPLE
    Database drive                 = /home/db2inst1
    Database release level         = 9.00
    Comment                        =
    Directory entry type           = Indirect
   ```

```
 Catalog node number              = 0
```

4. Test connectivity to the database:

```
$ db2 connect to sample
$ db2 diconnect current
```

12.4.4 Configure the DB2 server

After the DB2 server installation, a number of configuration tasks must be performed so that WebSphere is able to use it as the repository for its administrative database:

1. Update the root administrative groups.
2. Update the JDBC level.
3. Configure the TCP/IP communication mode.
4. Verify the DB2 environment.
5. Update the root environment file.

Update the root administrative groups

The DB2 server installation should add the following administrative group to the root user:

- db2asgrp

Perform the following steps to check whether the root account's administrative groups have been amended:

1. Log in as root.
2. Start a terminal session.
3. Issue the following command:

```
# groups
```

4. If db2asgrp is not listed as one of the groups assigned to root, use the Red Hat tools to reconfigure the root user.

Update the JDBC level

IBM WebSphere Application Server V4.0 requires the use of JDBC2.0, whereas the default installation of IBM DB2 uses JDBC1.2. To update the DB2 JDBC level, complete the following steps:

1. Change user to <db2_instance_owner>:

```
# su - <db2_instance_owner>
```

2. Add the following content to the end of the <db2_instance_owner> .bashrc environment file:

```
if [ -f /home/<db2_instance_owner>/sqllib/java12/usejdbc2 ] ; then
   . /home/<db2_instance_owner>/sqllib/java12/usejdbc2
fi
```

Configure the TCP/IP communication mode

The DB2 Server may need to be reconfigured to use TCP/IP as its primary communication method:

1. Change user to <db2_instance_owner>:

   ```
   # su - <db2_instance_owner>
   ```

2. Check whether TCP/IP is the current DB2 communication method. The following command should return a value of tcpip:

   ```
   $ db2set DB2COMM
   ```

3. If not, reset the DB2COMM DB2 environment variable:

   ```
   $ db2set DB2COMM=tcpip
   ```

Verify the DB2 environment

After the above configuration steps, we need to check that the environment being set up by the db2profile and usejdbc2 scripts is correct:

1. Change to user <db2_instance_owner>:

   ```
   # su - <db2_instance_owner>
   ```

2. Issue the following command:

   ```
   # set | grep [Dd][Bb]2
   ```

3. Check that the environment variables in this output match the values in Table 12-10.

Table 12-10 DB2 server required environment variables

Environment variable	Required value
DB2DIR	/usr/IBMdb2/V7.1
DB2INSTANCE	<db2_instance_owner>
INSTHOME	/home/<db2_instance_owner>
LD_LIBRARY_PATH	:/home/<db2_instance_owner>/sqllib/lib
LIBPATH	.....:/home/<db2_instance_owner>/sqllib/java12:/home/<db2_instance_owner>/sqllib/lib
CLASSPATH	/home/<db2_instance_owner>/sqllib/function:/home/<db2_instance_owner>/sqllib/java12/db2java.zip:/home/<db2_instance_owner>/sqllib/java/runtime.zip

Environment variable	Required value
PATH	.....:/home/<db2_instance_owner>/sqllib/java12:/home/<db2_instance_owner>/sqllib/bin:/home/<db2_instance_owner>/sqllib/adm:/home/<db2_instance_owner>/sqllib/misc

Update the root environment file

The WebSphere Application Server will be run under root and will require access to the DB2 environment so that it can access the WebSphere administrative database. This requires that the root account's environment .bashrc file be edited to add the following content at the end of the file.

```
# Setup DB2 environment for root user.
if [ -f /home/<db2_instance_owner>/sqllib/db2profile ] ; then
   . /home/<db2_instance_owner>/sqllib/db2profile
fi

# Force DB2 to use JDBC 2.0.
if [ -f /home/<db2_instance_owner>/sqllib/java12/usejdbc2 ] ; then
   . /home/<db2_instance_owner>/sqllib/java12/usejdbc2
fi
```

12.4.5 Set up the WebSphere administrative database

Next, set up a database in DB2 to use as the WebSphere administrative repository. The database will be populated with WebSphere schema and default values in a later task.

To set up the WebSphere database, complete the following steps:

1. Log in as <db2_instance_owner>.
2. Create the WebSphere database and configure its heap size to suit WebSphere requirements:

   ```
   $ db2 create db was1
   $ db2 update db cfg for was1 using applheapsz 256
   ```

   Note: Although the repository database created here is called was1, any valid DB2 database name can be used.

3. Check that the new database is known to DB2:

   ```
   $ db2 list db directory
   ```

This should give output containing the following:

```
Database 1 entry:
 Database alias                 = WAS1
 Database name                  = WAS1
 Database drive                 = /home/db2inst1
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Indirect
 Catalog node number            = 0
```

4. In order to access the administrative database via TCP/IP, the DB2 node must first be cataloged:

```
$ db2 catalog tcpip node <node_name> remote <local_hostname> server
<service_name>
```

Important: The <service_name> used to catalog the node must be the same as the database instance connection port name in the /etc/services file. The <node_name> chosen can be any valid DB2 nodename.

5. The administrative database must now be cataloged as part of this TCP/IP node:

```
$ db2 catalog db was1 as was at node <node_name>
```

6. Check that the database TCP/IP alias is known to DB2:

```
$ db2 list db directory
```

This should give output containing the following:

```
Database 2 entry:
 Database alias                 = WAS
 Database name                  = WAS1
 Node name                      = <node_name>
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Remote
 Catalog node number            = -1
```

7. Verify the connection to the local database via TCP/IP:

```
$ db2 connect to was user <db2_instance_owner> using <db2owner_passwd>
$ db2 disconnect current
```

where <db2owner_passwd> is the DB2 instance owner password on the local DB2 server.

Tip: When the DB2 administration account is used to create a new database, it is automatically granted dba access rights. You need to specifically grant access to another user only if you plan to access the database from an account other than <db2_instance_owner>.

12.5 Install WebSphere Application Server

This section provides detailed instructions for installing, configuring, and verifying WebSphere Application Server V4.0, Advanced Edition for Linux.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install WebSphere.
3. Verify the WebSphere installation.

12.5.1 Preinstallation tasks

Prior to installing IBM WebSphere Application Server V4.0, the following checks and tasks need to be completed:

1. Check that IP ports are unused.
2. Stop the Web server processes.

Check that IP ports are unused

To check that the required ports are not in use, perform the following steps:

1. Check that there are no existing active services that use the following IP ports on the server:
   - 900 (bootstrap port)
   - 9000 (Location Service Daemon)
   - 9080 (default application server)

   We suggest using the following command for this task:

```
$ netstat -a | grep LISTEN
```

Stop the Web server processes

The IBM HTTP Server process must be stopped while WebSphere is installed. The WebSphere installation changes the httpd.conf configuration file as part of the Web server plug-in component installation.

1. Log in as root on the IBM HTTP Server machine.

2. Start a terminal session.
3. Issue the following commands:

```
# cd <http_server_install_path>/bin
# ./apachectl stop
```

12.5.2 Install WebSphere

To install IBM WebSphere Application Server V4.0, Advanced Edition using the GUI installer interface, complete the following steps on the WebSphere server machine:

Tip: The WebSphere installer (install.sh) also provides a non-GUI scripted or "silent" mode of operation. See 12.7, "Install WebSphere Application Server - silent mode" on page 435 for details.

1. Log in as root.
2. Start a terminal session.
3. Load the IBM WebSphere Application Server V4.0, Advanced Edition CD-ROM into the CD-ROM drive and mount the CD.

```
# mount -t iso9660 -r /dev/cdrom /cdrom
```

4. Change the directory to the installation root.
5. Ensure the DISPLAY and TERM environment variables are properly set.
6. Run the install.sh installation script:

```
# ./install.sh
```

7. In the Welcome window, click **Next**.

Important: If prerequisite checking is enabled for UNIX, then an alert may be displayed indicating that some of the installation prerequisities have not been met. This can even occur if newer patches or packages than those listed in prereq.properties are installed. If such an alert is displayed, recheck all prerequisites, and if they are met or exceeded, perform the following steps:

- Exit the installation.
- Disable prerequisite checking.
- Restart the installation from the beginning.

8. In the Installation Options window, select **Custom Installation** and click **Next**.

9. In the Choose Application Server Components window, choose all options except IBM HTTP Server, then click **Next**.

Important: Although not listed in the Application Server Components window, the IBM JDK 1.3.0 is automatically installed under the WebSphere installation directory. There is no need to separately install a JDK for use by:

- WebSphere Application Server
- Web server plug-ins

10. In the Choose Webserver Plugin window, choose only the **IBM HTTP Server Plugin**, then click **Next**.
11. In the Database Options window, shown in Figure 12-2, enter the following then click **Next**.
    - Database Type: DB2
    - Database Name (Database SID): <was database alias>

      Enter the DB2 TCP/IP alias of the WebSphere administrative database, as created in 12.4.5, “Set up the WebSphere administrative database” on page 423.
    - DB Home: /home/<db2_instance_owner>
    - Database User ID: <db2_instance_owner>
    - Database Password: <db2owner_password>
    - Remote DB: leave unselected

Note: Whether the DB2 database is local or remote, in our test environment we configure the DB2 client to access the database through a catalog alias. Under these conditions, the Remote DB setting should not be selected.

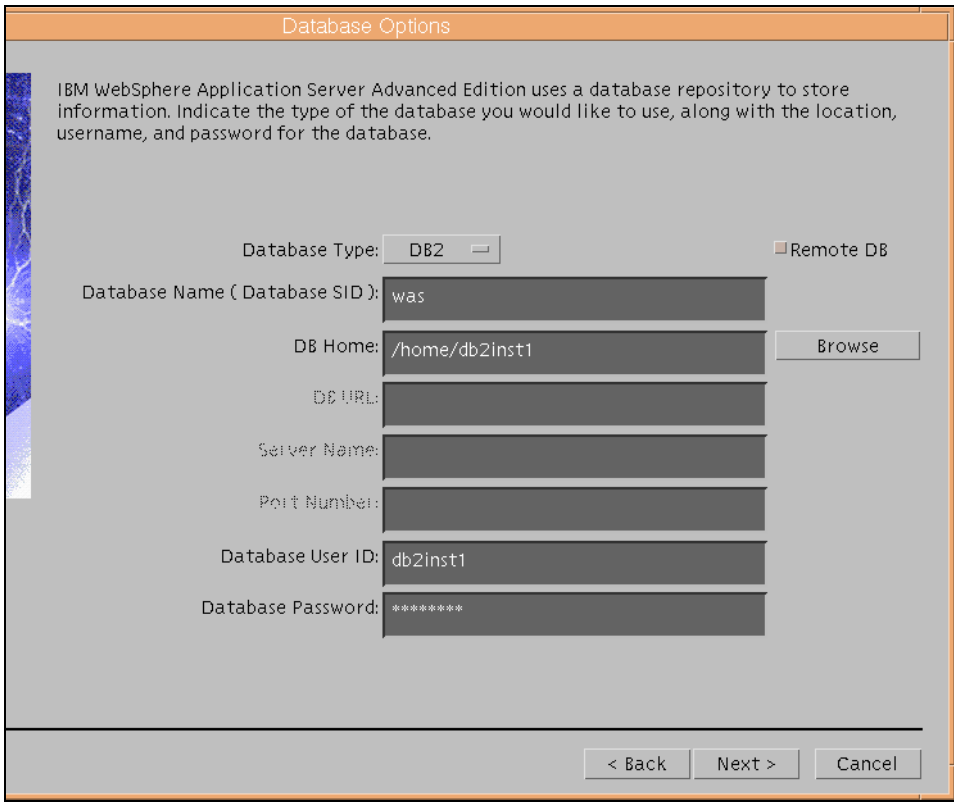


Figure 12-2 Specify DB2 database connection settings

12. In the Select Destination Directory window, accept the default location for the WebSphere Application Server (/opt/WebSphere/AppServer). Click **Next** to continue.
13. In the Install Options Selected window, check that the correct components have been selected. If yes, click **Install** to start the installation.
14. In the Location of Configuration Files window, enter the path to the IBM HTTP Server configuration file (httpd.conf), then click **Next**.
15. In the Setup Complete window, click **Finish**. The installation of the WebSphere Application Server is now complete.

12.5.3 Verify the WebSphere installation

In order to verify the installation of IBM WebSphere Application Server V4.0, Advanced Edition, the following tasks must be completed in order:

1. Check the installation log.

2. Check the admin.config settings.
3. Check the Web server configuration file changes.
4. Start the WebSphere administrative server processes.
5. Start the WebSphere Default Server.
6. Regenerate the Web server plug-in settings.
7. Restart the Web server processes.
8. Verify the Web server plug-in configuration.

Check the installation log

Check that the installation log <WAS_HOME>/logs/install.log does not contain any errors.

Check the admin.config settings

To check the admin.config settings, perform the following steps:

1. Check that the repository database settings are correct for the database type (DB2), instance (was) and user ID (<db2_instance_owner>) used in our test environment:

```
com.ibm.ejs.sm.adminServer.dbdataSourceClassName=COM.ibm.db2.jdbc.DB2Connec
tionPoolDataSource
com.ibm.ejs.sm.adminServer.dbserverName=
com.ibm.ejs.sm.adminServer.dbportNumber=
com.ibm.ejs.sm.adminServer.dbdatabaseName=<was database alias>
com.ibm.ejs.sm.adminServer.dbuser=<db2_instance_owner>
com.ibm.ejs.sm.adminServer.dbpassword=<db2owner_password>
com.ibm.ejs.sm.adminServer.dbdisable2Phase=true
```

2. Check the DB2 path-related parameter listed in Table 12-11.

Table 12-11 DB2-related path required in admin.config

Parameter	Must contain path...
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs	<db2_install_path>/sqllib/java12/db2java.zip

3. Check that the WebSphere schema and initial configuration (for example, the Default Server) will be written to the repository database on startup, as listed in Table 12-12.

Table 12-12 Required schema creation and initial configuration flags

Parameter	Required value...
com.ibm.ejs.sm.adminServer.createTables	true
install.initial.config	true

Check the Web server configuration file changes

To check the Web server configuration file changes, complete the following steps:

1. Check that following required settings have been added to the IBM HTTP Server configuration file (httpd.conf) as a result of the WebSphere installation:

```
LoadModule ibm_app_server_http_module
/opt/WebSphere/AppServer/bin/mod_ibm_app_server_http.so
WebSpherePluginConfig /opt/WebSphere/AppServer/config/plugin-cfg.xml
AddModule mod_app_server_http.c
```

2. If not, manually add the above lines to the end of the httpd.conf file and save the changes.

Start up the WebSphere administrative server processes

The WebSphere administrative server needs to be started in order to test the installation as well as the connectivity between WebSphere and the WebSphere administrative database hosted in DB2:

1. Log in as root.
2. Start a terminal session.
1. Run the WebSphere administrative server by issuing the following commands:

```
# cd <WAS_HOME>/bin
# ./startupServer.sh
```

2. The startup of WebSphere administrative server is successful if the following conditions are met:
   a. There are no administrative server error logs in <WAS_HOME>/logs with names that start with __adminServer.
   b. The last line of the <WAS_HOME>/logs/tracefile file is similar to the following:

```
[01.07.05 11:28:01:591 EDT] 160042d2 Server        I WSVR0023I: Server
__adminServer open for e-business
```

Start up the WebSphere Default Server

The WebSphere installation sets up a default application server (Default Server) in the WebSphere administrative domain. We use this application server and its Web applications to verify that the WebSphere installation is working correctly.

To start up the Default Server, perform the following steps:

1. Run the WebSphere Administrative Console by issuing the following commands:

```
# cd <WAS_HOME>/bin
# ./adminclient.sh
```

2. Right-click the **Default Server** under the <hostname> node and select **Start** from the pop-up menu.
3. The Default Server has been successfully started if the following conditions are met:
   a. The administrative console event messages pane shows the lines:

      ```
      Transport http listening on port 9,080.
      Command "Default Server.start" completed successfuly.
      ```

   b. The last line of the <WAS_HOME>/logs/Default_Server_stdout.log file is similar to the following:

      ```
      [01.07.03 11:55:03:103 EDT]   6ff4af Server        I WSVR0023I: Server
      Default Server open for e-business
      ```

4. Verify that the Default Server Web container has been properly installed and configured by accessing its servlets through the Web server “embedded” within the WebSphere V4.0 Web container:
   a. Using a Web browser, request the following URL:

      ```
      http://<hostname>:9080/servlet/snoop
      ```

      A window similar to the one shown in Figure 12-3 should be displayed in your browser.

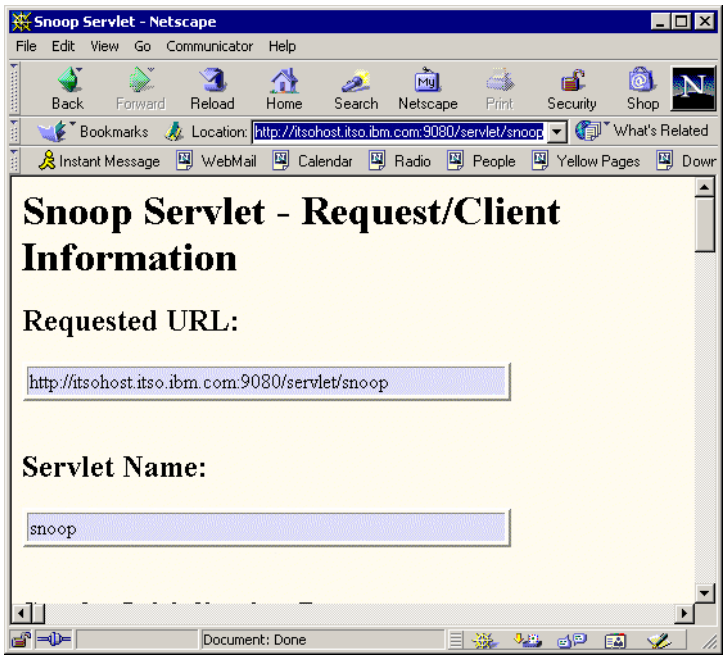


Figure 12-3 Snoop servlet accessed through embedded Web server

b. Using a Web browser, request the following URL:

```
http://<hostname>:9080/webapp/examples/showCfg
```

Note: The embedded Web server is a new feature introduced with WebSphere V4.0. In previous releases, a stand-alone Web server was required in order to access any resource hosted in WebSphere.

5. Close the WebSphere Administrative Console.

Regenerate the Web server plug-in settings

Before the Default Server can be accessed from a stand-alone Web server (such as IBM HTTP Server) the Web server plug-in settings file <WAS_HOME>/config/plugin-cfg.xml must be regenerated to reflect the following settings used by the Web server plug-in:

- Virtual host settings
- Application server transports
- Web container URIs

Perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Run the WebSphere Administrative Console by issuing the following command:

```
# cd <WAS_HOME>/bin
# ./adminclient.sh
```

4. Right-click the node <hostname> that contains the Default Server application server and select **Regen Webserver Plugin** from the pop-up menu, as shown in Figure 12-4.

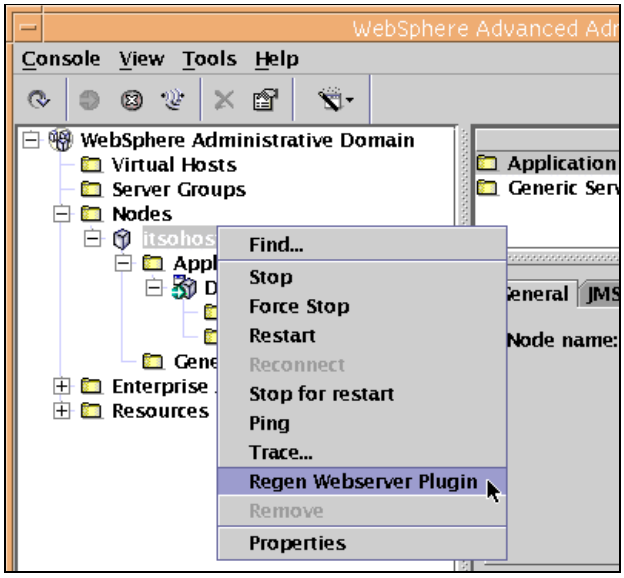


Figure 12-4 Regenerate Web server plug-in settings

Tip: WebSphere provides a command-line tool that can be used to regenerate the Web server plug-in configuration without having to run the WebSphere Administrative Console:

```
<WAS_HOME>/bin/GenPluginCfg.sh -adminNodeName <hostname>
```

5. Check that the content of the <WAS_HOME>/config/plugin-cfg.xml file has been updated to include the URIs of servlets contained within the Default Server.

Tip: The plug-in regeneration command generates a <Server> element for the Default Server that contains a CloneID attribute:

```
<Server CloneID="stsul7n0" Name="Default Server">
   <Transport Hostname="itsohost" Port="9080" Protocol="http"/>
</Server>
```

In a non-cloned environment this attribute can be removed, resulting in performance improvements in the Web server plug-in.

Restart the Web server processes

The IBM HTTP Server process must be restarted before the Web server plug-in configuration can be tested.

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <http_server_install_path>/bin
# ./apachectl restart
```

Verify the Web server plug-in configuration

The Web server plug-in configuration can be verified by requesting a servlet through the Web server that has already been successfully requested through the Web container's embedded Web server:

1. Using a Web browser, request the following URL:

   ```
   http://<web_server_hostname>/servlet/snoop
   ```

   or:

   ```
   http://<web_server_hostname>/webapp/examples/showCfg
   ```

12.6 Configure WebSphere HTTP transport for SSL

See 9.9, "Configure WebSphere HTTP transport for SSL" on page 245 for a description of how to configure a WebSphere HTTP transport to use SSL encryption. Apart from file system path changes, the method used on each platform is the same.

Note: To execute the Key Management Utilities on the Linux platform, perform the following steps:

1. To start the WebSphere Key Management Utility:
   a. Log in as root on the WebSphere server machine.
   b. Start a terminal session.
   c. Execute the following commands:

```
# cd <WAS_HOME>/bin
# ./ikeyman.sh
```

2. To start the Web server IBM Key Management Utility:
   a. Log in as root on the IBM HTTP Server machine.
   b. Start a terminal session.
   c. Execute the following commands:

```
# cd /usr/bin
# ./ikeyman
```

12.7 Install WebSphere Application Server - silent mode

This section describes how to install WebSphere Application Server V4.0, Advanced Edition using the non-interactive, or silent, mode. To complete a silent installation, you will use the default response file or create a customized one, and then execute the installation script for WebSphere Application Server, supplying the response file as a command-line parameter.

These instructions assume the following:

- Your machine has sufficient memory and disk space for your installation.
- You have installed and configured a supported database.
- You do not have a previous version of WebSphere Application Server already installed on this machine. If you do have a previous version of WebSphere Application Server already installed, do not follow these instructions. Instead, see Chapter 25, “Migration” on page 1031.
- If you are using IBM HTTP Server as your Web server, you will install it at the same time and onto the same node as you install WebSphere Application Server. If you are using another supported Web server with WebSphere Application Server, you have already installed it onto the same node as WebSphere Application Server.

Note: IBM HTTP Server is supplied with WebSphere Application Server. If you plan to use a different Web server, you must purchase it and install it separately. It is recommended that the Web server be installed before WebSphere Application Server.

12.7.1 Using the default response file

A default response file, named install.script, is supplied with WebSphere Application Server. You can use this default response file to install WebSphere Application Server using the default options, or as a template for creating a customized response file.

If you use the default response file to install WebSphere Application Server using the default options, the following software and other resources are installed:

- IBM Java 2 Software Developer's Kit (SDK) 1.3.0
- IBM HTTP Server V1.3.19
- IBM WebSphere Application Server V4.0
- WebSphere Application Server application samples
- Documentation in U.S. English

Note: All products except IBM HTTP Server are installed into the directory /opt/WebSphere/AppServer. IBM HTTP Server is installed into the directory /opt/IBMHTTPServer. In addition, WebSphere Application Server is configured for use with IBM HTTP Server.

12.7.2 Using a customized response file

You can also use the default response file as a template for creating a customized response file. You can edit the default response file to enable the configuration of WebSphere Application Server with a different supported Web server or database or to install the products into a different directory. Detailed comments within the default response file guide you through the installation and configuration options available for performing a silent installation.

12.7.3 Performing a silent installation

Perform the following steps to create a customized response file (if desired) to install WebSphere Application Server. These instructions assume that the installation is being performed from the product CD-ROM:

1. Ensure that you are logged into the machine with superuser (root) privileges.

2. If a Web server is running on your system, stop the Web server. If you plan to install IBM HTTP Server 1.3.19 as part of the WebSphere Application Server installation and you have a level of IBM HTTP Server prior to 1.3.19 on your system, you must uninstall it for the WebSphere Application Server installation program to install IBM HTTP Server 1.3.19.
3. Insert the WebSphere Application Server CD-ROM into the CD-ROM drive.
4. If necessary, use the **mkdir** command to create a mount point for the CD-ROM. The following command creates a mount point at the directory /cdrom. You can mount the CD-ROM at any location on the machine's local file system.

   ```
   # mkdir /cdrom
   ```

   The commands in these steps assume the CD-ROM is mounted at /cdrom. If you mount the CD-ROM at a different location, use that location when issuing commands.
5. Mount the CD-ROM drive by entering the following command:

   ```
   # mount -t isso9660 -r /dev/cdrom /cdrom
   ```

   Note: Some window managers automatically mount a CD-ROM for you. Consult your operating system documentation for more information.

6. Ensure that the DISPLAY and TERM environment variables set correctly.
7. Navigate to the /cdrom directory.
8. Ensure that you are in the /cdrom directory and create a copy of the default response file by using the **cp** command, as follows:

   ```
   # cp install.script <new_install.script>
   ```

   In this command, <new_install.script> represents the full path name of the copy of the default response file you are creating (for example, /tmp/my_install.script). The name of your response file must have a .script extension.
9. If you plan to install WebSphere Application Server by using the default options included in the default response file, proceed to the next step.

   If you plan to use the default response file as a template for creating a customized response file, perform the following steps:

   a. Use a text editor to open your copy of the default response file, <new_install.script>.
   b. Use the detailed comments throughout the file to help you select the appropriate options for your WebSphere Application Server installation.
   c. Save the changes that you have made to the customized response file.

10. Run the installation script by using the following commands. The install.sh script uses the response file to install the components and options that you have selected. The variable <new_install.script> represents the full path name of the copy of the default response file or the customized response file that you have created (for example, /tmp/new_install.script).

    ```
    # ./install.sh -silent -responseFile <new_install.script>
    ```

    If you choose to install the plug-in for IBM HTTP Server, the installation process checks if you have the correct version of the Web server on your machine. If you do not have IBM HTTP Server installed on your machine, the installation process performs one of the following actions based on whether you have indicated in your response file to install IBM HTTP Server:

    – If you indicated in your response file that you do want to have IBM HTTP Server installed, the installation process installs the plug-in for it.

    – If you indicated in your response file that you do not want to have IBM HTTP Server installed, the script exits without installing the plug-in.

11. After installation is complete, refer to the log file named install.log located in the /tmp directory to determine if the silent installation was successful. A copy of this file also exists in the directory <WAS_HOME>/logs.

12. Unmount the CD-ROM before removing it from the CD-ROM drive by using the **umount** command, as follows:

    ```
    # umount /cdrom
    ```

13. If you installed IBM HTTP Server as part of the WebSphere Application Server silent installation, you might need to configure it. Perform the following steps to verify that the IBM HTTP Server is installed correctly:

    a. Ensure that the Web server is running or start it by entering the following command:

       ```
       # /opt/IBMHTTPServer/bin/apachectl start
       ```

    b. Start a Web browser and type the name of the host machine as the URL (`http://host_machine`). If you see the Welcome to the IBM HTTP Server Web page, the server has been installed correctly.

See the “Using silent installation on Linux (Intel)“ article in the InfoCenter for further details.

Part 4

Configuring WebSphere

In this part we guide you through WebSphere Administrative Console configuration tasks. We also provide examples of how to package and deploy your J2EE enterprise applications.

13

WebSphere administration basics

In this chapter we introduce the WebSphere Administrative Console and describe some of the basic tasks that are commonly performed by WebSphere administrators. The tasks we look at include:

- Working with nodes, applications servers and enterprise applications
- Viewing installed enterprise application properties, including servlet URLs
- Regenerating the Web server plug-in configuration
- Saving and restoring your WebSphere configuration
- Checking product versions

13.1 Introducing the WebSphere Administrative Console

The WebSphere Administrative Console, shown in Figure 13-1 on page 444, is the graphical Java-based client that administrators use to configure and control a WebSphere domain. A WebSphere domain is made up of one or more physical machines sharing a single WebSphere administrative database, which holds information on each of the components running on those machines. The administrative console doesn't connect to the administrative database directly. It connects to the WebSphere administrative server which accesses the database. It is the administrative server that controls the WebSphere environment on a given node. In order for the administrative console to control multiple nodes in a domain, the administrative server must be running on each node.

There have been major changes to the WebSphere Administrative Console between WebSphere V3.5 and WebSphere V4.0. In WebSphere V4.0, the administrative console has been reorganized to reflect the structure of J2EE applications. To effectively manage a WebSphere V4.0 environment, the administrator should be familiar with the concepts underlying a J2EE runtime environment, particularly:

- The separation of roles between development, deployment, and administration
- Containers
- J2EE enterprise applications
- Web modules
- EJB modules
- XML deployment descriptors
- Resources

We recommend that you have at least an overall understanding of the these concepts before trying to administer a WebSphere V4.0 environment. You can do this by:

- Referring to the J2EE overview in Chapter 3, "The Java 2 platform" on page 29
- Visiting the Sun J2EE Web site:

  `http://java.sun.com/j2ee`
- Reading the J2EE specification, see:

  `http://java.sun.com/j2ee/download.htm`

Many of the properties that were once visible on the console have been moved to XML deployment descriptors for enterprise applications, and for Web and EJB modules. Instead of seeing individual servlets, JSPs or EJBs as one did using the WebSphere V3.5 Administrative Console, the WebSphere V4.0 Administrative Console focuses on enterprise applications and the Web and/or EJB modules that they contain. These modules are assigned to application servers or clones that provide the runtime containers for the modules. It is the modules that contain the individual servlets, JSPs and EJBs.

The servlets, JSPs and EJBs are not displayed directly on the console. You can see these details in the administrative console by displaying the XML deployment descriptors for the enterprise application, Web modules and EJB modules. For further information, see 13.2.5, “Viewing installed applications” on page 467.

The administrative console also allows the administrator to change the parameters for deployed applications. However, if the application or module is reinstalled, these changes may be overridden by the XML deployment descriptor values.

Note: There are command-line utilities that can perform many, if not all, of the tasks that the administrative console can carry out. See Chapter 23, “Command-line administration and scripting” on page 881.

13.1.1 The graphical interface

The WebSphere Administrative Console GUI, shown in Figure 13-1 on page 444, consists of the following views:

- Tree view
- Details view
- Properties view
- Messages view

An additional view is provided by the Runtime Inspector, shown in Figure 13-3 on page 446. This window extends the properties view with runtime values.

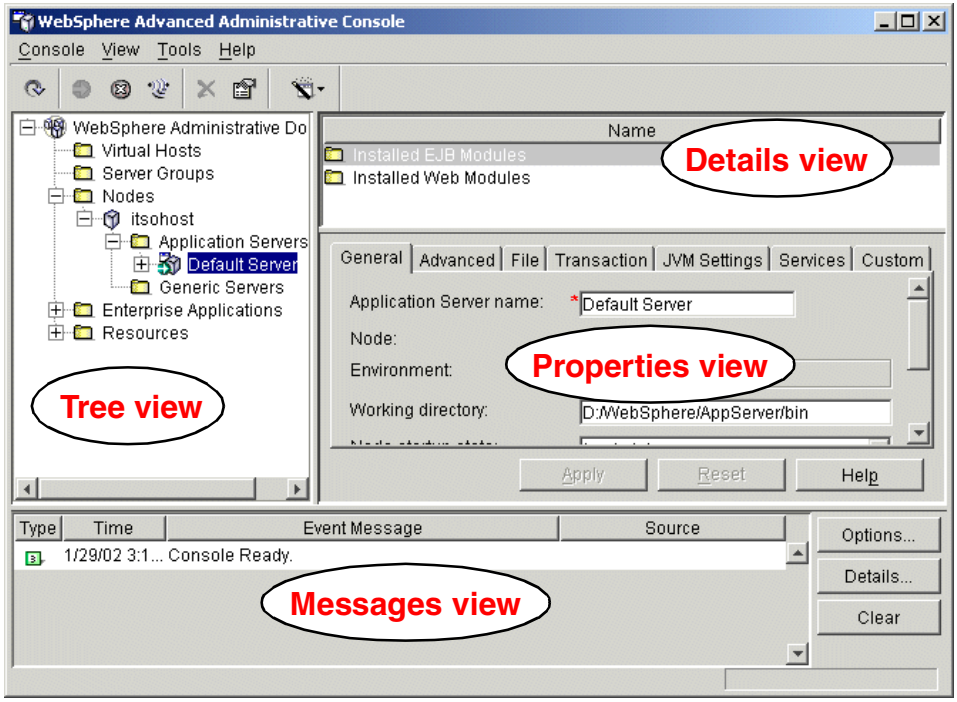


Figure 13-1 WebSphere Administrative Console views

The task bar, shown in Figure 13-2, provides fast access to commonly used actions. Actions from the task bar are refresh, start, stop, ping, remove, properties, and wizards.

Figure 13-2 WebSphere Administrative Console task bar

Tree view

You use the tree view on the left side of the console to survey, select, and manage components in the WebSphere administrative domain.

Clicking a "+" beside a tree folder or item expands the tree for the folder or item. Similarly, clicking a "-" collapses the tree for the folder or item. Double-clicking an item in the tree view toggles its state between expanded and collapsed.

Clicking (selecting) a folder or item in the tree view controls the content displayed on the right side of the console. Selecting a folder or item also enables menu actions that an administrator can use with the selected folder or item.

Details view

The details view on the upper-right side of the console displays information on the selection in the tree view. When you click a folder in the tree view, the details view lists information on instances of that folder type. When an item in the details view is selected, the properties for the item are shown below the details view, provided the properties view is visible.

Double-clicking an item in the details view expands a collapsed tree view to show the item. Double-clicking an item in the details view also selects that item in the tree view and populates the details view with its sub-items.

Properties view

The properties view on the lower-right of the console displays the properties sheet for an item selected in the details view or in the tree view. You use a properties sheet to view and edit property values.

Note that the properties view is shown only when the **Show Property Pane** toggle on the console View menu is enabled (the default).

Also note that a properties sheet is like a properties window. To access a properties window, you right-click an item in the tree view and select **Properties** from the pop-up menu.

Further, a properties sheet does not display runtime (in use) property values. To see runtime property values use the Runtime Inspector, which is available by clicking **View** -> **Show Runtime Inspector**.

Messages view

The messages area at the bottom of the console lists messages returned by the WebSphere administrative server as well as messages about events such as successful completions and fatal errors.

You can customize the contents of the message list and limit the message log size by clicking the **Options** button and setting message preferences in the window that opens. For further details see 13.1.9, “Filtering messages” on page 455.

Note that the messages area is shown only when the **Show Console Messages** toggle on the console View menu is enabled (the default).

Runtime Inspector

To inspect the runtime (in use) values of properties, use the Runtime Inspector. The Runtime Inspector lists attributes of a property selected in the administrative console and their runtime values. Examples of runtime values include a process ID or classpath for an application server.

The inspector is a separate window that stays on top of other console windows.

The contents of the inspector update when a different item in the console tree view or details view is selected, provided the selection has runtime attributes. Clicking a "+" beside an attribute expands the list of attributes. Similarly, clicking a "-" collapses the list.

The Runtime Inspector is shown when the **Show Runtime Inspector** toggle on the console View menu is enabled.

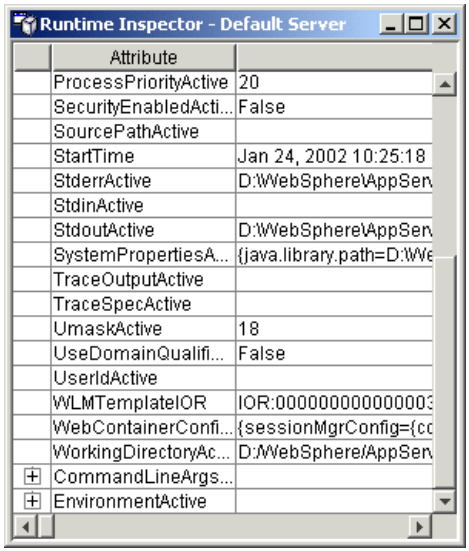


Figure 13-3 Runtime Inspector

13.1.2 Starting and stopping the administrative console

In order to start the WebSphere Administrative Console first make sure your administrative server is running. Otherwise the WebSphere Administrative Console will not start.

The WebSphere Administrative Console is a Java process. As such, it must be able to locate a JDK supported by IBM WebSphere Application Server V4.0, as specified on the prerequisites Web page:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

Furthermore it is necessary to have the same fix pack level on both server and client sides. For information on how to check WebSphere and JDK versions, see 13.2.10, “Checking versions” on page 478.

On Windows platforms

To start the WebSphere Administrative Console do one of the following:

- Click **Start** -> **Programs** -> **IBM WebSphere** -> **Application Server 4.0** -> **Administrator’s Console**.
- At the command prompt, type:

```
cd <WAS_HOME>\bin
.\adminclient.bat
```

To stop the WebSphere Administrative Console, select **Console** -> **Exit** from the console main menu.

On UNIX platforms

To start the WebSphere Administrative Console, start a terminal session and enter the following commands:

```
$ cd <WAS_HOME>/bin
$ ./adminclient.sh
```

To stop the WebSphere Administrative Console, select **Console** -> **Exit** from the console main menu.

Note: Ensure that the DISPLAY environment variable is set in the shell from which the administrative client script will be run:

```
$ export DISPLAY=localMachineName:0.0
```

Connecting to a remote system

If your WebSphere administrative server is not on your local machine, then you have to specify the host name and port number in order to connect to the remote system. If the port number is 900, then it can be left out because it is the default. The command-line format is as follows:

```
adminclient [host_name] [port_number]
```

For example, if your host name is `adminhost` and the bootstrap port `900`, use the following command to start the administrative console on a Windows platform:

```
adminclient adminhost 900
```

Connecting to a secured environment

If security is enabled for your WebSphere administrative domain, then you will get a login window as shown in Figure 13-4. To log in, enter a valid user identity and password, and click **OK**.

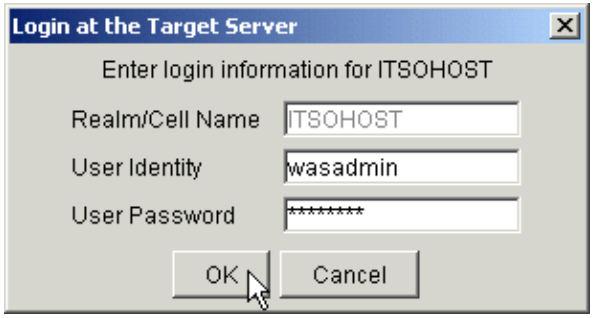


Figure 13-4 WebSphere Administrative Console login window

You can edit the sas.client.properties files to avoid being prompted. Please refer to the procedure described for WebSphere Control Program (WSCP) in “Connecting to a secure administrative server” on page 890.

Note: By default, WebSphere Security is disabled. This means that any user can use a local administrative console to connect the administrative server. Once they are connected they have complete control over the WebSphere configuration, they can start or stop application servers, add nodes, and so on.

Care should be take to avoid this problem by enabling WebSphere Administrative Console password checking. In WebSphere V4.0, this can be enabled without turning on security for all of the applications running in WebSphere. See 21.1.6, “Securing only the administrative server” on page 750.

13.1.3 Starting and stopping items

To start or stop an item, do the following:

1. Highlight the item either in the tree view or detail view.
2. Do one of the following:
   - Select **Console** -> **Start** or **Stop** from the main menu.
   - Right-click the item and select **Start** or **Stop** from the pop-up menu.
   - Click the start or stop icon in the task bar.

The following start and stop actions are possible:

Start Starts the item.

Restart Stops an item, then returns the item to the state it was in before it (or its parent item) was stopped. For example, suppose you performed Stop for Restart (or just Restart) on a running administrative server node containing application server "A" in the stopped state and application server "B" in the running state. The Restart action would cause application server A to remain stopped. It would return application server B to the running state, along with the administrative server.

Stop Stops the item. For an administrative server node, this action stops the administrative server running on the node. It does not shut down the operating system. If you do this on the node that the administrative console is attached to, the administrative console will shut down as well.

Force Stop Stops the item when the regular Stop action is ineffective. Use the option when the administrative server or other item is in a strange state and cannot be stopped using the ordinary Stop action.

Stop for Restart Stops an item and any items contained by the item, in preparation for the Restart action. It can be useful for stopping and restarting an administrative server that contains multiple application servers, some of which are running and others of which are stopped. Using a combination of Stop for Restart and then Restart will return the application servers to their former states (stopped or running) when the administrative server is restarted.

Note: Not all items can be started or stopped. Depending on the item type, all, some, or none of the actions might be available. Options could be unavailable (grayed out or omitted entirely) for items that cannot be started or stopped, or are presently in the wrong state to be started or stopped. For example, trying to stop an already stopped item is not a valid action.

For example, use the following steps to start the Default Server from the pop-up menu:

1. Expand the WebSphere administrative domain tree to see the Default Server.
2. Right-click the **Default Server**.
3. Select **Start** from the pop-up menu, as shown in Figure 13-5.

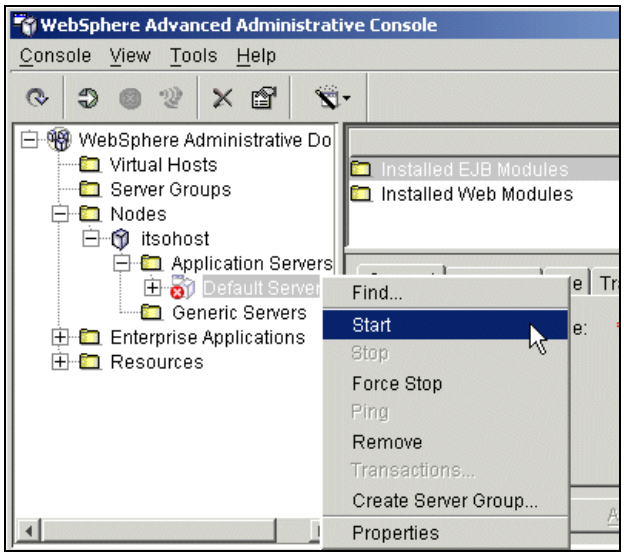


Figure 13-5 Start the Default Server from the pop-up menu

Alternatively, simply select the **Default Server** and click the start icon on the task bar, as shown in Figure 13-6.

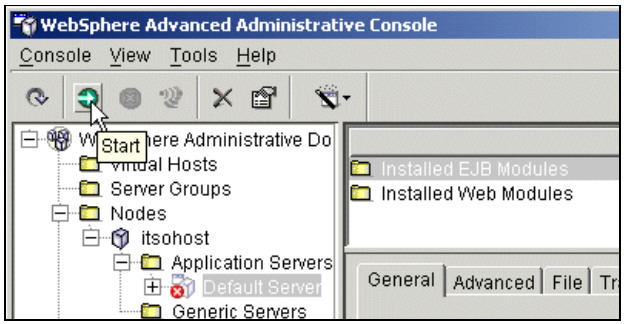


Figure 13-6 Start the Default Server from the task bar

13.1.4 Updating existing items

To edit the properties of an existing item, select the item either in the tree view or detail view. The properties of the selected item will then be shown in the properties view. (Make sure that **View** -> **Show Property Pane** is enabled.)

To view the properties of the selected item in a new window, do one of the following:

- Select **Console** -> **Properties** from the main menu.

- Right-click the item and select **Properties** from the pop-up menu.
- Click the properties icon in the task bar.

For example, use the following steps to change the Java memory settings for the Default Server in the properties view:

1. Expand the WebSphere administrative domain tree to see the Default Server.
2. Highlight the Default Server.
3. On the properties pane, select the **JVM Settings** tab.
4. Enter the required Java heap size settings, as shown in Figure 13-7.
5. Click **Apply**.

Note that you need to restart the application server in order for the changes to take effect.

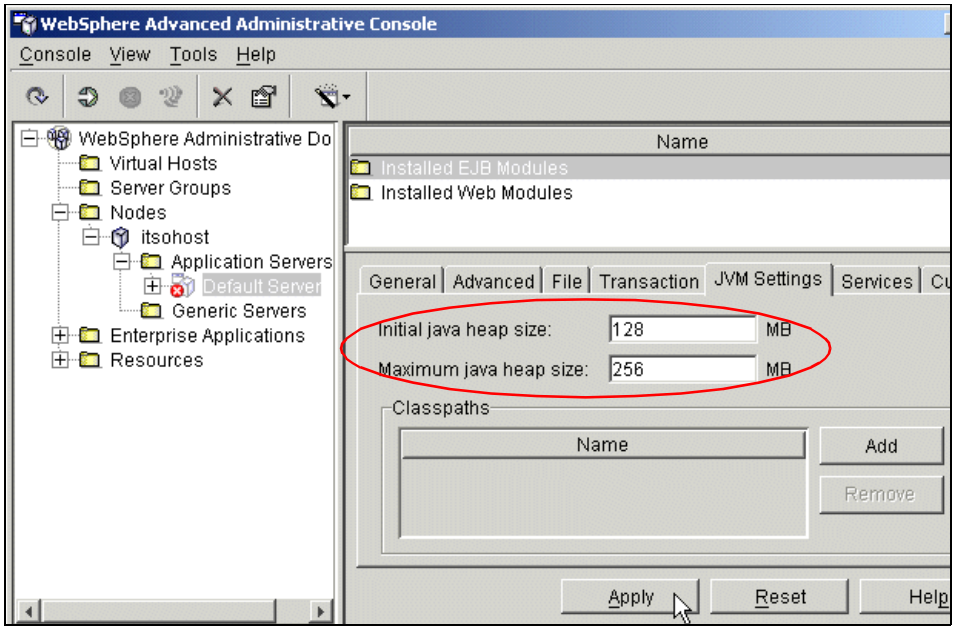


Figure 13-7 Setting the JVM heap size for Default Server

13.1.5 Adding new items

To configure new instances of most item types, you can also use the wizards available from the **Console** -> **Wizards** menu.

To create and configure new items, do one of the following:

- Start the wizard for the appropriate item. This can be done by one of the following actions:
  - Select **Console** -> **Wizards** from the main menu, then select the wizard for the item you want to create.
  - Click the wizards icon in the task bar, then select the wizard for the item you want to create from the drop-down list.
- Select **Console** -> **New** from the main menu, then select the item you want to create.
- Right-click the item folder in the tree view and select **New...** from the pop-up menu.

For example, use the following steps to create a new application server:

1. Expand the WebSphere administrative domain tree.
2. Right-click the **Application Servers** folder under the required node and select **New...** from the pop-up menu, as shown in Figure 13-8.

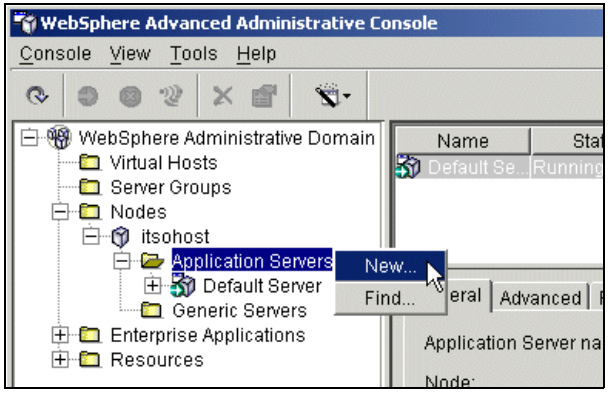


Figure 13-8 Create a new application server

3. In the Create Application Server window, set the required properties for the new application server. See 13.2.3, “Creating an application server” on page 461 for an example of setting up the basic properties of a new application server.
4. Click **OK**.

13.1.6 Removing items

To remove an item:

1. Select the item in the tree view or details view.

2. Do one of the following:
   – Right-click the item and select **Remove** from the pop-up menu.
   – Select **Console** -> **Remove** from the main menu.

If an item does not have a Remove menu option, it cannot be removed.

For example, use the following steps to remove an application server:

1. Select the application server to remove.
2. Right-click the application server and select **Remove** from the pop-up menu, as shown in Figure 13-9.
3. Click **OK** in the confirmation window.

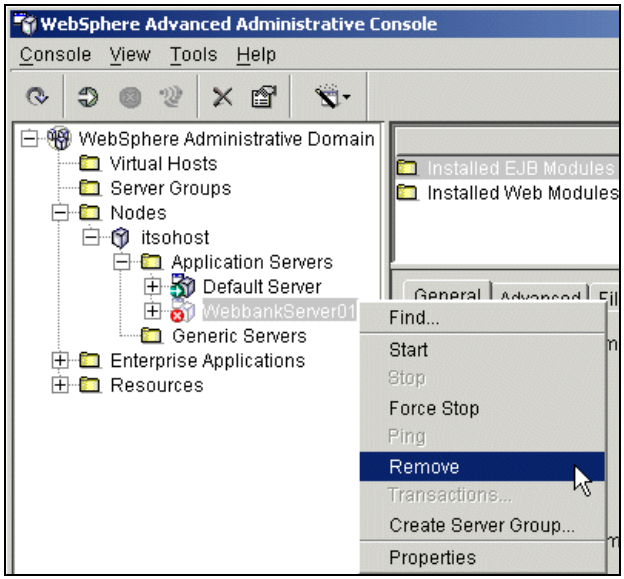


Figure 13-9 Select the application server to remove

13.1.7 Pinging items

Ping an item to see whether the item is responding. The ping action provides a basic "health check" of the item.

To ping an item:

1. Select the item from the tree view or details view.
2. Do one of the following:
   – Select **Console** -> **Ping** from the main menu.

– Right-click the item and select **Ping** from the pop-up menu.

– Click the ping icon in the task bar.

For example, to ping the node itsohost, do the following:

1. Select the node itoshost.
2. Click the ping icon on the task bar. You will get the success message if the node is responding, as shown in Figure 13-10.

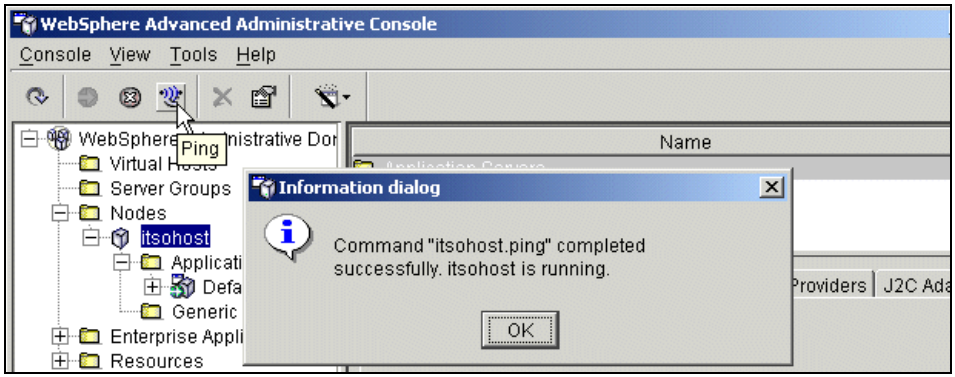


Figure 13-10 Pinging a node

13.1.8 Finding items

To locate items of the same type within the administrative domain:

1. Select **Console** -> **Find...** from the main menu to open the Find Objects window. See Figure 13-11 on page 455.
2. Select an object type from the drop-down list.
3. Select whether to retrieve all objects of the given type, or to retrieve objects having a designated string in the object name. A wildcard character (*) can be used when specifying the object name.
4. Click **OK** or **Apply** to start the search.

   The results will be displayed in a search results window.

For example, use the following steps to find the BeenThereBean enterprise bean (which is included in the sampleApp):

1. Select **Console** -> **Find...** from the main menu, as shown in Figure 13-11.
2. In the Find Objects window, select Object Type **Module** from the drop-down list.

3. Type in the enterprise bean name, including the wildcard character if needed. We entered `BeenThere*`, as shown in Figure 13-11.

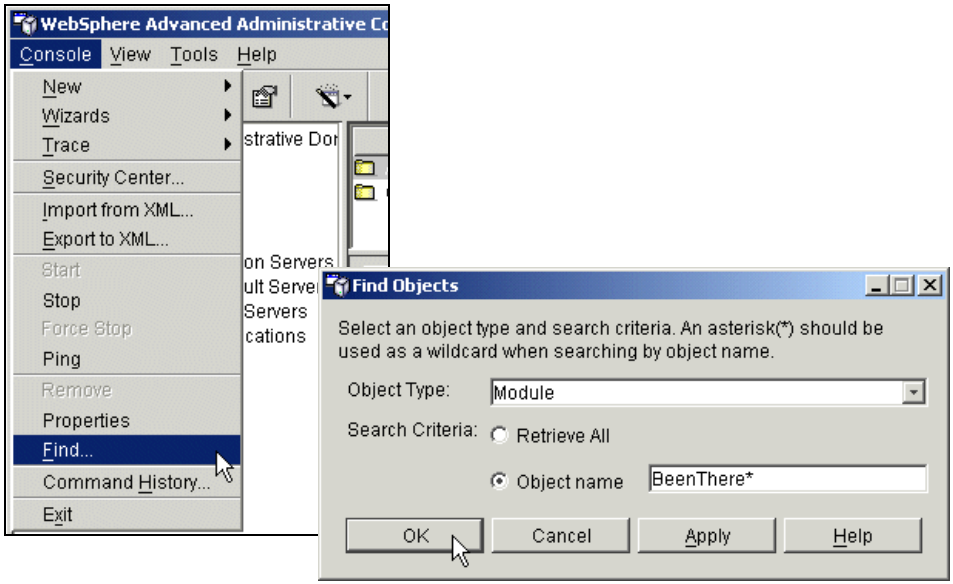


Figure 13-11 Finding an item

4. Click **OK** or **Apply**. The search results window pops up, as shown in Figure 13-12, indicating that the BeenThereBean module was found in the itsohost_sampleApp enterprise application, on the Default Server application server.

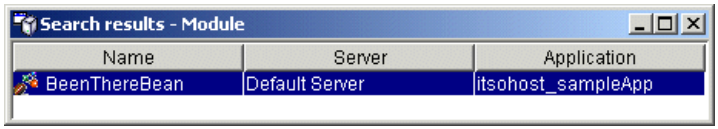


Figure 13-12 Module search results

13.1.9 Filtering messages

To filter the messages displayed in the messages view do the following:

1. Make sure the message view is enabled in **View** -> **Show Console Messages**.
2. Click the **Options** button in the messages view. You will get the Event Viewer Options window shown in Figure 13-13.

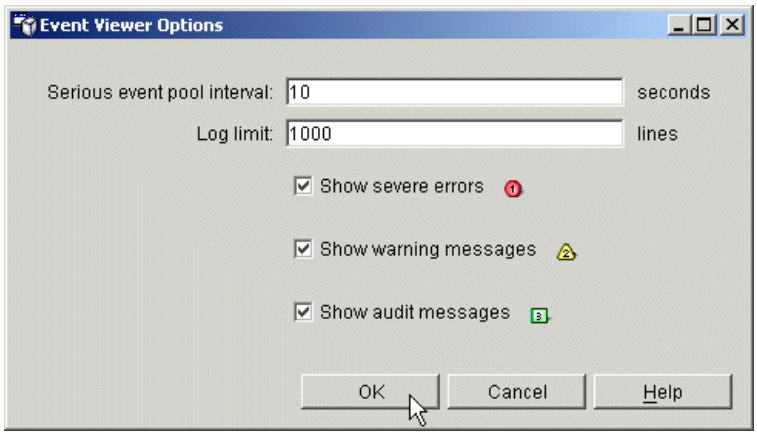


Figure 13-13 Filter console messages

3. Set the required properties and click **OK**.

13.1.10 Showing the command history

To see what commands have been used in the current session, do the following:

- Click **Console** -> **Command History...** to bring up the Command History window.

You can click the **View Error** button for details on unsuccessful commands.

13.1.11 Creating items using wizards

As shown in Figure 13-14, the following wizards are available to assist administrators with the creation of WebSphere items:

- Install Enterprise Application (see Chapter 19, “Deploying an application” on page 687)
- Create Application Server
- Create Server Group
- Create Data Source (see 16.1, “JDBC providers” on page 564)
- Create JMS Resources (see 16.5, “JMS providers” on page 595)
- Create J2C Connection Factory (see 16.4, “J2C resource adapters” on page 584)
- Create URL Provider (see 16.3, “URL providers” on page 578)

- Performance Tuner (see Chapter 22, “Monitoring and tuning your runtime environment” on page 839)

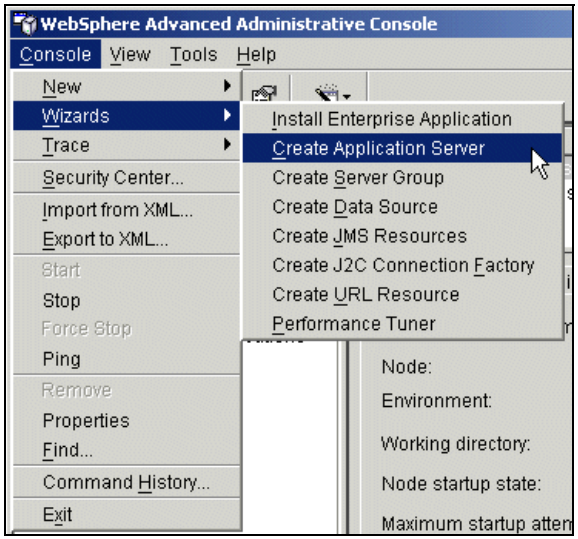


Figure 13-14 WebSphere Administrative Console wizards

These wizards lead the administrator through the process of creating a component, highlighting associated areas that may need to be configured. For example, if you use the Create Application Server wizard it will:

- Ask you to specify a name for the application server and select the node that it should run on.
- Ask you to specify other services that you want to enable for the application server, for example, the EJB Container Service and the Web Container Service.
- Display a summary of the settings you provided and ask you to confirm them.

These wizards are good place for administrators to familiarize themselves with the steps required to create selected components.

All of the tasks that the wizards can perform can also be performed by right-clicking the appropriate folder in the console tree view. This approach is more direct and probably quicker, but the administrator is expected to know what settings are required.

Once the administrator has some experience with WebSphere V4.0, they will probably adopt the latter approach and it is this approach that is documented in this book.

13.1.12 Starting tools

You can start the following WebSphere tools by selecting **Tools** from the console main menu:

- Application Assembly Tool
- Log Analyzer
- Resource Analyzer

The Application Assembly Tool (AAT) is for packaging enterprise applications. For further information, please refer to Chapter 18, “Packaging an application” on page 639.

The Log Analyzer is a new tool in IBM WebSphere Application Server V4.0. It provides a graphical user interface for analyzing log files using a symptom database. This database is provided by IBM and can be downloaded from the Web. For further information please refer to 24.10, “Log Analyzer” on page 998.

With Resource Analyzer you can monitor the performance of your runtime environment. It provides a GUI with chart and table views of performance data. For further information, please refer to 22.2, “Using WebSphere Resource Analyzer” on page 846.

13.1.13 Getting help

The WebSphere Administrative Console online help is HTML based. Context-sensitive help is separated into three subjects:

- Concept Help
- Task Help
- Field Help

To view context-sensitive help, first select the required item in the console tree view, then select **Help** from the main menu and choose the subject you want. Alternatively, you can browse the whole InfoCenter by selecting **Help** -> **Information Center** from the main menu.

The InfoCenter can be viewed online or downloaded from:

`http://www.ibm.com/software/webservers/appserv/infocenter.html`

13.2 Common administrative tasks

This section examines some of the tasks commonly performed by WebSphere Application Server V4.0 administrators.

13.2.1 Starting and stopping a node

Starting a node in the WebSphere environment means to start the nanny process which in turn starts the administrative server on a machine. On Windows platforms the nanny consists of adminservice.exe and on UNIX platforms it is simply a Java program running in a separate JVM.

Starting a node on Windows

To start the administrative server do one of the following:

- Click **Start** -> **Programs** -> **IBM WebSphere** -> **Application Server V4.0 AE** -> **Start Admin Server**.
- From the Windows Services panel choose **IBM WS AdminServer 4.0** and click **Start**.
- At the command prompt, type:

```
net start "IBM WS AdminServer 4.0"
```

You can also start the administrative server with the adminserver.bat script located in the <WAS_HOME>\bin directory. With this method, the administrative server is not run as a Windows service so you cannot stop it like a service.

The administrative server is completely up and running when the following line appears in <WAS_HOME>\logs\tracefile:

```
A WSVR0023I: Server __adminServer open for e-business
```

Stopping a node on Windows

To stop the administrative server do one of the following:

- From the WebSphere Administrative Console, right-click the node and select **Stop** from the pop-up menu.
- From the Windows Services panel locate the **IBM WS AdminServer 4.0** service,and select **Stop** from the pop-up menu.
- At the command prompt, type:

```
net stop "IBM WS AdminServer 4.0"
```

- Use the WSCP **Node** command:

```
wscp -c "Node stop /Node:node_name/"
```

Starting a node on UNIX

To start the administrative server, use the following commands in a UNIX shell:

```
cd <WAS_HOME>/bin
startupServer.sh &
```

The administrative server is completely up and running when the following line appears in the <WAS_HOME>/logs/tracefile:

```
A WSVR0023I: Server __adminServer open for e-business
```

Stopping a node on UNIX

To stop the administrative server do one of the following:

- From the WebSphere Administrative Console, right-click the node and select **Stop** from the pop-up menu.
- Use the WSCP **Node** command:

  ```
  wscp -c "Node stop /Node:node_name/"
  ```

- Find the pid of nanny and adminserver processes with the **ps -ef** command. Look for processes they are running Java. In addition the nanny and administrative server processes contain the following in their command lines:

nanny	com.ibm.ejs.sm.util.process.Nanny
adminserver	com.ibm.ejs.sm.server.AdminServer

  Once you have the pids, you can stop the processes with the **kill** command.

13.2.2 Starting and stopping an application server

In this section we look at how to start and stop application servers using the WebSphere Administrative Console, the WebSphere Control Program (WSCP), and XMLConfig.

Starting an application server

In order to start an existing application server the administrative server must be started beforehand. To start an application server do one of the following:

- From the WebSphere Administrative Console, right-click the required application server, for example Default Server, and select **Start** from the pop-up menu.
- From the WSCP prompt, type the following to start the Default Server:

  ```
  wscp> ApplicationServer start {/Node:node_name/ApplicationServer:Default
  Server/}
  ```

- See 23.4.9, “Starting and stopping resources” on page 934 for how to start an application server using XMLConfig.

Stopping an application server

To stop an application server do one of the following:

- From the WebSphere Administrative Console, right-click the required application server, for example Default Server, and select **Stop** from the pop-up menu.
- From the WSCP prompt, type the following to stop the Default Server:

  ```
  wscp> ApplicationServer stop {/Node:node_name/ApplicationServer:Default
  Server/}
  ```

- See 23.4.9, “Starting and stopping resources” on page 934 for how to stop an application server using XMLConfig.

13.2.3 Creating an application server

In this section we provide an example of creating a new application server, highlighting the basic properties that should be considered. Other application server properties are discussed in subsequent chapters.

You can create a new application server with the administrative console using the following steps:

1. We recommend that you create a specific working/logs directory for each application server, for example:

   ```
   mkdir D:\webbank\WebbankServer01\logs
   ```

2. Expand the WebSphere administrative domain tree.
3. Right-click the **Application Servers** folder under the required node and select **New...** from the pop-up menu, as seen in Figure 13-8 on page 452.
4. In the Create Application Server window, set the required general properties for the new application server, as shown in Figure 13-15.

   In this window you must specify a name for the new application server. We recommend that you also specify a unique working directory so you will know where to find any JVM core dumps, and so on.

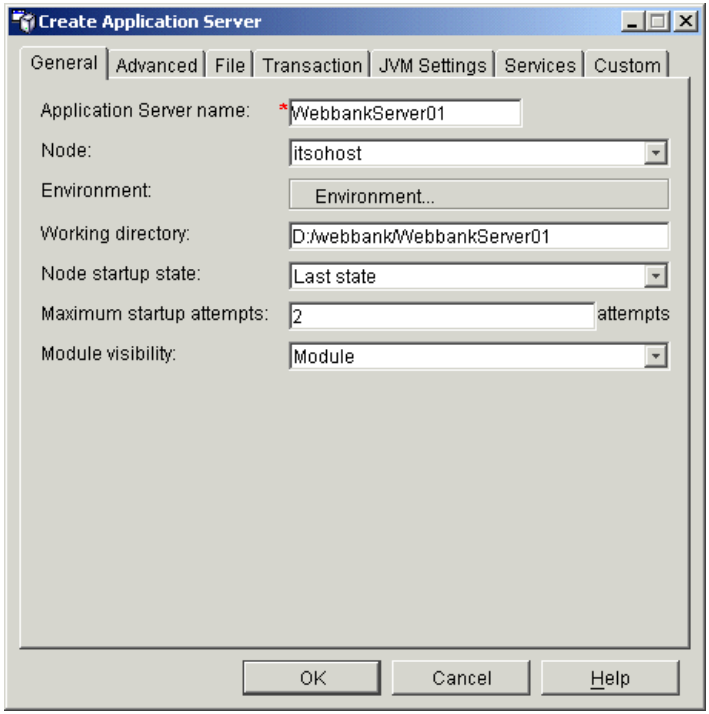


Figure 13-15 Create application server - general properties

5. Select the **File** tab to specify standard output and standard error log files for your application server, as shown in Figure 13-16.

   Always set standard output and standard error log files for each new application server, because these log files are an important source of information when problems arise. We recommend that you specify the log file paths relative to the unique working directory, as we did in Figure 13-16.

   Note: The WebSphere administrative server process must have the correct file system permissions to be able to generate log files.

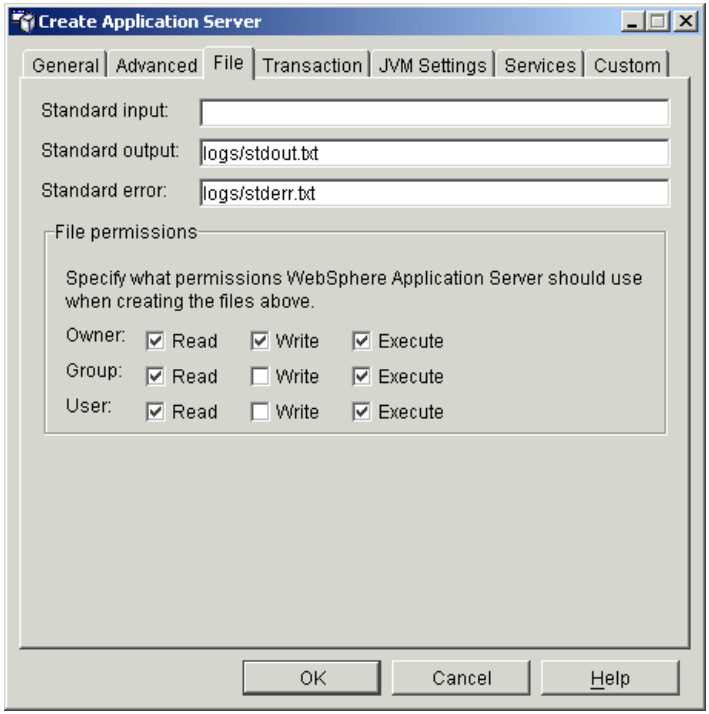


Figure 13-16 Create application server - file properties

WebSphere maintains a counter, starting at port 9080, for assigning a default Web container HTTP transport port. We recommend that you check that the assigned port is okay.

6. Select the **Services** tab, then the **Web Container Service**, then click the **Edit Properties...** button, as shown in Figure 13-17.

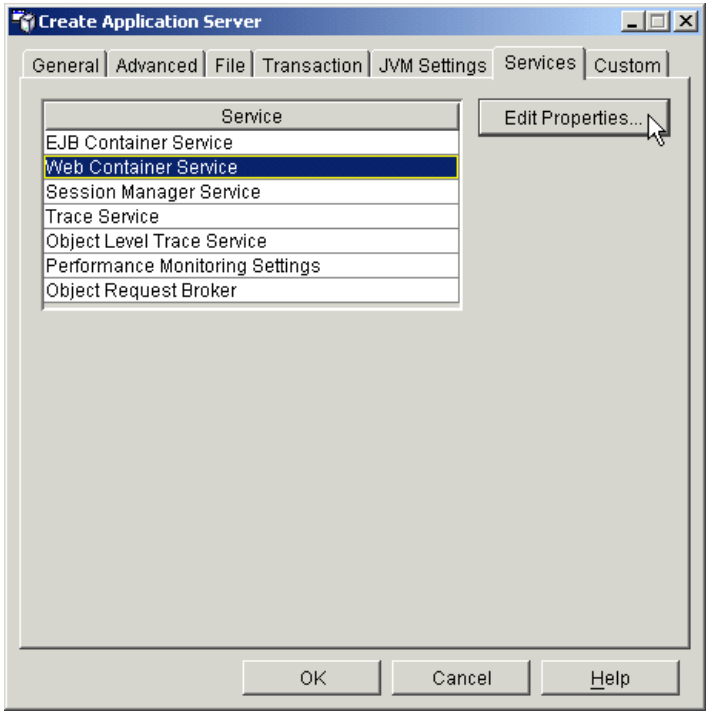


Figure 13-17 Create application server - services properties

7. In the Web Container Service window, click the **Transports** tab. In the Web Container Service transport properties, shown in Figure 13-18, check that the HTTP transports port is okay.

   If you need to change the port, select the transport and click the **Edit** button.

   Tip: If you want to access the Web container HTTP transport directly (not through a Web server plug-in) then check that the new port has an alias in the required virtual host.

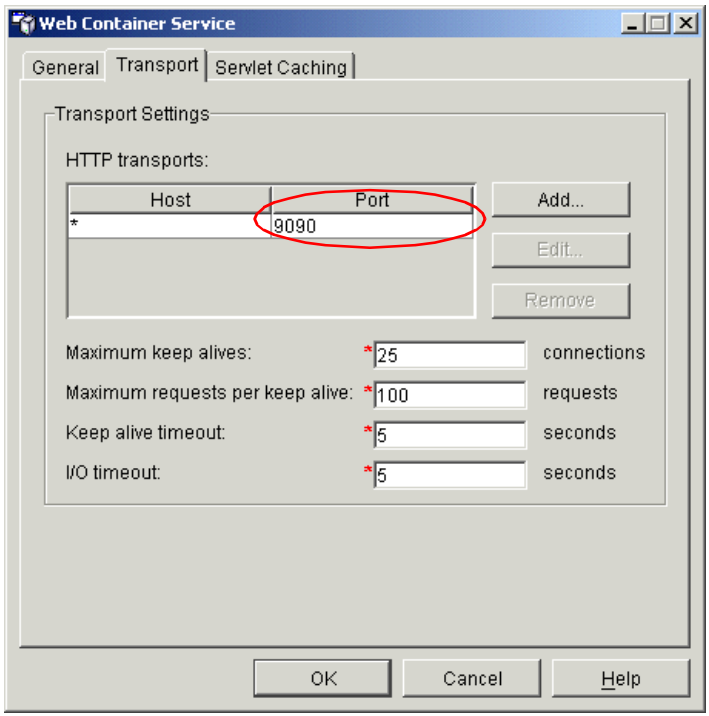


Figure 13-18 Web container service - transport properties

8. Click **OK** to close the Web container service window.
9. Click **OK** to create the application server on that node.

Your new application server should now appear in the console tree view, under the required node's Application Servers folder.

13.2.4 Working with enterprise applications

In this section we look at how to start, stop, and export enterprise applications using the WebSphere Administrative Console and the WebSphere Control Program (WSCP).

Starting an enterprise application

In order to start an existing enterprise application the administrative server must be started beforehand. Starting the enterprise application will also start any stopped application servers needed by the enterprise application. To start an enterprise application do one of the following:

- From the WebSphere Administrative Console, right-click the required enterprise application, for example itsohost_sampleApp, and select **Start** from the pop-up menu, as shown in Figure 13-19.
- From the WSCP prompt, type the following to start the itsohost_sampleApp:

  ```
  wscp> EnterpriseApp start /EnterpriseApp:itsohost_sampleApp/
  ```

  If you are not sure of the name of your enterprise application, use the `list` operation first:

  ```
  wscp> EnterpriseApp list
  ```

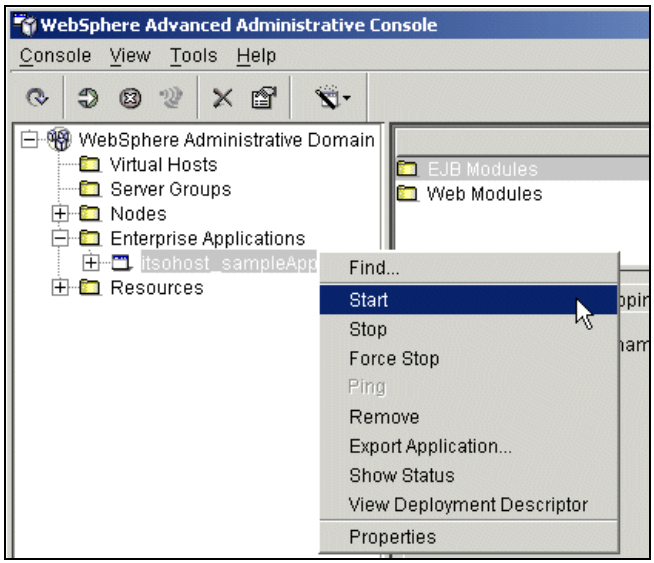


Figure 13-19 Starting an enterprise application from the console

Stopping an enterprise application

To stop an enterprise application, do one of the following:

- From the WebSphere Administrative Console, right-click the required enterprise application, for example itsohost_sampleApp, and select **Stop** from the pop-up menu.
- From the WSCP prompt, type the following to stop the itsohost_sampleApp:

  ```
  wscp> EnterpriseApp stop /EnterpriseApp:itsohost_sampleApp/
  ```

Showing status

To display the current state of Web modules and EJB modules in the enterprise application as well as the application server in which they are running, do the following:

- Right-click the enterprise application and select **Status** from the pop-up menu. This displays the module status window, as shown in Figure 13-20.

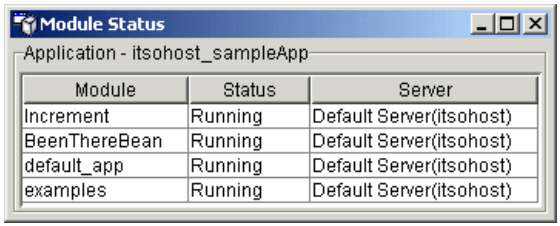


Figure 13-20 Status of the sampleApp Enterprise Application

Exporting an application

If you have modified the binding information of an enterprise application, you may want to export the changed bindings to a new EAR file. To export an enterprise application to an EAR file:

- Right-click the enterprise application and select **Export Application...** from the pop-up menu. This displays the Export Application window, as shown in Figure 13-21.

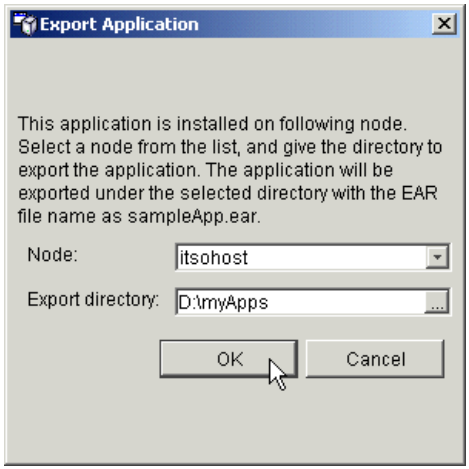


Figure 13-21 Export the enterprise application to an EAR file

13.2.5 Viewing installed applications

In WebSphere V4.0, the administrative console does not display the deployed servlets, JSPs, or EJBs directly on the console. To see this kind of information, the administrator can use the console to display XML deployment descriptors for the enterprise application, Web modules and EJB modules.

Viewing enterprise applications

The place to start is with the Enterprise Applications folder in the console tree view. This lists all of the enterprise applications installed in this WebSphere domain. Each enterprise application has a set of EJB modules and Web modules associated with it. These modules are made up of a number of files (including WAR files and EJB JAR files) which hold the servlets, JSPs and EJBs for the application.

To see the WAR files and JAR files associated with an enterprise application:

1. Right-click the enterprise application that you are interested in and select **View Deployment Descriptor** from the pop-up menu, as shown in Figure 13-22.

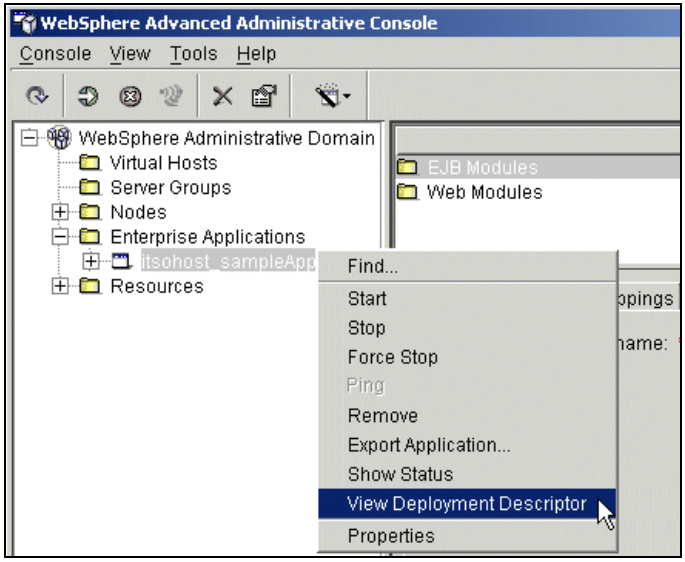


Figure 13-22 Viewing the enterprise application deployment descriptor from the console

In the View Deployment Descriptor window, shown in Figure 13-23, you can see the details of the application, such as:

- The name and description of the enterprise application.
- The Web modules, WAR files in this enterprise application, and their context roots, that is the URIs served by these modules.
- The EJB modules and their associated JAR files.
- The Security roles associated with the enterprise application.

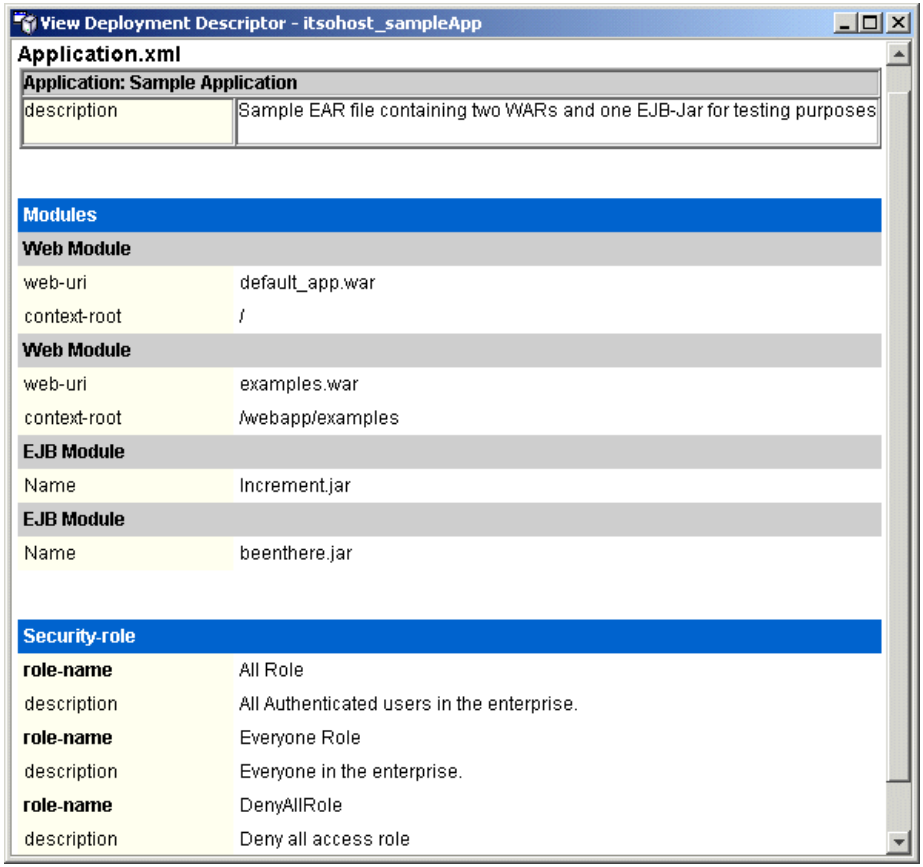


Figure 13-23 Enterprise application deployment descriptor

Viewing EJB modules

To see the EJBs that are part of an enterprise application:

1. In the console tree view, click the EJB Modules folder for the enterprise application. This will display all of the EJB modules in the enterprise application in details view.
2. To see the contents of an EJB module, right-click the module and select **View Deployment Descriptor** from the pop-up menu, as shown in Figure 13-24.

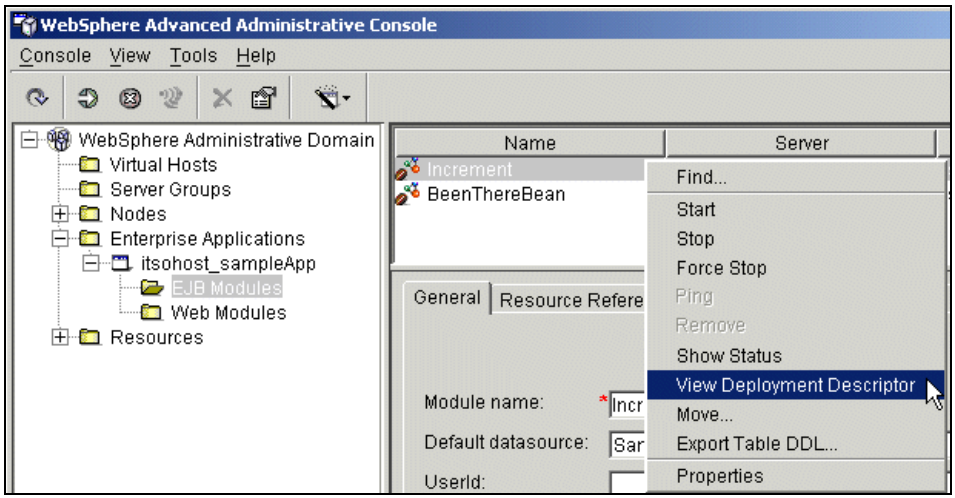


Figure 13-24 View contents of an EJB module

In the View Deployment Descriptor window, partly shown in Figure 13-25, you can see the details of the EJB module, such as:

- The name of the EJB JAR file containing the EJBs.
- The EJBs in the module. For each EJB you can see:
  - The type of bean.
  - The name bean and its interface classes.
  - Security role information (partly shown).
  - Method permissions (not shown).
  - Transaction information (not shown).

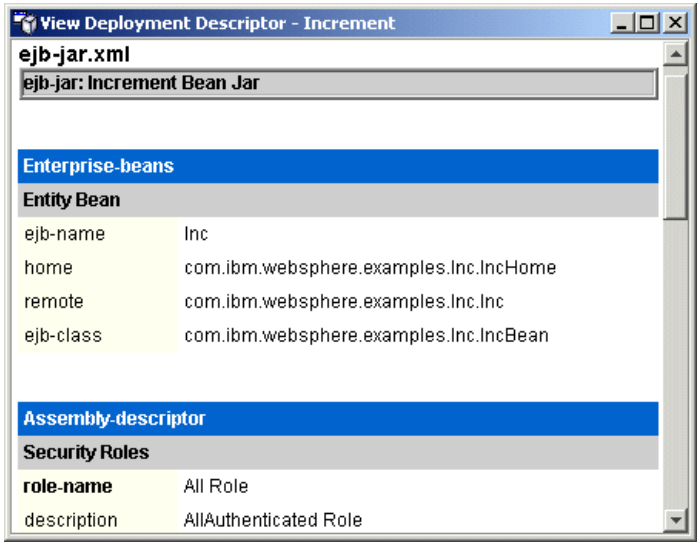


Figure 13-25 Part of the deployment descriptor for the Increment EJB module

Viewing Web modules

To see the servlets and JSPs that are part of an enterprise application:

1. In the console tree view, click the Web Modules folder for the enterprise application. This will display all of the Web modules in the enterprise application in details view.
2. To see the contents of a Web module, right-click the module and select **View Deployment Descriptor** from the pop-up menu, as shown in Figure 13-26.

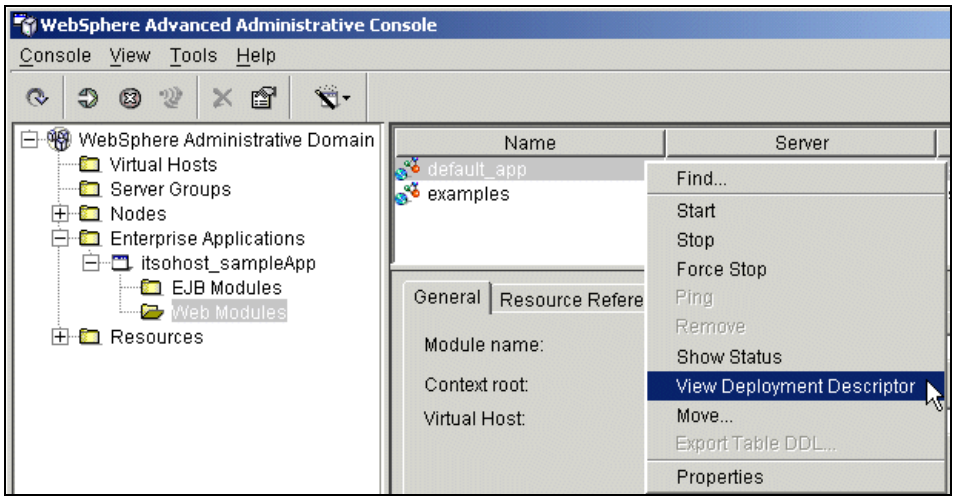


Figure 13-26 View the contents of a Web module

In the View Deployment Descriptor window, partly shown in Figure 13-27, you can see the details of the Web module, such as:

- The name and description of the Web module.
- The JSPs and servlets in the module and a description of each (partly shown).
- URLs for each servlet and JSP starting from the context-root (not shown).
- Security information about the servlets/JSPs (not shown).
- A list of EJBs referenced by the servlets (not shown).

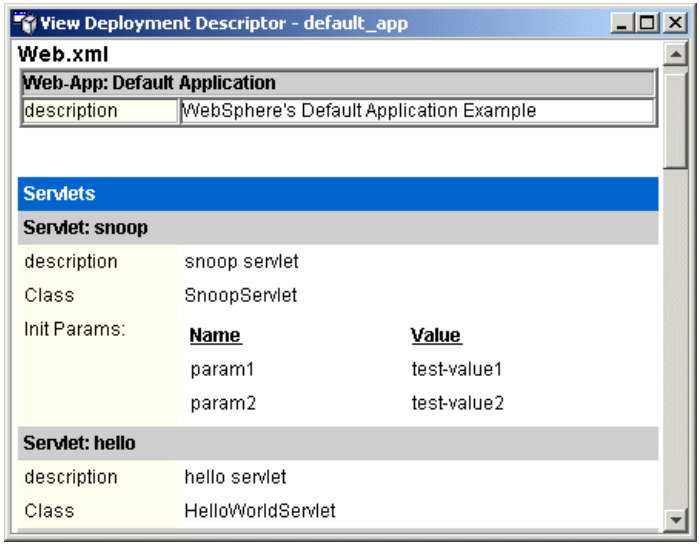


Figure 13-27 Part of the deployment descriptor for default_app Web Module

13.2.6 Finding a URL for a servlet or JSP

The URL for a servlet or JSP is the path used to access it from a browser. In WebSphere V3.5.x this information was found in a property pane in the administrative console. In WebSphere V4.0, it is partly defined in the deployment descriptor provided in the EAR file.

For example, to look up the URL for the snoop servlet:

1. From the WebSphere Administrative Console locate the required application in the Enterprise Applications folder. Clicking **Console** -> **Find...** may help here.
2. Double-click the application, in our case **itsohost_sampleApp**, to view its EJB Modules and Web Modules folders.
3. On the properties view, right-click the **default_app** Web module and select **Properties** from the pop-up menu.

   This displays the Web module properties window, as shown in Figure 13-28. Note that the virtual host is “default_host” and the context root is “/”. We will use these properties later.

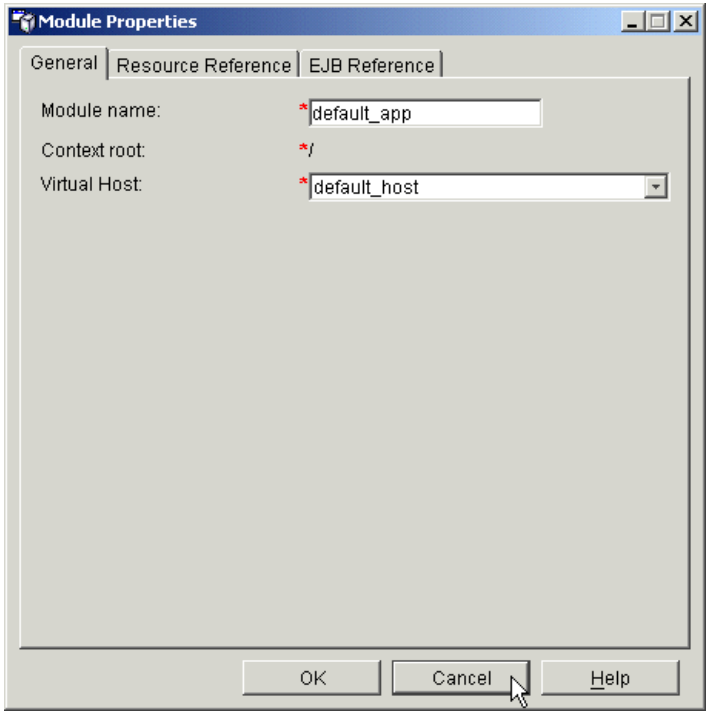


Figure 13-28 Web module properties

4. Click **Cancel** to close the Web Module Properties window.
5. Back in the console properties view, right-click the **default_app** Web module and select **View Deployment Descriptor** from the pop-up menu, as seen in Figure 13-26 on page 472.

   This displays the View Deployment Descriptor window. As shown in Figure 13-29, the URL pattern of the snoop servlet is “/servlet/snoop/*”.

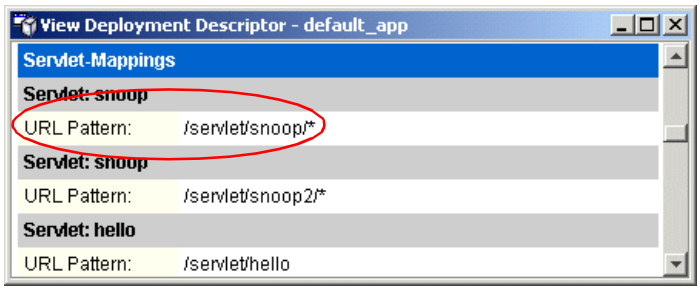


Figure 13-29 Deployment descriptor of default_app Web module

6. Combine the virtual host, context root, and URL pattern, as follows:

   `http://<virtual host><context root><URL pattern>`

   With the standard default_host, virtual host, you should be able to access the snoop servlet with a URL of the form:

   `http://*:80/servlet/snoop/*` or `http://*:9080/servlet/snoop/*`

   If the host names localhost and itsohost.itso.ibm.com resolve to the correct Web server and/or application server, then some examples of the full URL used to access the snoop servlet are:

   ```
   http://localhost/servlet/snoop
   http://localhost:9080/servlet/snoop
   http://itsohost.itso.ibm.com/servlet/snoop
   http://itsohost.itso.ibm.com/servlet/snoop/
   http://itsohost.itso.ibm.com/servlet/snoop/hello
   http://itsohost.itso.ibm.com/servlet/snoop?a=b
   http://itsohost.itso.ibm.com:9080/servlet/snoop
   ```

   and so on.

13.2.7 Regenerating Web server plug-in configuration

Changing certain WebSphere configuration properties makes it necessary to regenerate the Web server plug-in configuration. To accomplish this do one of the following:

- From the WebSphere Administrative Console right-click the node and select **Regen Webserver Plugin** from the pop-up menu, as shown in Figure 13-30.

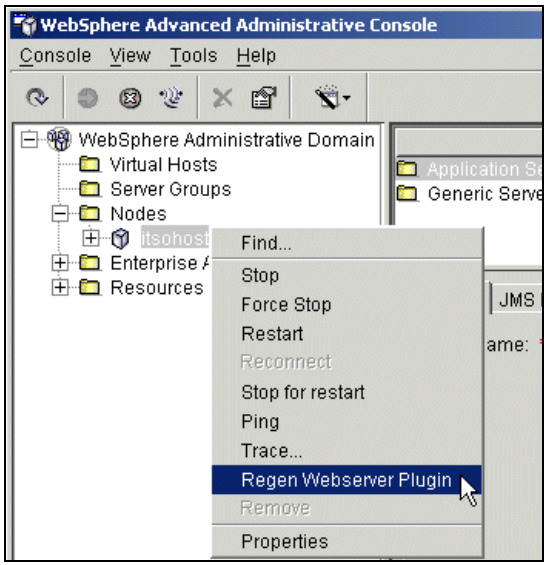


Figure 13-30 Regenerating the Web server plug-in configuration from the console

- Use the script GenPluginCfg.bat (Windows) or GenPluginCfg.sh (UNIX):

  ```
  <WAS_HOME>/bin/GenPluginCfg.bat -adminNodeName <adminnode_name> -nodeName
  <node_name>
  ```

  Where:

`adminnode_name`	Specifies the administrative console to which you want to connect.
`node_name`	Specifies the node for which the plug-in configuration should be regenerated.

- Use WSCP as follows:

  ```
  wscp -c "Node regenPluginCfg /Node:nodename/"
  ```

- Use XMLConfig as follows:

  ```
  XMLConfig -adminNodeName nodename -export trash.xml -generatePluginCfg true
  ```

  The import or export option must be given. You could use export and drop the exported file if it is not needed.

In a development environment, you may want to enable automatic regeneration of the Web server plug-in configuration. See 14.1.2, “Generating the plug-in’s XML configuration file” on page 487 for details.

13.2.8 Saving the WebSphere configuration

To save the configuration of your WebSphere administrative domain, do the following:

1. Select **Console** -> **Export to XML...** from the main menu. The Save window pops up, as shown in Figure 13-31.
2. In the Save window, choose the location and the name for the XML file.

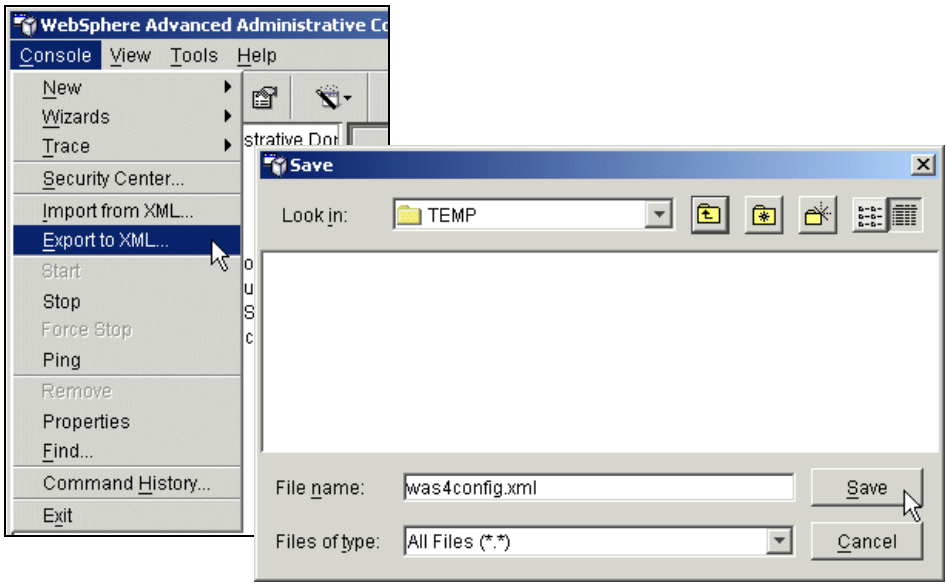


Figure 13-31 Export WebSphere configuration to XML

3. Click **Save**.

The message view should show you that the export was successful, for example:

```
Command "itsohost.export" completed successfully.
```

Only a full XML export can be performed from the WebSphere Administrative Console. For partial export using XMLConfig, refer to “Partial export” on page 927.

Refer also to Appendix A, “Back up and restore your WebSphere environment” on page 1057.

13.2.9 Restoring the WebSphere Configuration

To restore the configuration of your WebSphere administrative domain, do the following:

1. Select **Console** -> **Import from XML...** from the main menu. The Import window pops up, as shown in Figure 13-32.
2. In the Import window, choose the appropriate XML file.

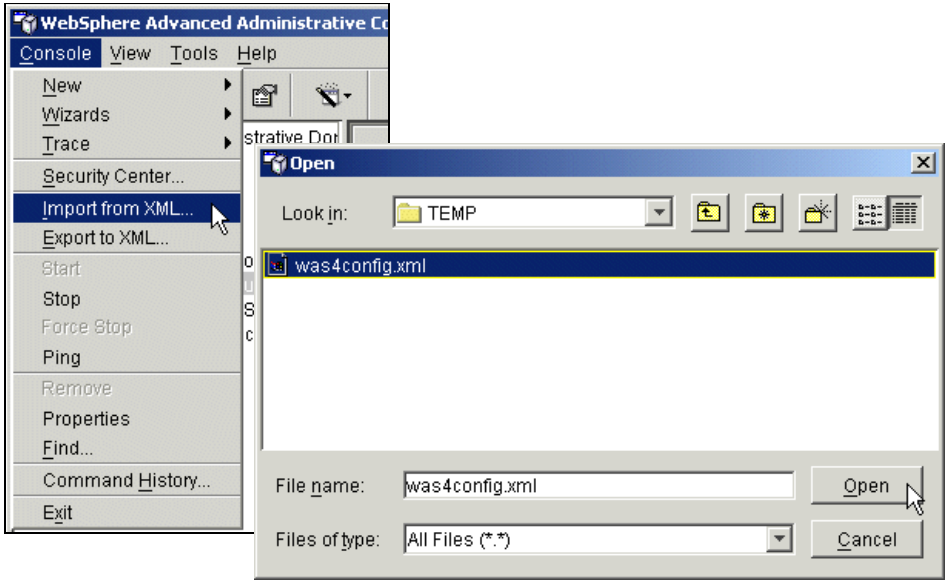


Figure 13-32 Import WebSphere configuration from XML file

3. Click **Open**.

The message view should indicate that the import was successful.

Refer also to Appendix A, “Back up and restore your WebSphere environment” on page 1057.

13.2.10 Checking versions

The WebSphere Application Server version and the versions of related software are important. All components need to be at the correct versions for proper inter-operation. In this section we describe how to figure out what the versions and build levels are of the various components of your environment. We are not specifying the supported versions here. We are showing you how to find out the versions of your installed components.

WebSphere Application Server version

To figure out which version of WebSphere Application Server is currently installed, investigate one of the following places:

- In the WebSphere Administrative Console, select **Help** -> **About** from the main menu.
- Look in the product version file:

  <WAS_HOME>/properties/com/ibm/websphere/product.xml
- Look in the administrative server trace file:

  <WAS_HOME>/logs/tracefile

  In the tracefile look for version information, as shown in Example 13-1.

Example 13-1 WebSphere and JDK versions in tracefile

```
************ Start Display Current Environment ************
WebSphere AE 4.0.1 a0131.07 running with process name itsohost/__adminServer
and process id 964
Host Operating System is Windows 2000, version 5.0
Java version = J2RE 1.3.0 IBM build cn130-20010609 (JIT enabled: jitc), Java
Compiler = jitc
...
```

JDK version

To figure out which version of the JDK you have installed in your environment do one of the following:

1. Look in the administrative server trace file, as shown in Example 13-1:

   `<WAS_HOME>/logs/tracefile`
2. Start the Java binary with the `-fullversion` option:

   `<WAS_HOME>/java/bin/java -fullversion`

   The output of the Java `-fullversion` option should be similar to:

   ```
   java full version "J2RE 1.3.0 IBM build cn130-20010609"
   ```

DB2 version

To find out the DB2 version installed on your system do the following:

- Open a DB2 command window (Windows) or switch to your DB2 instance owner (UNIX) and issue the following command:

  `db2level`

  The output should look be similar to:

  ```
  DB21085I  Instance "DB2" uses DB2 code release "SQL07021" with level
  identifier
  "03020105" and informational tokens "DB2 v7.1.0.43", "n010504" and
  "WR21254a".
  ```

IBM HTTP Server version

To check the version of your IBM HTTP Server or Apache Web server do the following:

On Windows platforms, running **apache.exe -v** will show you the version, for example:

```
"D:\IBM HTTP Server\apache.exe" -v
Server version: IBM_HTTP_SERVER/1.3.19  Apache/1.3.20 (Win32)
Server built:   Jul 11 2001 17:22:19
```

On UNIX platforms, running **httpd -v** will show you the version, for example:

```
/usr/HTTPServer/bin/httpd -v
Server version: IBM_HTTP_SERVER/1.3.19  Apache/1.3.20 (Unix)
Server built:   Jun 26 2001 17:14:53
```

14

Configuring the Web server interface

In this chapter we discuss how you configure the interface between WebSphere Application Server and your Web server. Our discussion includes the following topics:

- The Web server plug-in
- Virtual hosts
- Configuring transports for WebSphere

14.1 Web server plug-in

The WebSphere plug-in is a module that runs as part of the Web server process. It acts as the gatekeeper between the Web server and WebSphere. It decides which HTTP requests should be passed to WebSphere and which ones are for the Web server alone to handle.

Once the plug-in has decided that an HTTP request should be passed to WebSphere, it packages up the request and sends it to an application server or a clone in a server group. It uses a set of rules in an XML file to decide which application server or clone should handle the request. See Example 14-1 on page 484 for a sample plugin-cfg.xml file. This file can be generated from the WebSphere administration console (see 14.1.2, “Generating the plug-in’s XML configuration file” on page 487) from the command line using GenPluginCfg.bat/sh. For a full list of the possible parameters used in this file see Appendix C, “The plugin-cfg.xml file definitions” on page 1071.

Note: By default, WebSphere does *not* automatically regenerate the plug-in’s XML configuration file after changes have been made to the configuration or after the application server is restarted. See 14.1.2, “Generating the plug-in’s XML configuration file” on page 487.

The plug-in in WebSphere V4.0 uses the HTTP 1.1 transport to communicate between the Web server and the WebSphere applications. Prior to WebSphere V4.0, a proprietary protocol known as OSE (Open Servlet Engine) was used. As of WebSphere V4.0, the OSE plug-in is not supported. The servlet redirection capabilities of WebSphere 3.x are also not supported in WebSphere V4.0. Web servers using the OSE plug-in or servlet redirection will have to migrate to the HTTP plug-in before they can be used with WebSphere V4.0.

The new HTTP plug-in has the following advantages over the old OSE plug-in:

- The HTTP version of the plug-in is much faster than the OSE plug-in. In tests it has been shown to be as much as 30 to 60 percent faster.
- HTTP is firewall friendly, that is most firewalls will allow HTTP to pass through. Also, due to the use of HTTP 1.1, connections between the plug-in and the application server can be reused by multiple requests, which reduces the load on the firewall.
- HTTPS enables us to encrypt messages sent between the Web server and WebSphere using a standard protocol.
- It is much easier to configure the HTTP plug-in. A single XML file controls the HTTP plug-in instead of the four, somewhat cryptic files used by the OSE

plug-in. The XML file is much more readable and benefits from having a defined XML document structure.

- It supports the new session affinity mechanism for cloned applications. This ensures that HTTP requests in the same HTTP session are always routed to the same Web application, in the same JVM. See Chapter 15, “Configuring session management” on page 513 for more information on session affinity.
- Using SSL server-side authentication, the client can be sure that it is communicating with a valid application server. This avoids the “man in the middle” problem where a third party pretends to be the application server and intercepts the client’s requests.
- Using client-authenticated SSL, the application server can be sure that only valid clients are connected to it. This ensures that no one can pretend to be your Web site and use a direct connection to attack your Web application.
- Using HTTP as the transport allows WebSphere to simplify the development environment. In WebSphere V4.0, HTTP requests can be sent directly to the Web application server, so in a development environment you don’t need a Web server. Developers can send their requests directly to the host name and port number of their application server. The default application server port number is 9080.

Note: In production, it is recommended that a separate Web server be used instead of sending the requests directly to the WebSphere Application Server because:

- The Web server plug-in allows requests to be workload managed across multiple clones.
- A Web server is probably better at serving static information such as HTML and GIF files.
- Web server security can be used to protect information against unauthorized access.
- This allows the physical configuration to change without changing widely held URLs. If all request are sent to a Web server and redistributed to WebSphere via the plug-in, the administrator can change the number and name of the machines that WebSphere is running on without affecting the URL the users are accessing.
- If you are using the IBM HTTP Server or Apache, then there isn’t any extra cost involved to get the benefits above.

14.1.1 How an HTTP request is processed by the plug-in

When the Web server receives an HTTP request, the plug-in needs to determine whether the request is meant for a Web application that is running in WebSphere. If the request isn't meant to be processed by WebSphere, the plug-in passes a return code to the Web server to say that the Web server should attempt to handle the request instead. If the request is meant for WebSphere, the plug-in needs to decide which node the application is running on and, in the case of a cloned application, which clone should handle the request.

The rules that the plug-in uses are in the file called plugin-cfg.xml. Example 14-1 is a simple sample.

Example 14-1 HTTP plugin-cfg.xml configuration file

```
<?xml version="1.0" ?>
<Config>
   <Log LogLevel="Error" Name="D:\WebSphere\AppServer\logs\native.log" />
   <VirtualHostGroup Name="default_host">
      <VirtualHost Name="*:80" />
      <VirtualHost Name="*:9080" />
   </VirtualHostGroup>
   <ServerGroup Name="Default Server">
      <Server CloneID="sus6tcqk" Name="Default Server">
         <Transport Hostname="itsohost" Port="9080" Protocol="http" />
      </Server>
   </ServerGroup>
   <UriGroup Name="itsohost/sampleApp/default_app_URIs">
      <Uri Name="/servlet/snoop/*" />
      <Uri Name="/servlet/snoop" />
      <Uri Name="/servlet/snoop2/*" />
      <Uri Name="/servlet/snoop2" />
      <Uri Name="/servlet/hello" />
      <Uri Name="/ErrorReporter" />
      <Uri Name="*.jsp" />
      <Uri Name="*.jsv" />
      <Uri Name="*.jsw" />
      <Uri Name="/j_security_check" />
      <Uri Name="/servlet/*" />
   </UriGroup>
   <UriGroup Name="itsohost/sampleApp/examples_URIs">
      <Uri Name="/webapp/examples" />
   </UriGroup>
   <Route ServerGroup="Default Server"
      UriGroup="itsohost/sampleApp/default_app_URIs"
      VirtualHostGroup="default_host" />
   <Route ServerGroup="Default Server"
      UriGroup="itsohost/sampleApp/examples_URIs"
      VirtualHostGroup="default_host" />
```

```
</Config>
```

When an HTTP request is made to the Web server for a WebSphere application URL, the plug-in looks at the routes defined in the XML configuration file. If a VirtualHostGroup is defined for the route, it will attempt to match the incoming host name and port with the defined VirtualHosts. If no VirtualHostGroup is defined, the plug-in assumes that the route should match all virtual hosts. If a UriGroup is specified in the route, then the plug-in attempts to match the defined URIs with the incoming URI. If no UriGroup is defined, the plug-in assumes the route should match all URI groups. If a match is found for both the URI and the VirtualHostGroups defined in the route, the plug-in will then send the request to the appropriate ServerGroup using either workload management or session affinity, depending on which is applicable for the configuration.

Each route contains a ServerGroup tag. Once a route has been found, the plug-in examines it to get the ServerGroup. The next step is to determine which server in the ServerGroup will be sent the request. The ServerGroup may contain one or many servers depending on whether the Web application has been cloned or not. As part of determining which server should handle the request, the plug-in:

- Checks for a session affinity in the request. If the request has a session affinity cookie, the session has been allocated to a particular server by a previous request. In order to conform with the Servlet 2.2 specification, if the session has previously been allocated to a server, then this request must be passed to that server. The plug-in will check the incoming cookie header or URL for jsessionid. If the jsessionid is found then the plug-in will look for a CloneID. If there is a CloneID and it matches the CloneID of a server in the ServerGroup, the request will be sent to that server. Otherwise, load balancing will be used to allocate the request and session to the server.
- If no session affinity has been established load balancing is used to allocate the request to a Server. The default load balancing policy is round robin. The round robin implementation has a random starting point. This means that the first server will be picked randomly and then round robin will be used for each new session from that point forward. This is so that in multiple process-based Web servers, all of the processes don't start up by sending the first request to the same application server. The other possible policy is Random. This randomly allocates the sessions to the available servers.

For more information on session affinity, see Chapter 15, “Configuring session management” on page 513.

Tip: If you are not using session affinity then it is best to remove the CloneIds from the server tags, as they require extra processing by the plug-in when they are set. If CloneIds are not in the XML file plug-in it is assumed that session affinity is not enabled and it load balances the requests across the server group.

Once the identity of the server is known, the plug-in must work out what transport to use. Each server may define more than one transport, but if more than one transport of the same type is defined, for example more than one HTTP transport, then the first one in the XML file is used. Table 14-1 shows how HTTP and HTTPS requests are mapped by the plug-in to the available transports for an application server.

Table 14-1 Web server plug-in to WebSphere Application Server transport protocol

Is the incoming request HTTP or HTTPS?	Only HTTP transport available for the application server	Only HTTPS transport available for the application server	Both HTTP and HTTPS available for the application server
HTTP	HTTP	HTTPS	HTTP
HTTPS	HTTP	HTTPS	HTTPS

Once the transport has been determined, the plug-in knows the host, port and transport to send the request to by examining the attributes of the Transport tag in the XML file.

The I/O between the plug-in and the WebSphere Application Server is done using a high-speed HTTP client via HTTP Version 1.1. It maintains the connections between the plug-in and the WebSphere Application Server across multiple requests, which is much more efficient and firewall friendly than having to reconnect for each time.

As an example of how an HTTP request is processed by the plug-in, let's look at a request for URL `http://localhost/servlet/snoop` for the XML configuration file from Example 14-1 on page 484.

This URL will match the route with `VirtualHostGroup="default_host"` and `UriGroup="itsohost/sampleApp/default_app_URIs"`. The UriGroup contains matching `Uri Name="/servlet/snoop"` and the VirtualHostGroup contains matching `VirtualHost Name="*:80"` ("*:80" matches any host and the default HTTP port). The `ServerGroup="Default Server"` is defined for this route, so the plug-in will send the request to the application server at `Hostname="itsohost"` and `Port="9080"` using `Protocol="http"`.

Note: The plug-in log file is a useful tool for problem determination. Its name and location are specified in the plug-in's XML configuration file. If the log file doesn't exist, the plug-in will create it. If the file already exists, it will be opened in append mode and the previous plug-in log messages will remain. The amount of detail log to the log file is control by the LogLevel, which is set in the XML configuration file. There are three possible LogLevels: Trace, Warn, or Error. Trace allows you to see the steps in the request process in detail. Warn and Error means that only information about abnormal request processing will be logged. Be *very* careful when using Trace, since it can quickly fill up the disk and can be a serious performance hit.

14.1.2 Generating the plug-in's XML configuration file

Important: By default, WebSphere V4.0 doesn't automatically generate new XML configuration for the plug-in. The administrator must use the administrative console or GenPluginCfg.bat/sh from the command line to generate it. WebSphere can be configured to generated a new plug-in XML file every time an application server is restarted but care must be taken, since this option may overwrite manual edits of the file.

When the plug-in XML file is generated it contains information for all of the virtual hosts, server groups, nodes, application servers and Web modules in the WebSphere domain. It doesn't matter which node you use to generate the file because the output is the same; it contains information for the whole domain.

If changes to your WebSphere configuration are infrequent, as in a production environment, you should generate the plug-in XML file manually. If you need to manually edit the file, you should also generate the file manually to avoid having your changes accidentally overwritten. Some of the reasons to manually edit plug-in XML file are:

- If your Web server(s) are using a non-default directory structure, for example they are on a different operating system to WebSphere, you may have to edit this file before you copy it to the Web server machines.
- If you want to restrict knowledge of certain sensitive application(s) to particular Web servers and machines, where all of the Web servers may not need to know about all of your applications.
- If you want to specify particular key or trust files that the plug-in should use when connecting to the application server via SSL, that is, you don't want to use the default key or trust files.

On the other hand, for a development environment, you may find it more convenient to have WebSphere generate the file automatically.

Regenerate the plug-in's XML configuration file

To manually generate a new XML configuration file for the Web server plug-in:

1. Right-click the local node. Remote nodes can be used and will generate the same output, but the local node is faster.
2. Click **Regen Webserver Plugin**, as shown in Figure 14-1.

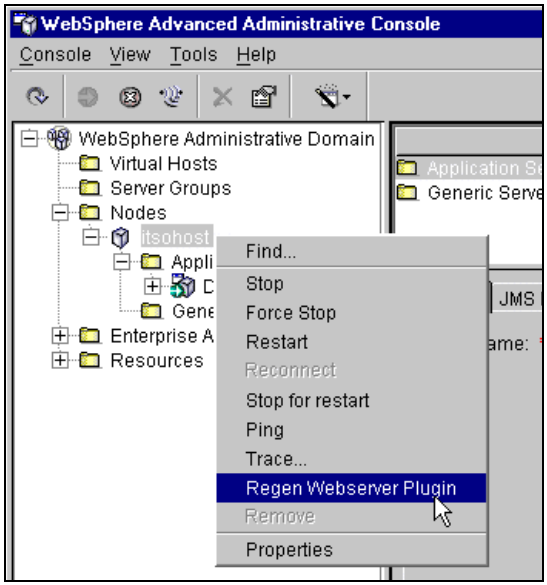


Figure 14-1 Generating a new XML configuration file for the plug-in

3. Wait for the command to complete by monitoring the Event Message window on the bottom of the console.

How to generate the plug-in configuration file automatically

To enable automatic generation of the plug-in's XML file:

1. Click the application server that should generate the file, as seen in Figure 14-2 on page 489.

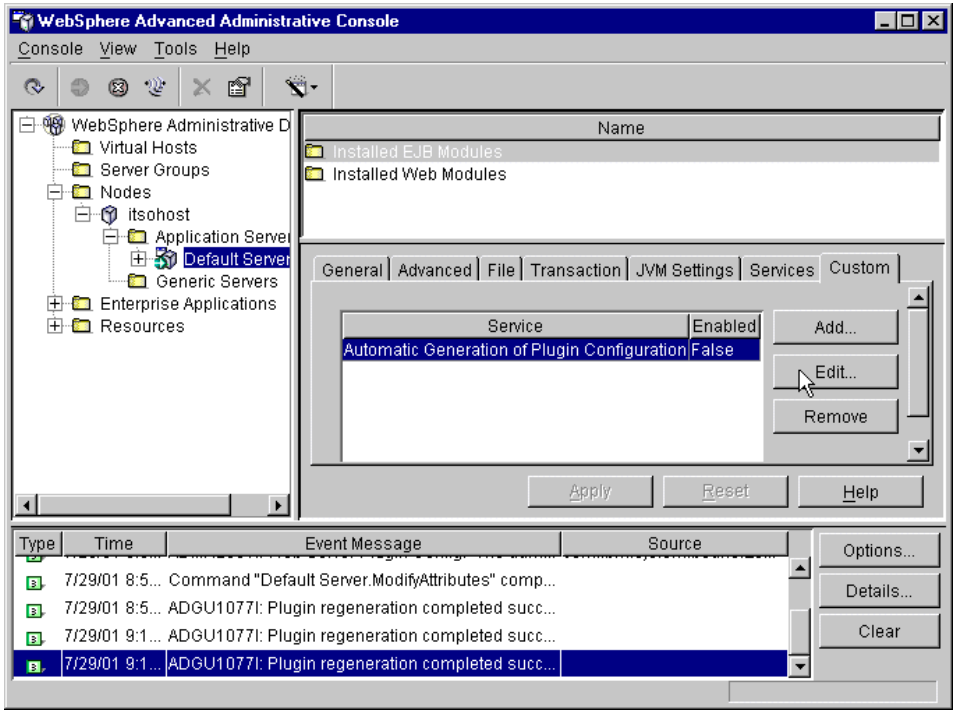


Figure 14-2 Edit automatic generation of plug-in configuration custom service

2. Click the **Custom** tab.
3. Click the **Automatic Generation Plugin Configuration** service.

   If this service does not appear, you will need to add it first:

   a. Click **Add...**.

   b. In the Add Custom Service window enter the following details:

      - Name

        Automatic Generation of Plugin Configuration

      - Description

        If enabled, the plugin configuration files will be regenerated when the application server is started.

      - Classname

        com.ibm.websphere.plugincfg.initializers.AEPluginCfgService

   c. Click **OK**.
4. Click the **Edit** button.

5. This will display the window in Figure 14-3. Click **Enabled**.

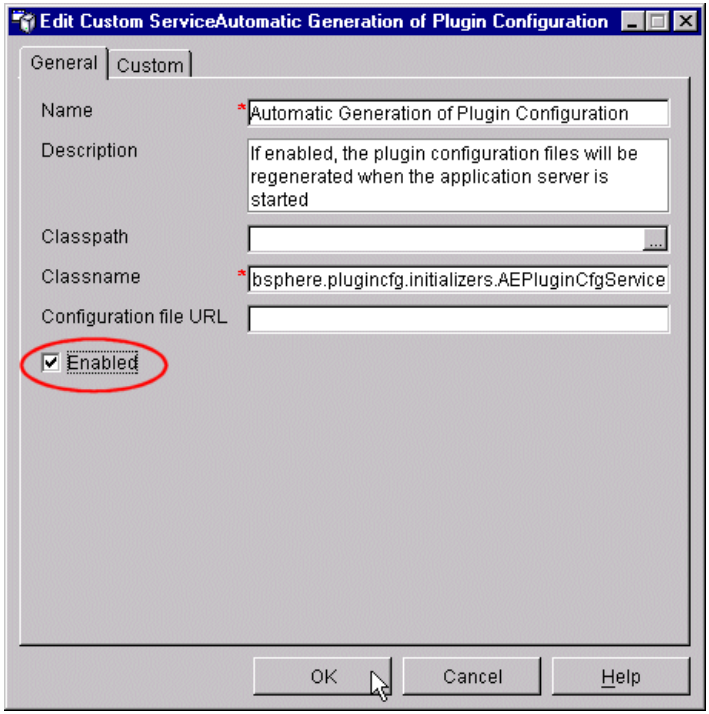


Figure 14-3 Enable the automatic generation of the plug-in's XML configuration file

6. Click **OK**.
7. Click **Apply**.

From now on the plug-in's XML file will be generated every time the selected application server is restarted.

14.1.3 Installing the plug-in XML configuration file

Once the new plug-in XML file has been generated, it has to be installed in a common directory so the plug-in can read it. By default, the administrator should install the plug-in XML file into the <WAS_HOME>/config/ directory. For most Web servers, this install directory can be changed. Please refer to Part 3, "Installing WebSphere" on page 125 for more information on installing the plug-in. For example, see 9.7, "Install the WebSphere plug-in on the remote Web server" on page 236 if using IBM HTTP Server on Windows 2000.

The directory in which the XML file should be installed may be on the same machine as WebSphere or on a remote machine. If the Web server is co-located with WebSphere and the default directory is used, WebSphere automatically will put the new plug-in XML file into the correct directory. If the Web server is on a remote machine, then the file will have to be manually copied to the remote machine and placed in the correct directory.

The Web server doesn't have to be restarted after a new plug-in XML file is installed. By default, the plug-in polls for a new file every 60 seconds. To change the poll rate, open the plugin-cfg.xml file and insert the RefreshInterval attribute into the Config tag, for example, `<Config RefreshInterval=10>` to reload the file every 10 seconds.

In a development environment, where changes are usually made quite often to the plug-in configuration file, a lower setting than the default is recommended. In production, a higher value than the default is a good idea, since updates to the configuration will not occur as often.

If the plug-in reload fails for some reason, a message is written to the plug-in log file and the previous configuration is used until the plug-in configuration reload is successful. If you do not see the changes you have made to your plug-in configuration take effect, then check the plug-in log file for clues on what might be going wrong.

14.2 Virtual hosts

A virtual host is a filter used by the WebSphere plug-in. When the plug-in receives an HTTP request it compares the host name and port number of the incoming request against a list of host aliases in the virtual host. A host alias contains a host name and a port number. When no port number is specified in the alias, port 80 is assumed. If a match is found, the plug-in will then check the URI as described in, 14.1.1, "How an HTTP request is processed by the plug-in" on page 484, to decide whether the request should be passed to WebSphere.

A virtual host doesn't represent a physical machine; it is only a filter, which is why it cannot be started or stopped.

Note: The host aliases don't have to be the same as the host name and port number of the WebSphere Application Server(s). They are the host name(s) and port number(s) that the plug-in is expecting to receive from the browser. The plug-in will send the request to the application server using the host name and port number in the transport settings for that server. If the Web server is running on a separate machine to WebSphere then the host aliases are for Web server machines.

Mapping HTTP requests to host aliases is case sensitive, and the match must be alphabetically exact. Also, different port numbers are treated as different aliases.

For example, the request:

```
http://www.myhost.com/myservlet
```

Does not map to:

```
http://myhost/myservlet
```

or to:

```
http://www.myhost.com/MyServlet
```

or to:

```
http://www.myhost.com:9876/myservlet
```

If the plug-in receives a request that does not match one of the virtual hosts, or if it does not match the URI of the incoming request against the URIs for the Web module associated with the virtual host, the user will receive an HTTP error in the browser used to issue the request.

Through the use of virtual hosts, the administrator can change the physical topology of an enterprise application without affecting the users of the Web site. The administrator can add, move or remove the machines the enterprise application is running on and, as long as the Web server hasn't changed, they can still use the same virtual host alias(es). The user won't notice any changes, because they will send their requests to the same URL(s).

When WebSphere is installed, it automatically creates a default virtual host, called default_host, shown in Figure 14-4 on page 493. The default virtual host is configured to match requests to ports 80 or 9080 (the WebSphere Application Server's default port) with any incoming host name. Many users will not need to create or change virtual hosts, since the default_host will be sufficient.

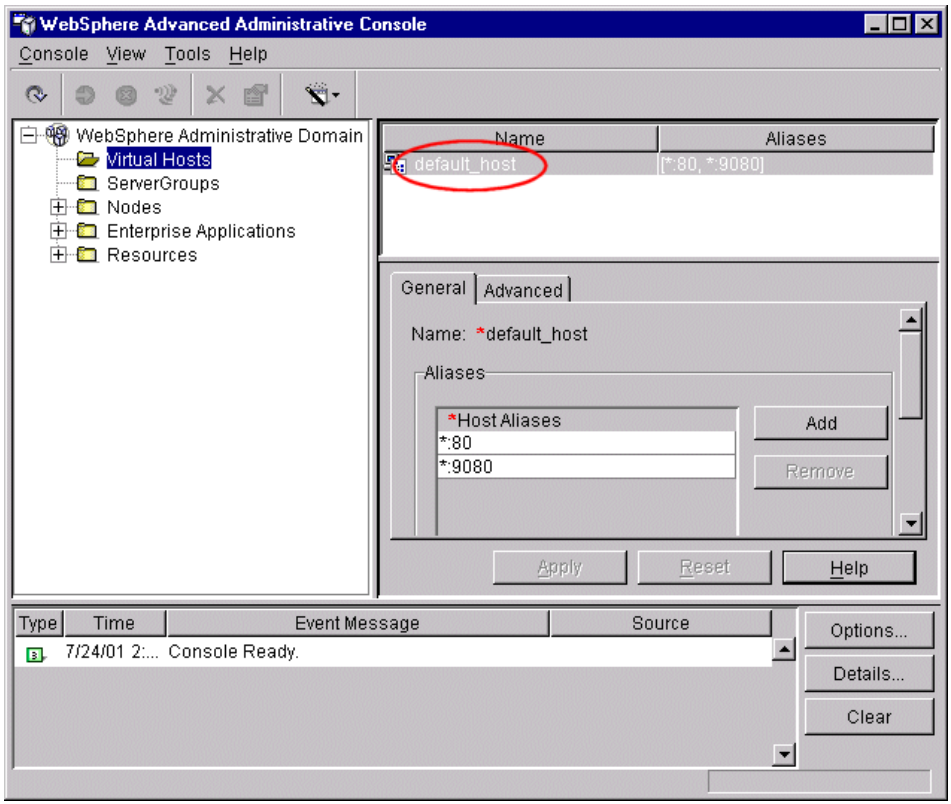


Figure 14-4 Virtual hosts

Each virtual host is associated with a Web module or Web modules. This association is established when the J2EE enterprise application is installed (see Chapter 19, “Deploying an application” on page 687). This can also be changed via the administration console after the application is installed. By associating a virtual host with a Web module we are saying that requests that match the host aliases for the virtual host should be processed by servlets/JSPs in this Web module. To determine whether a Web module can handle the request, the plug-in also checks the URI of the request against the URIs for the Web module. See 14.1.1, “How an HTTP request is processed by the plug-in” on page 484 for more information.

A single virtual host, for example default_host, can be used to direct requests to multiple J2EE enterprise applications from a single Web server. For this to work, all of the applications’ URIs have to be unique, that is, /servlet/* cannot be mapped to more than one application on the same Web server. When it isn’t feasible to have unique URIs, the administrator can use multiple virtual hosts to overcome the problem. The administrator would associate a different virtual host

with the Web module for each enterprise application. Each virtual host would contain a different set of aliases, that is, host names and or port numbers, which the plug-in would match against in order to decide which enterprise application should handle an incoming request.

14.2.1 Create virtual host

For many users this step will be unnecessary. WebSphere creates a default virtual host when it is installed, as we saw in Figure 14-4 on page 493. This matches the request received on ports 80 and 9080 for any incoming host name.

If the administrator decides that more precise filtering of the incoming requests is required or the Web server needs to support extra ports, for example, port 443 for incoming SSL requests, the administrator can create a new virtual host or modify an existing one.

To create a new virtual host:

1. Right click **Virtual Hosts**.
2. Click **New...**. See Figure 14-5.

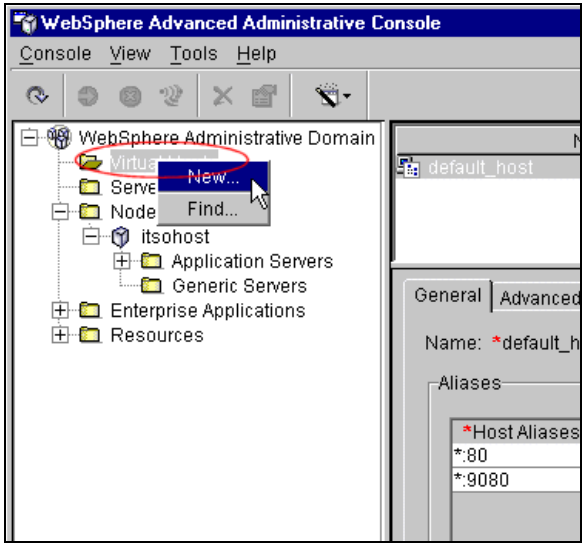


Figure 14-5 Creating a new virtual host

3. This will display the window shown in Figure 14-6.

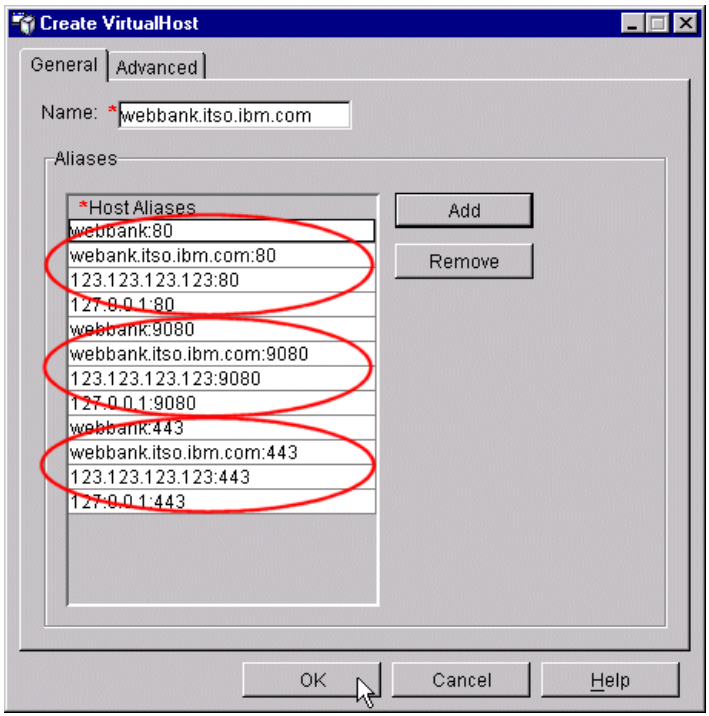


Figure 14-6 Settings for the new virtual host

4. Define the name of the virtual host in the Name field.
5. Add the required host aliases by clicking **Add** and entering the required host name and port number. The host name and port number should be separated by a colon, for example, myHost:80.
6. Click **OK** once all of the required host aliases have been added.
7. Regenerate the plug-in's XML configuration file, as described in 14.1.2, "Generating the plug-in's XML configuration file" on page 487.

Figure 14-6 has three groups of aliases. The ones for port 80 will match requests received on the default Web server port. Those on port 9080 will be used if requests are sent directly to the application server in a development environment. This assumes that the transport port of the application server is set to the default port 9080. The final group, on port 443, will be used to match incoming SSL requests received by the Web server.

Simple wild cards can be used in the host aliases. A * may be used for the host name or for the port or both. A * for both means that any request will match this rule. If no port is specified in the definition, the default HTTP port of 80 is assumed.

In Figure 14-6 on page 495, there are a number of host name synonyms for each port. They are:

- The short form of the host name
- The fully qualified host name
- The local host alias, that is the request is sent to the local machine
- The IP address of the Web server
- The loopback address

If you are not using wild cards for host names, then defining these synonyms will guarantee that the host name will match the virtual host, no matter which synonym is used in the browser.

14.2.2 MIME types

The window shown in Figure 14-7 on page 497 specifies Multi-Purpose Internet Mail Extensions (MIME) mappings for a given virtual host. MIME mappings associate a file name extension with a type of data file (text, audio, image). From here we can add, edit or remove mappings.

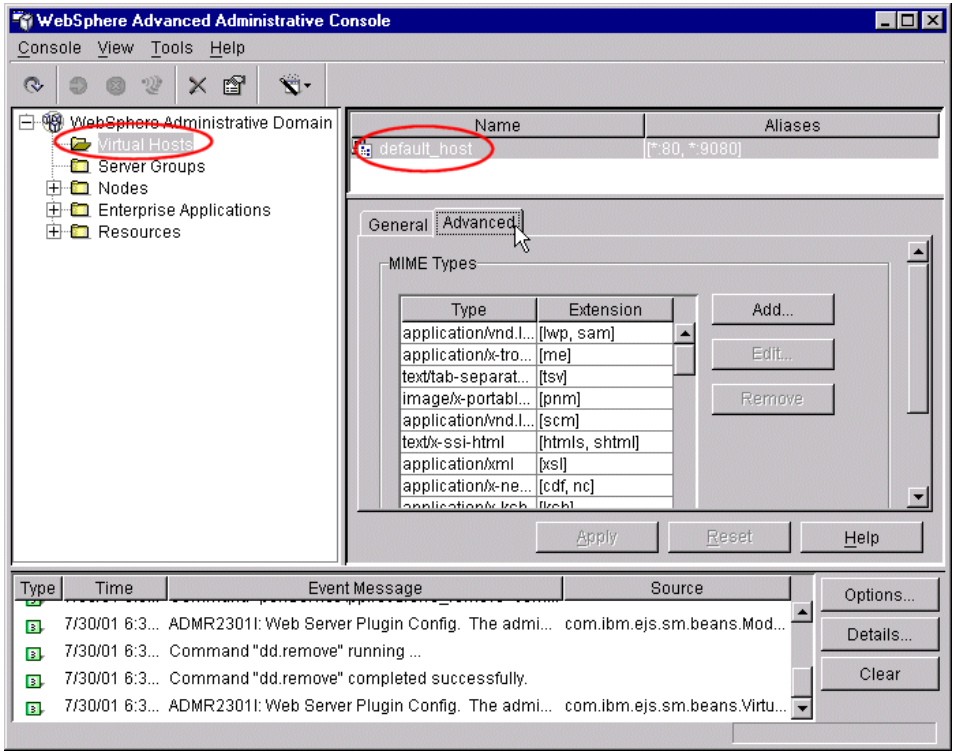


Figure 14-7 Viewing MIME types mappings the for a virtual host

To access the virtual hosts MIME Types window:

1. Click **Virtual Hosts**.
2. Select the virtual host that you want access, default_host in our case.
3. Click the **Advanced** tab.

Note: WebSphere comes with a set of predefined MIME types that normally don't need to be modified.

14.3 Configuring transports for WebSphere

WebSphere transports define the host, port, and protocol (HTTP or HTTPS) used by plug-in to communicate with an application server or clone.

Transport properties specify:

- How to manage the set of connections; for example, how many concurrent requests to allow.
- Whether to secure the connections between the Web server and WebSphere with SSL.
- Host and IP information for the transport participants.

To avoid port contention, the administrator must specify a unique port number for each application server or clone on a given machine.

14.3.1 The internal transport

For applications in a test or development environment (in other words, a non-production environment), a developer can use the WebSphere Application Server's internal HTTP transport system to serve servlets/JSPs without a Web server. Simply use the application server's HTTP transport port (typically on port 9080).

For example, to serve the "snoop" servlet without an HTTP server, use the URL:

```
http://<your.server.name>:<port>/servlet/snoop
```

with `<port>` being the application server's transport port number (typically 9080) and `<your.server.name>` being localhost if the application server is on the local machine.

For a production environment, do not use the internal transport, since it lacks the performance and security available when using a Web server plug-in.

14.3.2 Creating or removing a transport

Most of the time the administrator will not have to create new HTTP transports. When the administrator creates a new application server or clone, WebSphere will automatically create a non-SSL transport on a unique port for the application server or clone. If the administrator defines more than one transport of a given type (HTTP or HTTPS) for an application server or clone, the plug-in will only use the first definition and ignore the rest.

The main reasons for creating or removing a transport are:

- To enable SSL between the plug-in and an application server or clone.
- To control the mapping of incoming requests to a particular transport protocol, that is to HTTP or HTTPS, as shown in Table 14-1 on page 486.

14.3.3 Adding a non-SSL transport

To add a new non-SSL transport:

1. Select the application server that requires the transport.
2. Select the **Services** tab.
3. Select the **Web Container Service**.
4. Click on **Edit Properties...**, as seen Figure 14-8.

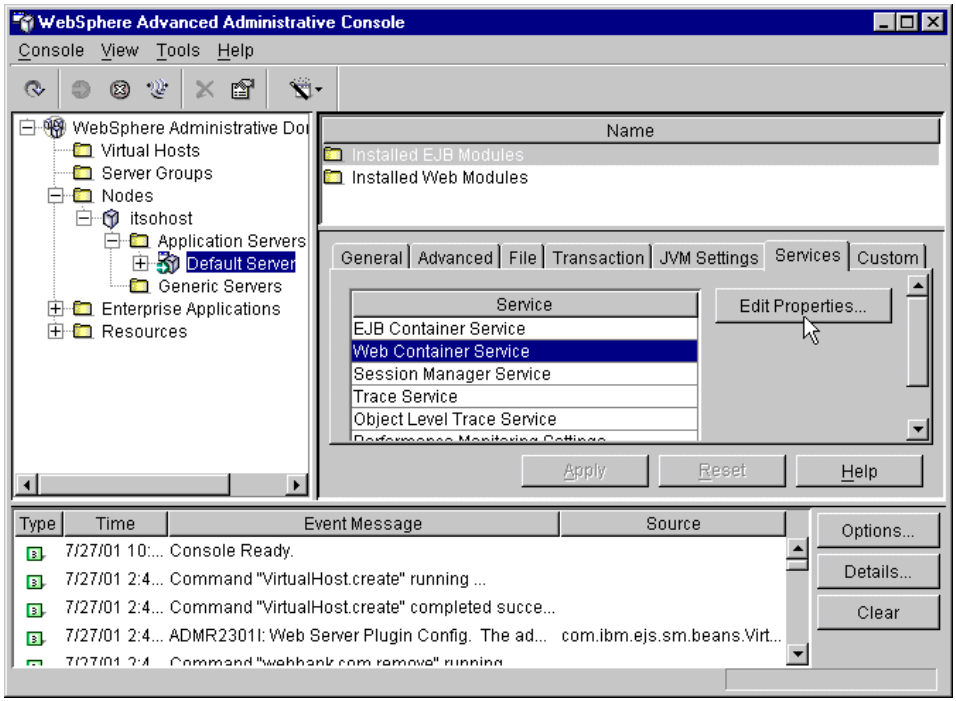


Figure 14-8 Access the transport settings

The Web Container Service window should appear as shown in Figure 14-9 on page 500. This window shows the transports that have been created for this application server. In our example, all of transports have been deleted so we can demonstrate how to add a new transport from the beginning.

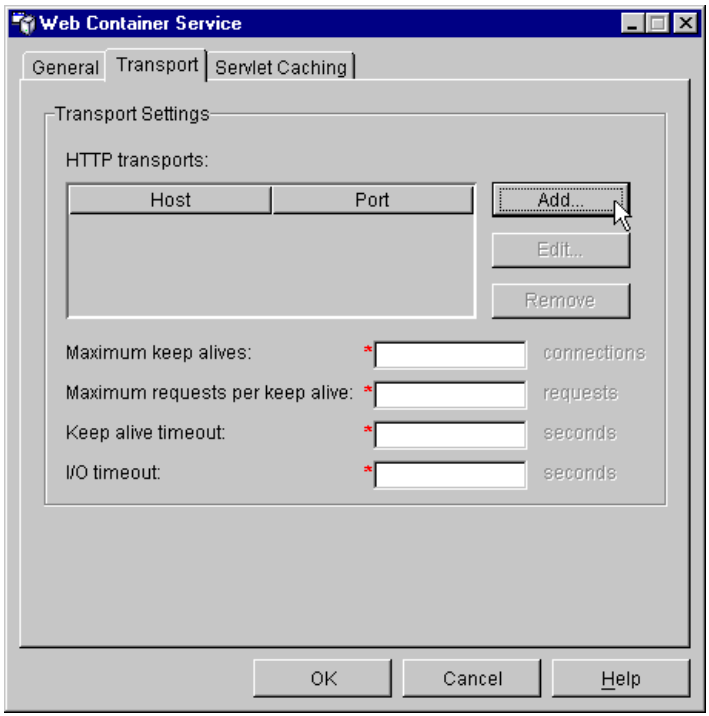


Figure 14-9 Adding a new HTTP transport

5. To add a new HTTP transport, click the **Add...** button. This will display the window seen in Figure 14-10 on page 501. In this window you can select the type of transport you want. In this example we will create a non-SSL transport, that is one for HTTP. For information on creating an SSL transport, see 14.3.4, “Adding an SSL transport” on page 504.

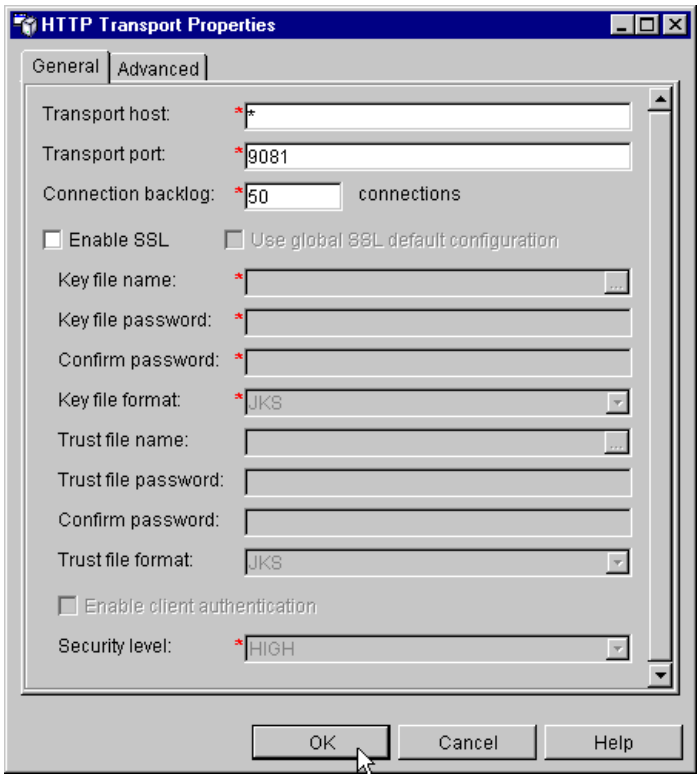


Figure 14-10 Configuring an HTTP transport

The administrator needs to define:

- Transport host

  This is the host name or IP address of the machine on which the application server is running. This is normally set to *. When the plug-in's XML configuration file is regenerated * is converted to the host name on which the application server is running. If the host name is not set to * then the value entered in this field will be used as the target host by the plug-in.

- Transport port

  This is the TCP/IP port number that the plug-in will use when requests are sent to the application server. To avoid port contention, a unique port number must be used for each transport. If the Web server is running on the same machine, take care that the transport's port number doesn't clash with the ports used by the Web server, such as port 80, port 443, and so on.

- Connection backlog

  This is the maximum number of outstanding connect requests that the operating system will buffer while it waits for the application server to accept the connections. If a client attempts to connect when this operating system buffer is full, the connect request will be rejected.

  Set this value to the number of concurrent connections that the administrator would like to allow. Keep in mind that a single client browser might need to open multiple concurrent connections. However, also keep in mind that increasing this value consumes more kernel resources.

  The value of this property is specific to each transport.

  On some operating systems increasing this parameter may require changes to the kernel settings before the queue length is really increased. Please refer to the documentation for your operating system.

6. Click **OK** when you have finished.

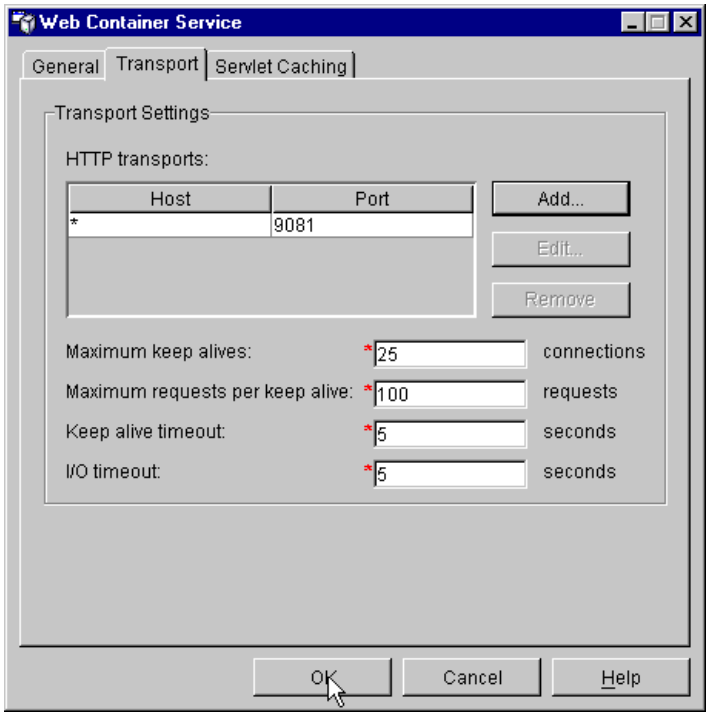


Figure 14-11 Setting transport limits

Please refer to Figure 14-11. Now that a transport has been created, the administrator has to set certain maximums and time outs. This values are:

- Maximum keep alives

  The maximum number of concurrent keep-alive (persistent) connections across all HTTP transports. HTTP 1.1 allows multiple requests to be sent on the same TCP/IP connection. The default value is 90% of the maximum number of threads in the Web container thread pool. This prevents all of the threads from being held by keep-alive connections so that there are threads available to handle new incoming connect requests.

- Maximum requests per keep alive

  The maximum number of requests that can be processed on a single keep-alive connection.

- Keep alive time out

  The maximum time (in seconds) to wait for the next request on a keep-alive connection.

- I/O timeout

  The maximum time (in seconds) to wait when trying to read or write data during a request.

The value of these properties, after being specified for the first transport, will be applied globally to all other HTTP transports in the administrative domain.

7. Once these values have been set, click **OK** to return to the Application Server Properties window, as seen in Figure 14-12 on page 504.

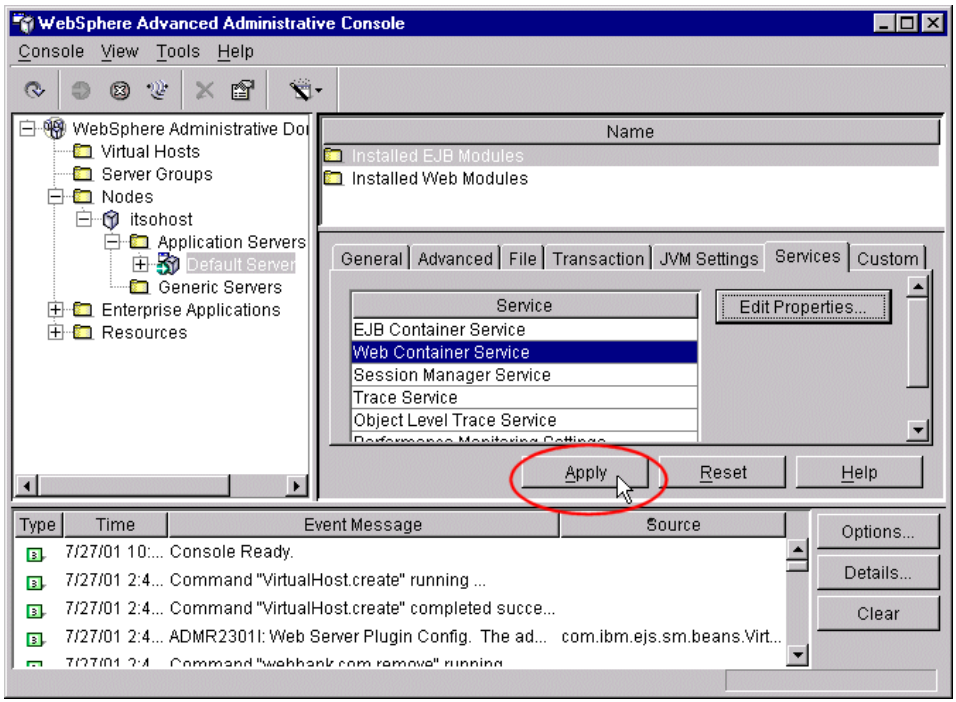


Figure 14-12 Save the new transport

To make the transport changes operational:

8. Click **Apply** as shown in Figure 14-12. If you don't click Apply the changes will be lost.
9. Regenerate the plug-in's XML configuration file and install it in the appropriate directory, as described in 14.1.2, "Generating the plug-in's XML configuration file" on page 487.
10. Restart the application server.

14.3.4 Adding an SSL transport

An SSL transport may be required in environments where:

- Messages sent between the Web server and the application server contain sensitive or valuable information.
- The client requires the application server to prove that it is a valid/known server. SSL server-side authentication is carried out when the client connects to the application server using SSL. The server sends a digital certificate to the client to prove its identity. This avoids the "man in the middle" problem

where messages are intercepted by a third party pretending to be the actual server.

- The Web server must be authenticated before it is allowed to communicate with the application server. This restricts access to the application server to known clients using client-authenticated SSL. If client-authenticated SSL is enabled, both the application server and the plug-in exchange certificates to prove their identities. Once the identity of the application server has been accepted by the client, the client will identify itself by sending its own certificate to the application server. This ensures that rogue/unknown clients cannot access the application server.

Creating an SSL transport is done using the same windows as a non-SSL transport. It is possible to create an SSL transport:

- Using the global default SSL configuration
- Using a specific SSL configuration
- Either of the above with client authentication enabled

Each of these options are discussed below. It is recommended that you read 21.9, “Global default SSL configuration” on page 830 before proceeding, as it contains definitions used in the other options.

You can also find another example of creating an SSL transport in 9.9, “Configure WebSphere HTTP transport for SSL” on page 245.

Note: The following covers the configuration of the WebSphere SSL transport only. Unless you are using the dummy server key and trust files, which are intended for development and test use only, you will have to generate and populate key and trust files. For an example of how this done, see 9.9, “Configure WebSphere HTTP transport for SSL” on page 245. There is also more information available in the InfoCenter article "Configuring SSL in WebSphere Application Server".

Creating an SSL transport using the default SSL configuration

To create an SSL transport using the global default SSL configuration:

1. Go to the HTTP Transport Properties window, as seen in Figure 14-13 on page 506. You can access this window as described in 14.3.3, “Adding a non-SSL transport” on page 499.

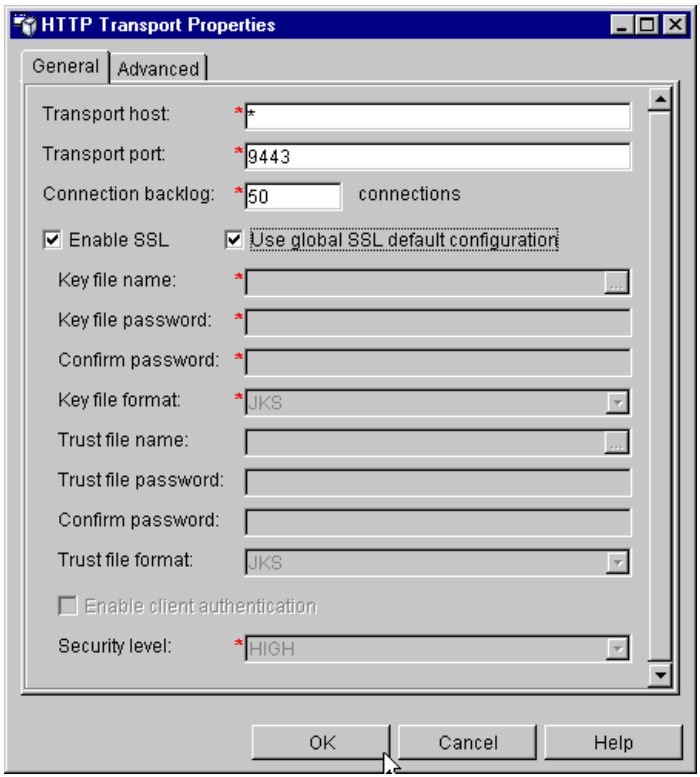


Figure 14-13 Using the global default SSL configuration

2. Enter the transport host name. Refer to 14.3.3, “Adding a non-SSL transport” on page 499 for more information.
3. Select a transport port number. Make sure that this port is unique on the WebSphere machine. If WebSphere is on the same machine as the Web server then use a port other than port 443 since the Web server will by default listen on port 443 for incoming SSL browser requests.
4. Check **Enable SSL**.
5. Check **Use global SSL default configuration**. This will grey out the rest of the SSL options.
6. Click **OK**.
7. Click **OK** on the Web Container Service window.
8. Save the changes by clicking **Apply**.
9. Regenerate the plug-in XML file, as described in 14.1.2, “Generating the plug-in’s XML configuration file” on page 487.

10. Check that the keyring and stashfile properties in the plug-in XML file are correct for your installation, as seen in Example 14-2 on page 507. If the Web server is on a different operating system from WebSphere or it isn't using the default directories settings, the location of the keyring and stash files will not match the settings generated from the WebSphere Administrative Console. If these settings are incorrect, you need to edit the generated file and correct them.

Example 14-2 WebSphere plug-in XML properties for an HTTPS transport

```
<ServerGroup Name="Default Server">
   <Server CloneID="svoasodq" Name="Default Server">
      <Transport Hostname="itsohost" Port="9443" Protocol="https">
         <Property name="keyring"
            value="D:\WebSphere\AppServer\etc\plugin-key.kdb" />
         <Property name="stashfile"
            value="D:\WebSphere\AppServer\etc\plugin-key.sth" />
      </Transport>
   </Server>
</ServerGroup>
```

11. Install the plug-in XML file into the appropriate directory (see 14.1.3, "Installing the plug-in XML configuration file" on page 490).
12. Restart the application server.

Creating an SSL transport using a specific configuration

The administrator can choose to configure an application server's SSL transport using settings that are unique to that application server, that is, they can override the global default SSL configuration.

To create an SSL transport using an individual SSL configuration:

1. Go to the HTTP Transport Properties window, as seen in Figure 14-14 on page 508. You can access this window as described in 14.3.3, "Adding a non-SSL transport" on page 499.

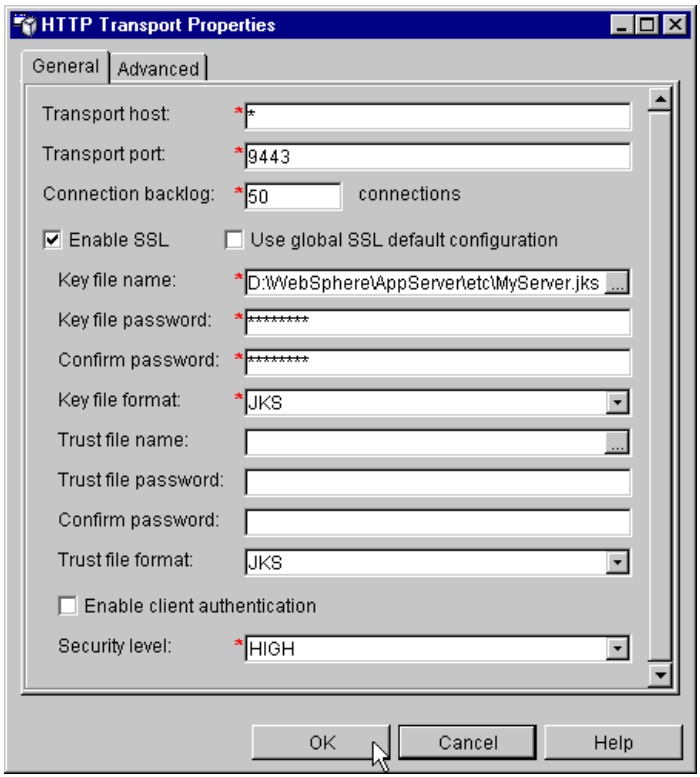


Figure 14-14 Using an individual SSL configuration

2. Enter the transport host name. Refer to 14.3.3, “Adding a non-SSL transport” on page 499 for more information.
3. Select a transport port number. Make sure that this port is unique on the WebSphere machine. If WebSphere is on the same machine as the Web server then use a port other than port 443 as the Web server will by default listen on port 443 for incoming SSL browser requests.
4. Check **Enable SSL**.
5. Make sure **Use global SSL default configuration** is un-checked to enable the rest of the SSL options.
6. Enter the key file name (including path) for this application server.

Note: This is not the same key file as the one used for the WebSphere plug-in since it has a different format. The plug-in uses CMS format, whereas the WebSphere Application Server use the JKS format by default.

7. Enter the key file password and confirm password for the application server's key file.
8. Click **OK**.
9. Click **OK** in the Web Container Service window.
10. Save the changes by clicking **Apply**.
11. Regenerate the plug-in XML file, as described in 14.1.2, "Generating the plug-in's XML configuration file" on page 487.
12. Check that the keyring and stashfile properties in the plug-in XML file are correct for your installation, as seen in Example 14-2 on page 507. If the Web server is on a different operating system from WebSphere or it isn't using the default directories settings, the location of the keyring and stash files will not match the settings generated from the WebSphere Administrative Console. If these settings are incorrect, you need to edit the generated file and correct them.
13. Install the plug-in XML file into the appropriate directory (see 14.1.3, "Installing the plug-in XML configuration file" on page 490).
14. Restart the application server.

Create an SSL transport that has client authentication enabled

The administrator can choose to enable client authentication for the SSL connection between the plug-in and the application server. This restricts access to the application server to known clients using a similar mechanism to SSL server-side authentication. Once the identity of the server has been accepted by the client, the client will identify itself by sending its own certificate to the server. This ensures that rogue or unknown clients cannot access the application server.

Client authentication can be enabled in either the global default SSL configuration or for an individual application server. This example concentrates on an individual application server but the process is essentially the same if we wanted to change the global defaults.

To create an SSL transport with client authentication enabled:

1. Go to the HTTP Transport Properties window, as seen in Figure 14-15 on page 510. You can access this window as described in 14.3.3, "Adding a non-SSL transport" on page 499.

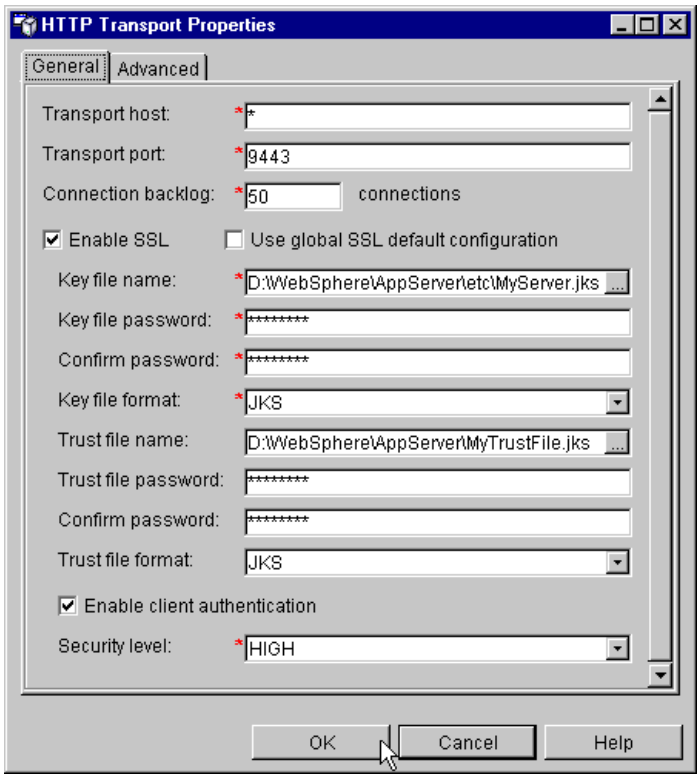


Figure 14-15 Using an SSL configuration with client authentication enabled

2. Enter the transport host name. Refer to 14.3.3, “Adding a non-SSL transport” on page 499 for more information.
3. Select a transport port number. Make sure that this port is unique on the WebSphere machine. If WebSphere is on the same machine as the Web server then use a port other than port 443, since the Web server will by default listen on port 443 for incoming SSL browser requests.
4. Check **Enable SSL**.
5. Make sure **Use global SSL default configuration** is un-checked to enable the rest of the SSL options.
6. Enter the key file name (including path) for this application server.

Note: This is not the same key file as the one used for the WebSphere plug-in since it has a different format. The plug-in uses CMS format, whereas the WebSphere Application Server use the JKS format by default.

7. Enter the key file password and confirm password for the application server's key file.
8. Optionally enter the trust file name, that is the file containing the signer certificates. The administrator may elect to put the personal and signer certificates into the key file, in which case this field should be left blank.
9. Optionally enter the trust file password and confirm password for the applications server's trust file. This is required only if the administrator configures a separate trust file.
10. Check **Enable client authentication**.
11. Click **OK**.
12. Click **OK** on the Web Container Service window.
13. Save the changes by clicking **Apply**.
14. Regenerate the plug-in XML file, as described in 14.1.2, "Generating the plug-in's XML configuration file" on page 487.
15. Check that the keyring and stashfile properties in the plug-in XML file are correct for your installation. See Example 14-2 on page 507. If the Web server is on a different operating system from WebSphere or it isn't using the default directories settings, the location of the keyring and stash files will not match the settings generated from the WebSphere Administration Console. If these settings are incorrect you need to edit the generated file and correct them.
16. Install the plug-in XML file into the appropriate directory. See 14.1.3, "Installing the plug-in XML configuration file" on page 490.
17. Restart the application server.

14.3.5 Browser to Web server SSL support

Configuring the Web server to handle SSL communications with the browser is a separate activity from configuring SSL between the plug-in and the application server. This chapter doesn't address how to set up browser-to-Web server SSL communications except to note that the Web server and the plug-in may use a separate set of key and trust files. In some configurations, for example, when using the IBM HTTP Server, it is possible to use the same key and trust files for the Web server and the plug-in. In others, this may not be possible.

Because the trust policies of the Web server and the plug-in are likely to be different, it is recommended that they maintain separate sets of key and trust files, even when it is possible to share them. For example, the administrator may want to allow many browsers to connect to the Web server's HTTPS port, whereas they may only want to allow a small, well-known number of WebSphere plug-ins to connect directly to a WebSphere Application Server's HTTPS port.

Please consult your Web server's documentation for information on how to enable browser-to-Web server SSL communication. Also see 21.8, "Using client certificate-based authentication" on page 792.

Note: In order for the client certificate (the certificate from the browser) to be forwarded by the plug-in to the application server, client authentication must be enabled at the Web server. Enabling SSL client authentication between the plug-in and the application server itself is not required unless the administrator wants the WebSphere Web server plug-in to be authenticated (or any other clients connecting directly to the application server over SSL).

15

Configuring session management

This chapter discusses HTTP session support in WebSphere Application Server V4.0. Session support allows a Web application developer to maintain state information across multiple user visits to the application.

In many Web applications, users dynamically collect data as they move through the site based on a series of selections on pages they visit. Where the user goes next, and what the application displays as the user's next page (or next choice) may depend on what the user has chosen previously from the site. For example, if the user clicks the checkout button on our site, the next page must contain the user's shopping selections.

In order to do this, a Web application needs a mechanism to hold the user's state information over a period of time. However, HTTP alone doesn't recognize or maintain a user's state. HTTP treats each user request as a discrete, independent interaction.

The Java servlet specification provides a mechanism for servlet applications to maintain a user’s state information. This mechanism, known as a session, addresses some of the problems of more traditional strategies such as a pure cookie solution. It allows a Web application developer to maintain all user state information at the host, while passing minimal information back to the user via cookies, or another technique known as URL encoding.

15.1 Version 3.5 vs. Version 4.0 session management

The following improvements have been made to session management WebSphere V4.0:

- Servlet 2.1 specification is no longer supported
- In line with the Servlet 2.2 specification:
  - Sessions can only be accessed by a single Web application. This is a change from WebSphere V3.5 where access was restricted to a single user but could be shared across multiple Web applications.
  - SSL session IDs can be used as a session identifier; see 15.3.2, “SSL ID tracking” on page 518.
  - Session objects are not locked when they are accessed by the servlet or JSP. Session locking was provided in some earlier versions of WebSphere because session affinity was not supported. Now, it is the programmer’s responsibility to ensure thread safe access if the session object is accessed by multiple threads.

Notes:

1. You must use an affinity mechanism to ensure that a client’s requests are routed to the same application server JVM. The WebSphere Web server plug-in provides this affinity mechanism. You could also use the IBM Network Dispatcher.
2. WebSphere Application Server V3.5 with Fix Pack 2 (Version 3.5.2) introduced full support for the Servlet 2.2 API, with the exception of the J2EE extensions to the specification.

- Session affinity has been improved; see 15.6, “Session affinity” on page 527.
- DB2 now supports larger page sizes. It supports 4 KB, 8 KB, 16 KB, and 32 KB page sizes. Session management can be configured to utilize this DB2 feature for improving performance with larger session objects. See 15.9.1, “Using larger DB2 page sizes” on page 553.
- Improved randomization of session IDs.

  WebSphere V4.0 uses the JCE package provided by the IBM security team to construct a more random session ID.
- Persistent session management can be configured using manual settings as well as the predefined ones; see “Manual tuning of persistent sessions” on page 535

- Use of security association with session is now configurable.

  WebSphere V3.5 would always associate the user identity of the servlet request with the session and throw an unauthorized exception they didn't match. In WebSphere V4.0, this has been made configurable; see "Enabling session security" on page 552.
- Added time based session writes; see 15.7.5, "Time based writes to the session database" on page 540.
- Added the ability to defer the clean up of invalid persistent sessions to a time in the off hours; see 15.8, "Invalidating sessions" on page 549.
- Can choose to write all of the session data instead of just the parts that have changed; see 15.7.8, "What is written to the persistent session database" on page 545.
- Use of the Merant JDBC drivers extends the list of supported databases.

  The Merant JDBC driver added Informix and MicroSoft SQLServer to the list of supported databases in which WebSphere can store persistent sessions.

15.2 Accessing session management properties

To access session management settings for an application server:

1. Select the required application server or server group.
2. Click the **Services** tab.
3. Select the **Session Manager Service** from the Services list.
4. Click the **Edit Properties...** button, as shown in Figure 15-1.

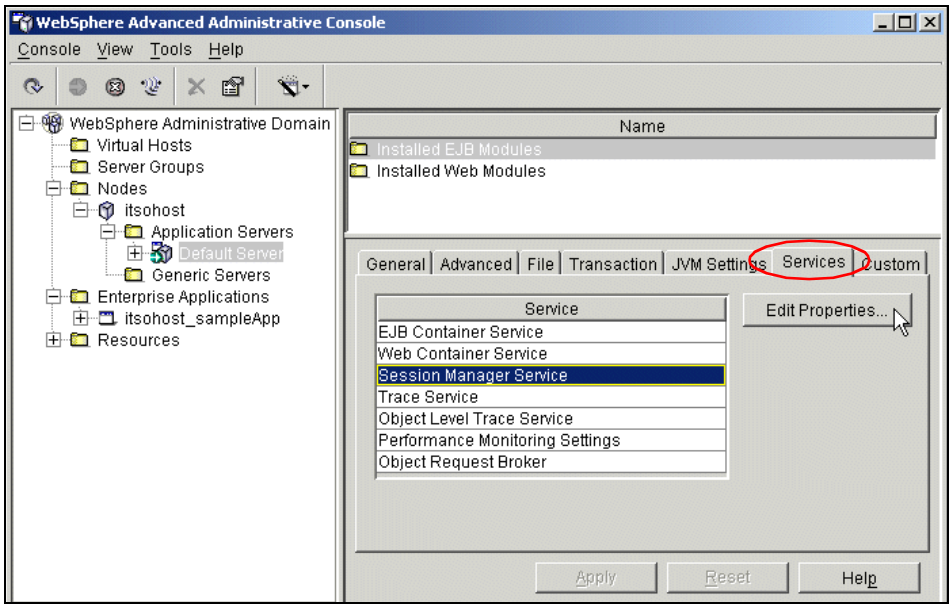


Figure 15-1 Accessing session management properties

This will display the window in Figure 15-2 on page 518. From this window you can control all of the session properties.

Note: Figure 15-1 shows how to access the session management properties for an application server. To set the session management properties for a server group or clone, select the server group that you are interested in instead of the application server. Session management properties are typically administered at the server group level and not on the individual clones.

15.3 Session identifiers

WebSphere session support keeps the user's session information on the server. WebSphere passes the user an identifier known as a session ID, which correlates an incoming user request with a session object maintained on the server.

Note: The example session IDs provided in this chapter are for illustrative purposes only and are *not* guaranteed to be absolutely consistent in value, format and length.

15.3.1 Choosing a session tracking mechanism

WebSphere supports three approaches to track sessions:

- SSL session identifiers
- Cookies
- URL encoding/rewritting

It is possible to select all three options for a Web application. If you do this then:

- SSL session identifiers are used in preference to cookie and URL rewriting.
- Cookie are used in preference to URL rewriting.

Note: If SSL session ID tracking is selected, it is recommended that you also select cookies or URL rewriting so that session affinity can be maintained.

To set or change the session mechanism type:

1. Open the Session Manager Service properties window, as described in 15.2, “Accessing session management properties” on page 515.
2. Select the session tracking mechanism that you require.
3. Click **OK**, as shown in Figure 15-2.

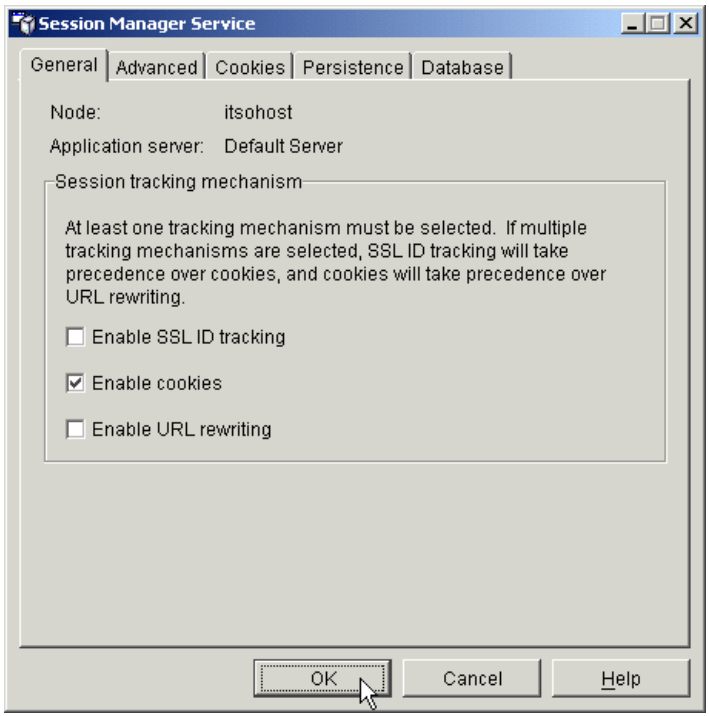


Figure 15-2 Session manager service - general properties

4. Click **Apply** to save the application server changes.
5. Restart the application server or server group.

15.3.2 SSL ID tracking

When SSL ID tracking is enabled for requests over SSL, SSL session information is used to track the HTTP session ID.

Because the SSL session ID is negotiated between the Web browser and HTTP server, it cannot survive an HTTP server failure. However, the failure of an application server does not affect the SSL session ID. Of course, if session persistence is not configured, the session itself is lost. In environments that use WebSphere Edge Server with multiple HTTP servers, an affinity mechanism must be used when SSL session ID is to be used as the session tracking mechanism.

SSL tracking is supported only for the IBM HTTP Server and iPlanet Web servers. The lifetime of an SSL session ID can be controlled by configuration options in the Web server. For example, in the IBM HTTP Server, the configuration variable SSLV3TIMEOUT must be set to allow for an adequate lifetime for the SSL session ID. Too short an interval could result in premature termination of a session. Also, some Web browsers might have their own timers that affect the lifetime of the SSL session ID. These Web browsers might not leave the SSL session ID active long enough to be useful as a mechanism for session tracking.

When the SSL session ID is to be used as the session tracking mechanism in a cloned environment, either cookies or URL rewriting must be used to maintain session affinity. The cookie or rewritten URL contains session affinity information that enables the Web server to properly route requests back to the same server once the HTTP session has been created on a server. The SSL ID is not sent in the cookie or rewritten URL but is derived from the SSL information.

Disadvantages of SSL ID tracking

The main disadvantage of using SSL ID tracking is the performance hit of using SSL. If you have a business requirement to use SSL, then this would be a good choice. If you don't have such a requirement, it is probably a good idea to consider using cookies instead.

As discussed previously, Web server and Web browser SSL session timeout settings may also limit the usefulness of SSL ID tracking.

15.3.3 Cookies

Many sites choose cookie support to pass the user's identifier between WebSphere and the user. WebSphere Application Server session support generates a unique session ID for each user, and returns this ID to the user's browser via a cookie.

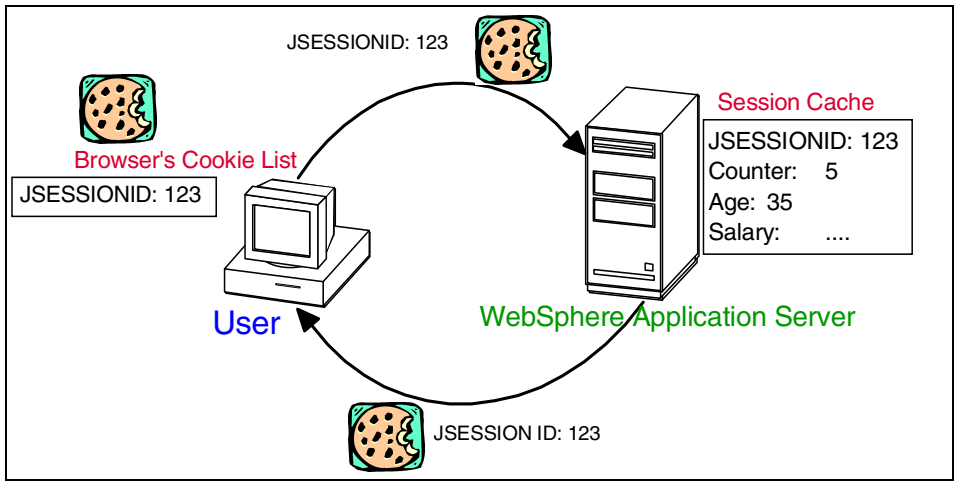


Figure 15-3 Cookie overview

A cookie consists of information embedded as part of the headers in the HTML stream passed between the server and the browser. The browser holds the cookie, and returns it to the server whenever the user makes a subsequent request. By default, WebSphere defines its cookies so they are destroyed if the browser is closed. This cookie holds a session identifier. The remainder of the user's session information resides at the server.

The Web application developer uses the HTTP request object's standard interface to obtain the session:

```
HttpSession session = request.getSession(true);
```

WebSphere places the user's session identifier in the outbound cookie whenever the servlet completes its execution, and the HTML response stream returns to the end user. Again, neither the cookie nor the session ID within it require any direct manipulation by the Web application. The Web application only sees the contents of the session.

Cookie disadvantages

The main disadvantage with cookies is that some users, either by choice or mandate, disable them from within their browser.

Cookie settings

When cookies are used as the session tracking mechanism, then the administrator has a number of configuration options, as shown in Figure 15-4:

- Cookie name

  The cookie name is hard-coded to JSESSIONID. This is required by the Servlet 2.2 specification for all cookie-based session IDs.

- Cookie domain

  This value will dictate to the browser whether or not to send a cookie to particular servers. For example, if you specify a particular domain, the browser will only send back session cookies to hosts in that domain. The default value in the session manager restricts cookies to the host that sent them.

  Note: In WebSphere Application Server V4.0 (and V3.5), the LTPA token/cookie that is sent back to the browser is scoped by a single DNS domain that is specified when global security is configured. This means that *all* application servers in an *entire* WebSphere Application Server domain must share the same DNS domain for security purposes.

- Cookie path

  The paths (on the server) to which the browser will send the session tracking cookie. Specify any string representing a path on the server. Use "/" to indicate the root directory.

  Specifying a value will restrict the paths to which the cookie will be sent. By restricting paths, you can keep the cookie from being sent to certain URLs on the server. If you specify the root directory, the cookie will be sent no matter which path on the given server is accessed.

- Restrict exchange of cookies to secure sessions

  Enabling the feature will restrict the exchange of cookies only to HTTPS sessions. If it is enabled the session cookie's body includes the 'secure' indicator field.

- Cookie maximum age

  The amount of time that the cookie will live in the client browser.

  WebSphere V4.0 offers two choices:

  – Expire at the end of the current browser session

  – Expire at a configurable maximum age

  If you choose the maximum age option, specify the age in seconds.

For more information on cookie properties, please refer to:

`http://home.netscape.com/newsref/std/cookie_spec.html`

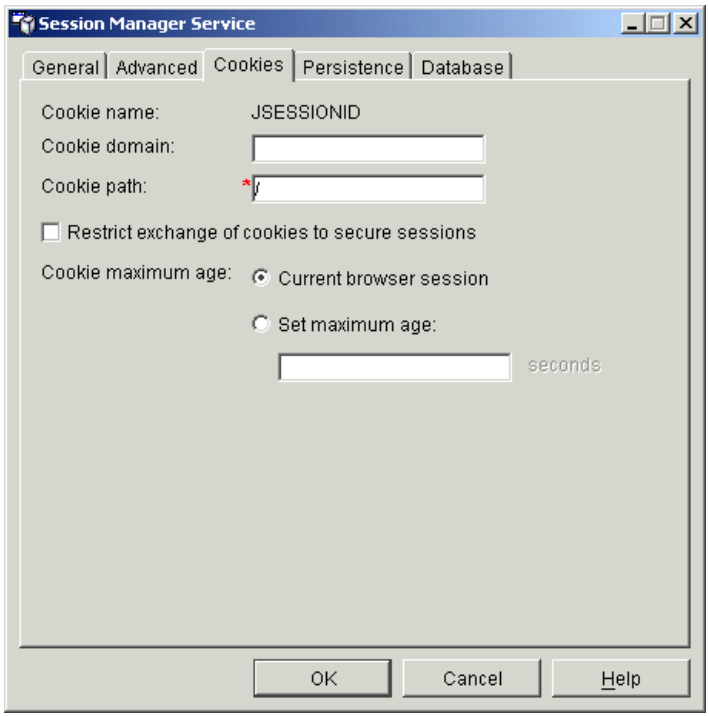


Figure 15-4 Session manager service - cookie properties

15.3.4 URL encoding/rewritting

WebSphere also provides URL encoding for session ID tracking. While session management using SSL IDs or cookies is transparent to the Web application, URL encoding requires the developer to use special encoding APIs, and to set up the site page flow to avoid losing the encoded information.

URL encoding works by actually storing the session identifier in the page returned to the user. WebSphere encodes the session identifier as a parameter on URLs that have been encoded programmatically by the Web application developer. Example 15-1 shows a Web page link with URL encoding.

Example 15-1 Web page link with URL encoding

```
<a href="/store/catalog;$jsessionid=DA32242SSGE2">
```

Note: The Servlet 2.2 specification requires that the name of the URL encoded session tracking variable be jsessionid, as seen in Example 15-1.

When the user clicks this link (Example 15-1) to move to the /store/catalog page, the session identifier passes into the request as a parameter.

URL encoding requires explicit action by the Web application developer. If the servlet returns HTML directly to the requester (without using a JavaServer Page), the servlet calls the API, as shown in Example 15-2, to encode the returning content.

Example 15-2 URL encoding from a servlet

```
out.println("<a href=\");
out.println(response.encodeURL ("/store/catalog"));
out.println("\>catalog</a>");
```

Even pages using redirection (a common practice, particularly with servlet or JSP combinations) must encode the session ID as part of the redirect, as shown in Example 15-3.

Example 15-3 URL encoding with redirection

```
response.sendRedirect(response.encodeRedirectURL("http://myhost/store/catalog")
);
```

When JavaServer Pages (JSPs) use URL encoding, the JSP calls a similar interface to encode the session ID, as shown in Example 15-4.

Example 15-4 URL encoding in a JSP

```
<% response.encodeURL ("/store/catalog"); %>
```

Disadvantages of using URL rewriting

The fact that the servlet or JSP developer has to write extra code is a major drawback over the other available session tracking mechanisms.

URL encoding limits the flow of site pages exclusively to dynamically generated pages (such as pages generated by servlets or JSPs). WebSphere inserts the session ID into dynamic pages, but cannot insert the user's session ID into static pages (.htm or .html pages).

Therefore, after the application creates the user's session data, the user must visit dynamically generated pages exclusively until they finish with the portion of the site requiring sessions. URL encoding forces the site designer to plan the user's flow in the site to avoid losing their session ID.

15.4 Local sessions

Many Web applications use the simplest form of session management: the in-memory, local session cache. The local session cache keeps session information in memory and local to the machine and WebSphere Application Server where the session information was first created.

Local session management doesn't share user session information with other clustered machines. Users only obtain their session information if they return to the machine and WebSphere Application Server holding their session information on subsequent accesses to the Web site.

Most importantly, local session management lacks a persistent store for the sessions it manages. A server failure takes down not only the WebSphere instances running on the server, but also destroys any sessions managed by those instances.

WebSphere allows the administrator to define a limit on the number of sessions held in the in-memory cache via the administrative console settings on the session manager, shown in Figure 15-5 on page 527. This prevents the sessions from acquiring too much memory in the Java Virtual Machine associated with the application server.

The session manager also allows the administrator to permit an unlimited number of sessions in memory. If the administrator enables the *Allow overflow* setting on the session manager via the administrative console, shown in Figure 15-5 on page 527, the session manager permits two in-memory caches for session objects. The first cache contains only enough entries to accommodate the session limit defined to the session manager (1000 by default). The second cache, known as the overflow cache, holds any sessions the first cache cannot accommodate, and is limited in size only by available memory. The session manager builds the first cache for optimized retrieval, while a regular, un-optimized hash table contains the overflow cache. For best performance, define a primary cache of sufficient size to hold the normal working set of sessions for a given Web application server.

Important: You should note that with overflow enabled, the session manager permits an unlimited number of sessions in memory. Without limits, the session caches may consume all available memory in the WebSphere instance's heap, leaving no room to execute Web applications. By way of example, two scenarios under which this could occur are:

- The site receives greater traffic than anticipated, generating a large number of sessions held in memory.
- A malicious attack occurs against the site where a user deliberately manipulates their browser so the application creates a new session repeatedly for the same user.

If you choose to enable session overflow, the state of the session cache should be monitored closely using the WebSphere Resource Analyzer or the WebSphere PMI API. Please refer to Chapter 22, "Monitoring and tuning your runtime environment" on page 839 for more information.

Note: Each Web application will have its own base (or primary) in-memory session cache, and with overflow allowed, its own overflow (or secondary) in-memory session cache.

15.5 Advanced settings for session management

The Session Manager Service - Advanced properties window, shown in Figure 15-5 on page 527, allows the administrator to tune a number of parameters that are important for both local or persistent sessions:

- Maximum in-memory session count

  Specifies the maximum number of sessions to maintain in memory.

  The meaning differs depending on whether you are using local or persistent sessions. For local sessions, this value specifies the number of sessions in the base session table. Check **Allow overflow** to specify whether to limit sessions to this number for the entire session manager, or to allow additional sessions to be stored in secondary tables.

  For persistent sessions, this value specifies the size of the general cache. This value specifies how many session will be cached before the session manager reverts to reading a session from the database automatically. Session manager uses an LRU (least recently used) algorithm to maintain the sessions in the cache.

This value holds when you are using local sessions, persistent sessions with caching, or persistent sessions with manual updates. The manual update cache keeps the last n time stamps representing "last access" times, where n is the maximum in-memory session count value.

- Allow overflow

  Whether to allow the number of sessions in memory to exceed the value specified in the Maximum in-memory session count field. If Allow overflow is not checked, then WebSphere will limit the number of sessions held in memory to this value.

  For local sessions, if this maximum is exceeded and Allow overflow is not checked, then sessions created thereafter will be dummy sessions and will not be stored in the session manager.

  As shown in Example 15-5, the IBM HttpSession extension can be used to react if sessions exceed the maximum number of sessions specified when overflow is disabled.

Example 15-5 Using IBMSession to react to session overflow

```
com.ibm.websphere.servlet.session.IBMSession sess =
   (com.ibm.websphere.servlet.session.IBMSession) req.getSession(true);
if(sess.isOverFlow()) {
   //Direct to a error page…
}
```

- Integrate with WebSphere security

  When security integration is enabled, the session manager will associate the identity of users with their HTTP sessions. See 15.9, “Session security” on page 550 for more information.

  Note: Do not enable this property if the application server contains a Web application that has form based login configured as the authentication method and local operating system is the authentication mechanism. It will cause authorization failures when user agents try to use the Web application.

- Enable protocol switch rewriting

  Defines whether the session ID, added to a URL as part of URL encoding, should be included in the new URL if a switch from HTTP to HTTPS or from HTTPS to HTTP is required. When URL encoding is enabled, this setting determines encoding of HTTPS URLs in HTTP requests, or encoding of HTTP URLs in HTTPS requests. That is, if a servlet is accessed over HTTP and if that servlet is doing encoding of HTTPS URLs, URL encoding will be performed only when protocol switch rewriting is enabled, and vice versa.

- Invalidation timeout

  When a session isn't accessed for this many seconds it can be removed from the in-memory cache and if persistent sessions are used, from the database. This is an important performance tuning property. It directly influences the amount of memory consumed by the JVM in order to cache the session information.

  Note: For performance reasons, the invalidation timer is not accurate "to the second." It is safe to assume that the timer is accurate to within two minutes. When the write frequency is time based, this value should be at least twice as large as the write interval.

  The value of this setting is used as a default when the session timeout is not specified in a Web module's deployment descriptor.

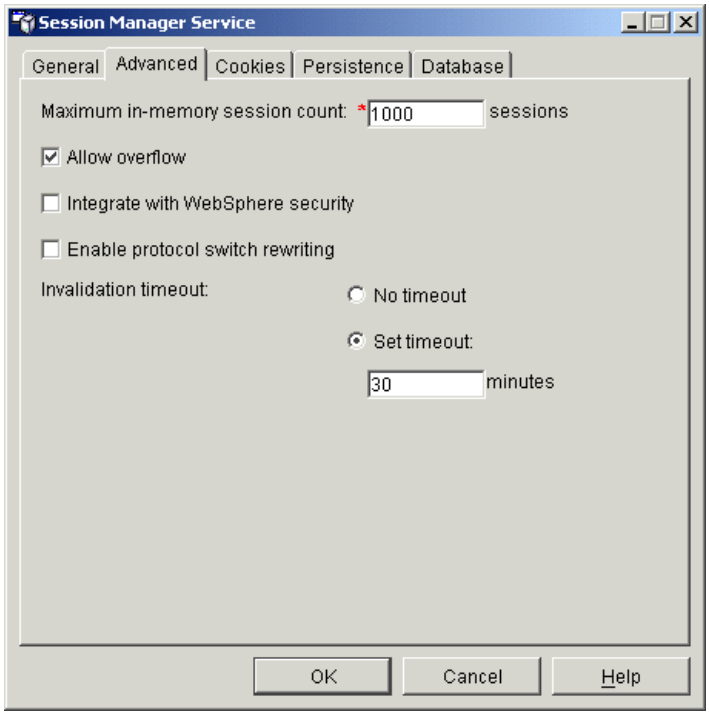


Figure 15-5 Session Manager Service - Advanced properties

15.6 Session affinity

The Servlet 2.2 specification requires that an HTTP session be:

- Accessible only to the Web application that created the session. The session ID can be shared across Web applications, but not the session data.
- Handled by a single JVM for that application at any one time.

This means that in a cloned environment, any HTTP requests that are associated with an HTTP session must be routed to the same Web application in the same JVM. This ensures that all of the HTTP requests are processed with a consistent view of the user's HTTP session. The exception to this rule is when the clone fails or has been shut down. For more information see 15.6.1, "Session affinity and failover" on page 530.

In order to implement this requirement, the affinity mechanism from WebSphere V3.5 that employed a hash algorithm has been replaced with an entirely new mechanism. In WebSphere V4.0, each server ID is appended to the session ID. When an HTTP session is created, its ID is passed back to the browser as part of a cookie or URL encoding. When the browser makes further requests, the cookie or URL encoding will be sent back to the Web server. The WebSphere HTTP plug-in examines the HTTP session ID, in the cookie or URL encoding, extracts the unique ID of the clone handling the session, and forwards the request. In this way WebSphere V4.0 is able to assure that session affinity is maintained.

This can be seen in Figure 15-6 on page 529, where the session ID from the HTTP header (request.getHeader("Cookie")) is displayed along with the session ID from session.getId(). The application server ID from the plugin-cfg.xml file, shown in Example 15-6, is appended to the session ID from the HTTP header. Note also that the first four characters of HTTP header session ID are the cache identifier used to determine the validity of cache entries.

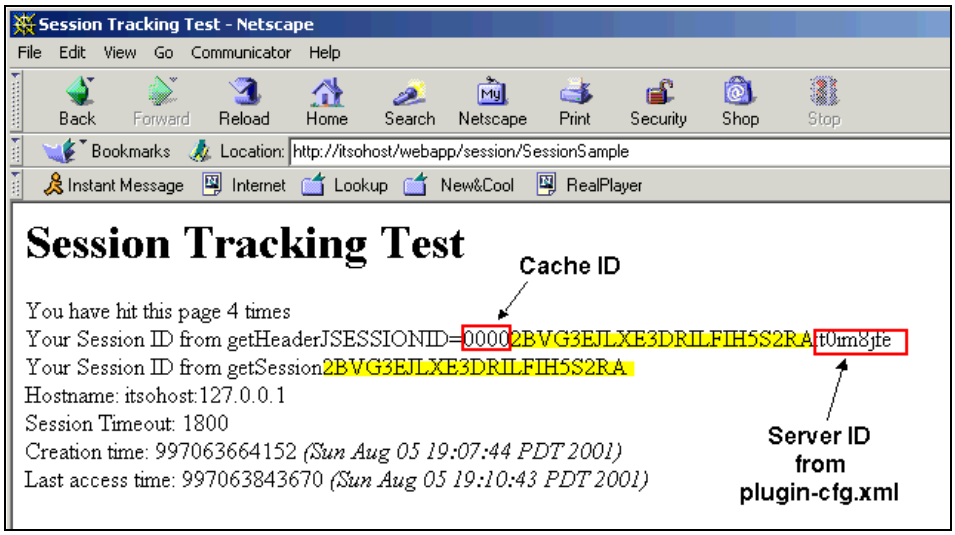


Figure 15-6 Session ID containing the server ID and cache ID

Example 15-6 Server ID from plugin-cfg.xml file

```
<?xml version="1.0"?>
<Config>
   <Log LogLevel="Error" Name="D:\WebSphere\AppServer\logs\native.log"/>
   <VirtualHostGroup Name="default_host">
      <VirtualHost Name="*:80"/>
      <VirtualHost Name="*:9080"/>
   </VirtualHostGroup>
   ...
   <ServerGroup Name="Session App Server">
      <Server CloneID="t0im8jfe" Name="Session App Server">
         <Transport Hostname="itsohost" Port="9081" Protocol="http"/>
      </Server>
   </ServerGroup>
   ...
   <UriGroup Name="Session Sample/SessionSample.war_URIs">
      <Uri Name="/webapp/session"/>
   </UriGroup>
   ...
   <Route ServerGroup="Session App Server"
      UriGroup="Session Sample/SessionSample.war_URIs"
         VirtualHostGroup="default_host"/>
</Config>
```

Note: Session affinity can still be broken if the clone handling the request fails. To avoid losing session data, you can use persistent session management. In persistent sessions mode, cache ID and server ID will change in the cookie when there is a failover or when the session is read from the database. So don't rely on the value of the session cookie remaining the same for a given session.

15.6.1 Session affinity and failover

Server groups provide a solution for failure of an application server. See Chapter 17, "Server groups and workload management" on page 605 for a definition of a server group. Sessions created by clones in the server group share a common persistent session database. Therefore, any clone in the server group has the ability to see any user's session saved to persistent storage. If one of the clones fail, the user may continue to use their session information from another clone in the server group. This is known as failover. Failover works regardless of whether the nodes reside on the same machine or several machines. Please refer to Figure 15-7.

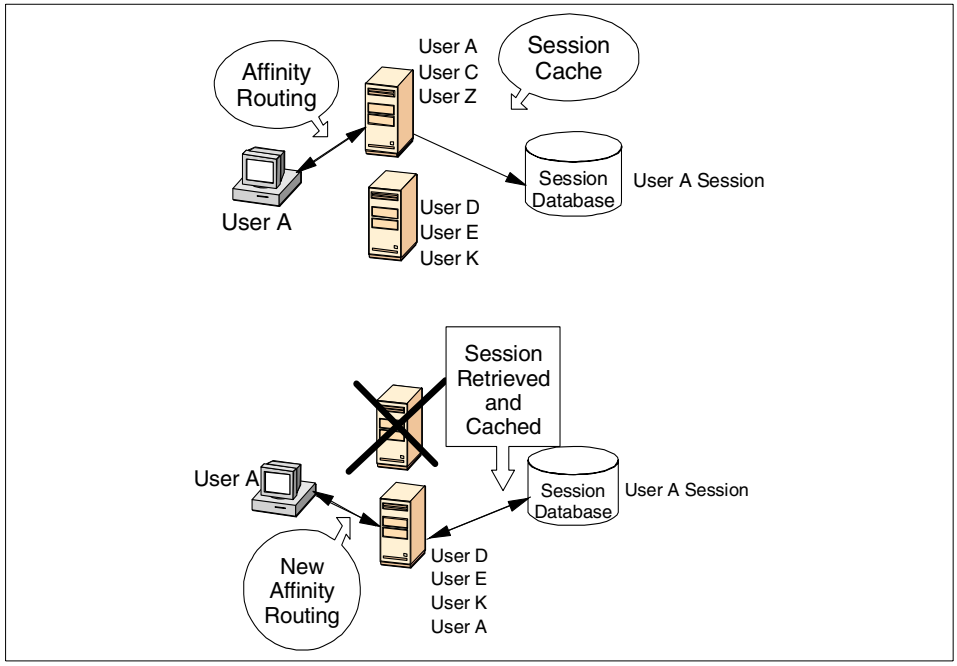


Figure 15-7 Session affinity and failover

Note: According to the Servlet 2.2 specification, only a single clone may control/access a given session at a time.

After a failure, WebSphere redirects the user to another clone, and the user's session affinity switches to this replacement clone as well. After the initial read from the database, the replacement clone places the user's session object in the in-memory cache (assuming the cache has space available for additional entries).

At the time of writing, the WebSphere Web server plug-in will failover to a clone that is randomly selected from the list of available clones. From then onwards, requests for that session will go to the selected clone. The requests for the session will go back to the failed clone when the failed clone comes back up.

WebSphere provides session affinity on a best-effort basis. There are narrow windows where session affinity will fail. These windows are:

1. When a clone is recovering from a crash, there exists a window where concurrent requests for the same session could end up in different clones. This is because the Web server is multi-processed and each process separately maintains its own retry timer value and list of available clones. The end result is that requests being processed by different processes may end up being sent to more than one clone after at least one process has determined that the failed clone is up.

   To avoid or limit exposure due to this scenario, if your clones are expected to crash very seldom and are expected to recover fairly quickly, consider setting the retry timeout to a small value. This will narrow the window during which multiple requests being handled by different processes get routed to multiple clones.

2. A server overload occurring that can cause requests belonging to the same session to go to different clones. This can occur even if all the clones are up and running. For each clone, there is a backlog queue where an entry is made for each request sent by the plug-in waiting to be picked up by a worker thread in the servlet engine. If the depth of this queue is exceeded the Web server plug-in will start receiving responses that the clone is not available. This failure will be handled in the same way by the plug-in as an actual JVM crash.

   Examples of when this can happen are:

   – If the servlet engine has not been set up with an appropriate number of threads to handle the user load.

– If the servlet engine threads are taking a long time to process the requests for various reasons, such as, applications taking a long time to execute, resources being used by applications taking a long time, and so on.

This can be prevented by tuning the system properly. See Chapter 22, “Monitoring and tuning your runtime environment” on page 839.

Note: Session affinity enhancements are currently under consideration for the next WebSphere V4.0 service pack. When a clone is marked dead, the plug-in currently directs the request to one of the available clones randomly. In this case, there is a chance that two concurrent requests for a session might end up in different clones. To avoid this, use of a non-random algorithm is being considered, such as failover to the next clone in list (“clone + 1”), or to a “buddy server” set in the plug-in configuration file.

15.7 Persistent session management

By default, WebSphere places session objects in memory. However, the administrator has the option of enabling persistent session management, which instructs WebSphere to place session objects in a database.

Administrators should enable persistent session management when:

- The user’s session data must be recovered by another clone after a clone in a server groups fails or is shut down.
- The user’s session data is too valuable to lose through unexpected failure at the WebSphere node.
- The administrator desires better control of the session cache memory footprint. By sending cache overflow to a persistent session database, the administrator controls the number of sessions allowed in memory at any given time.

All information stored in a persistent session database must be serialized. As a result all of the objects held by a session must implement java.io.Serializable if the session needs to be stored in a persistent session database.

In general, consider making all objects held by a session serialized, even if immediate plans do not call for the use of persistent session management. If the Web site grows, and persistent session management becomes necessary, the transition between local and persistent management occurs transparently to the application if the sessions only hold serialized objects. If not, a switch to persistent session management requires coding changes to make the session contents serialized.

Persistent session management does not impact the session API, and Web applications require no API changes to support persistent session management. However, as mentioned above, applications storing un-serializable objects in their sessions require modification before switching to persistent session management.

Using multi-row sessions becomes important if the size of the session object exceeds the size for a row, as permitted by the WebSphere session manager. If the administrator requests multi-row session support, the WebSphere session manager breaks the session data across multiple rows as needed. This allows WebSphere to support large session objects. Also, this provides a more efficient mechanism for storing and retrieving session contents under certain circumstances. See 15.7.7, “Support for single or multi-row schemas” on page 543 for information on this feature.

Using a cache lets the session manager maintain a cache of most recently used sessions in memory. Retrieving a user session from the cache eliminates a more expensive retrieval from the persistent database. The session manager uses a “least recently used” scheme for removing objects from the cache. Session data is stored to the persistent store based on the write frequency and write option selected. The details of storing sessions in persistent store are discussed in 15.7.1, “Enable/disable persistent sessions” on page 533.

15.7.1 Enable/disable persistent sessions

Note: Before enabling persistent sessions, make sure that you have created a data source for the database in which you are going to store the session data. The persistent session database performs best if it is not shared with other databases, such as the WebSphere administrative database. This eliminates contention for resources, such as connections, which impacts performance.

See 16.1, “JDBC providers” on page 564 for details on creating data sources.

In order to enable persistent session management:

1. Select the **Persistence** tab in the Session Manager Service window.
2. Click **Enable persistent sessions**, as shown on Figure 15-8, then:
   a. Select from one of the predefined policies for session persistence.

   OR

   b. Manually configure the session persistence. For more information on this option refer to 15.7.2, “Manual tuning of persistent sessions” on page 535.

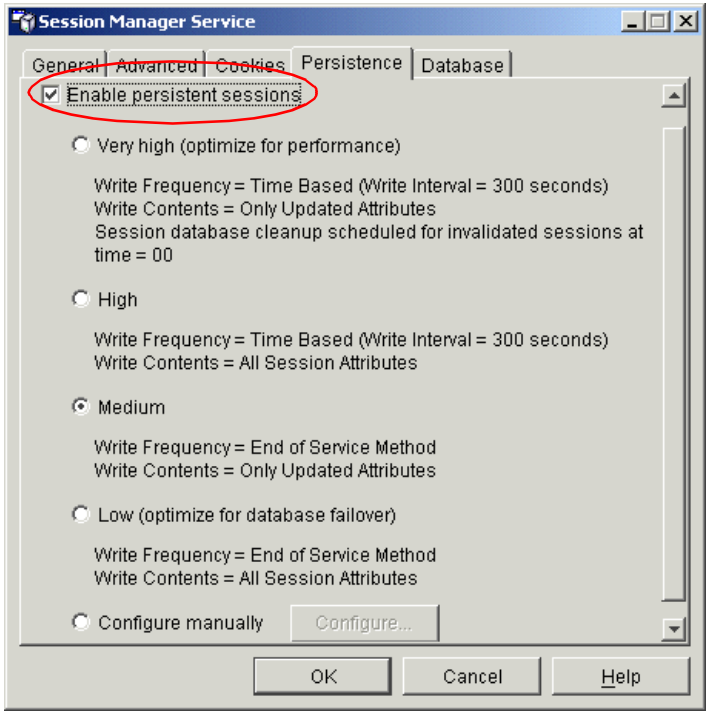


Figure 15-8 Session manager service - persistence properties

3. Select the **Database** tab to display the database properties window shown in Figure 15-9.
   a. Select the data source for the session database from the Data source pull-down. The data source selected must be non-JTA enabled data source.
   b. If required, set the user name and password.
   c. If you are using DB2 and you anticipate requiring row sizes greater than 4 KB, select the appropriate value from the DB2 row size pull-down. See 15.9.1, “Using larger DB2 page sizes” on page 553 for more information.
   d. If you intended using a multi-row schema, click **Use multirow sessions**. See 15.7.7, “Support for single or multi-row schemas” on page 543 for more information.

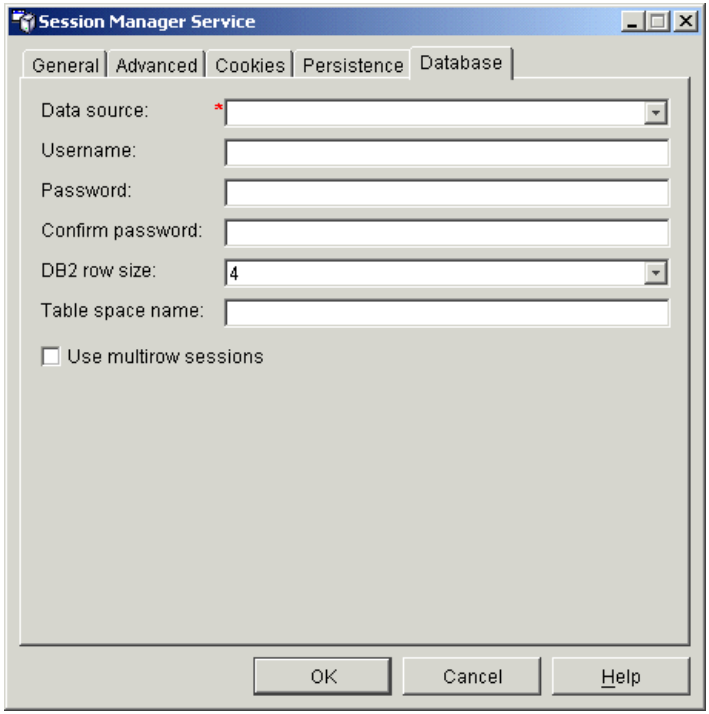


Figure 15-9 Session manager service - database properties

4. Click **OK**.
5. Click **Apply** to save the application server changes.
6. Restart the application server or server group to make these changes take effect.

15.7.2 Manual tuning of persistent sessions

To manually tune persistent session management:

1. Select the **Persistence** tab in the Session Manager Service window.
2. Click **Enable persistent sessions**.
3. Click the **Configure manually** option.
4. Click the **Configure...** button, as shown in Figure 15-10.

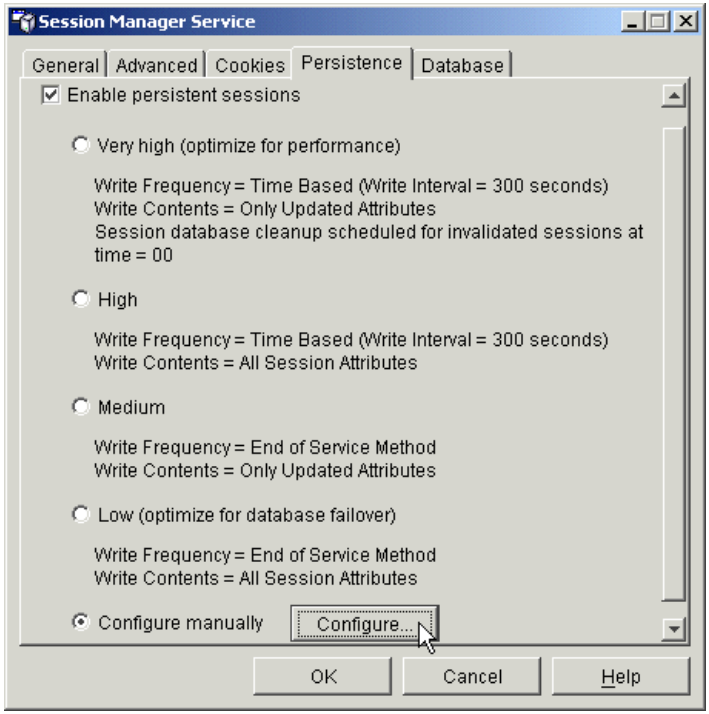


Figure 15-10 Session manager service - configuring persistence properties manually

This opens the Configure Persistence Tuning window, shown in Figure 15-11, which allows the administrator to manually tune:

- How often the session data is written to the database.
- How much data is written.
- When the invalid sessions are cleaned up in the database.

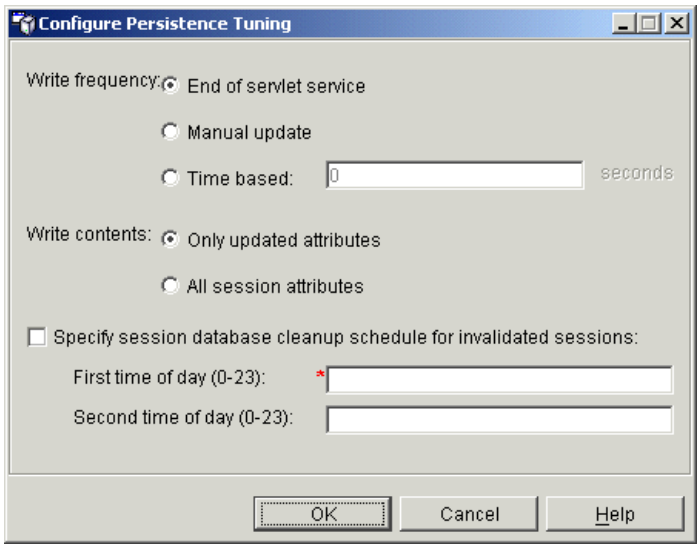


Figure 15-11 Configure persistence tuning

The persistence tuning properties are:

- Write frequency

  This group of options defines how often the HTTP session data is written to the persistent session database.

  - End of servlet service

    If the session data has changed, it will be written to the database after the servlet finishes processing an HTTP request. For more information, see 15.7.3, “Update session database at end of servlet service” on page 538.

  - Manual update

    The session data will be written to the database when the sync() method is called on the IBMSession object. For more information, see 15.7.4, “Manual update of the session database” on page 539.

  - Time based

    This option is new in WebSphere V4.0. If you select this option, session data will be written to the database based on the specified write interval value. For more information, see 15.7.5, “Time based writes to the session database” on page 540.

- Write contents

  These options control what is written. Please refer to 15.7.8, “What is written to the persistent session database” on page 545 before selecting one of the options, since there are several factors to take into account. The options available are:

  – Only update attributes

    Only updated attributes are written to the database.

  – All session attributes

    All attribute are written to the database.

- Specify session database cleanup schedule for invalidated sessions

  WebSphere V4.0 allows the administrator to defer the clearing of invalidated sessions (sessions that are no longer in use and timed out) from the database to the off hours. For more information, see 15.8, “Invalidating sessions” on page 549. This can be done either once or twice a day. The fields available are:

  – First time of day (0-23)

    The first hour during which the invalidated persistent sessions will be cleared from the database. This value must be a positive integer between 0 and 23.

  – Second time of day (0-23)

    The second hour during which the invalidated persistent sessions will be cleared from the database. This value must be a positive integer between 0 and 23.

  Also consider using schedule invalidation for intranet style applications that have a somewhat fixed number of users wanting the same HTTP session for the whole business day.

15.7.3 Update session database at end of servlet service

When the write frequency is set to the end of servlet service option, WebSphere writes the session data to the session database at the completion of the HttpServlet.service() method call. Exactly what is written depends on another setting. See 15.7.8, “What is written to the persistent session database” on page 545 for more information.

Note: The last access time attribute is always updated each time the session is accessed by the servlet or JSP. This done to make sure the session does not time out.

If you choose the **End of servlet service** option, each servlet or JSP access will result in a corresponding database update of the last access time. If you select the **Manual update** option, discussed in 15.7.4, “Manual update of the session database” on page 539, the update of the last access time in database occurs on sync() call or at later time.

See “Reduce persistent sessions database I/O” on page 557 for options available to change this database update behavior.

15.7.4 Manual update of the session database

WebSphere has a manual session writing mode that is enabled using the Manual update write frequency option. This mode allows the application to decide when a session should be stored persistently. In manual mode, the session manager only sends changes to the persistent data store if the application explicitly requests a save of the session information.

Note: Manual updates use an IBM extension to HttpSession that is not part of the Servlet 2.2 API.

Manual mode requires that an application developer use the IBMSession class for managing sessions. When the application invokes the sync() method, the session manager writes the modified session data and last access time to the persistent session database. The session data that is written out to the database is controlled by the write contents option selected.

If the servlet or JSP terminates without invoking the sync() method, the session manager saves the contents of the session object into the session cache (if caching is enabled), but does not update the modified session data in the session database. The session manager will only update the last access time in the database asynchronously, at later time.

Example 15-7 shows how the IBMSession class can be used to manually update the session database.

This interface gives the Web application developer additional control of when (and if) session objects go to the persistent data store. If the application does not invoke the sync() method, and manual update mode is specified, the session updates goes only to the local session cache, not the persistent data store. Web developers use this interface to reduce unnecessary writes to the session database, and thereby to improve overall application performance.

All servlets in the Web application server must perform their own session management in manual mode.

Example 15-7 Using IBMSession for manual update of the session database

```
public void service (HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
{
   // Use the IBMSession to hold the session information
   // We need the IBMSession object because it has the manual update
   // method sync()
   com.ibm.websphere.servlet.session.IBMSession session =
      (com.ibm.websphere.servlet.session.IBMSession)req.getSession(true);

   Integer value = 1;

   //Update the in-memory session stored in the cache
   session.putValue("MyManualCount.COUNTER", value);

   //The servlet saves the session to the database
   session.sync();
}
```

15.7.5 Time based writes to the session database

This new feature of WebSphere V4.0 is selected using the time based write frequency option. Time based write will write session data to the database every x seconds. The value of x is called the write interval.

Rationale for time base writes

The reasons for implementing time based write lies in the changes introduced with the Servlet 2.2 API. The Servlet 2.2 specification introduced two key concepts:

- It limits the scope of a session to a single Web application.
- It both explicitly prohibits concurrent access to an HttpSession from separate Web applications but allows for concurrent access within a given JVM.

Because of these changes, WebSphere provides the session affinity mechanism that assures us that an HTTP request is routed to the Web application handling its HttpSession. This assurance still holds in a WLM environment when using persistent HttpSessions. This means that the necessity to immediately write the session data to the database can now be relaxed somewhat in these environments (as well as non-clustered environments) since the database is now really only used for "failover" and "session cache full" scenarios.

With this in mind, it is now possible to gain potential performance improvements by reducing the frequency of database writes.

Using this mode, WebSphere can significantly reduce the I/O to the database caused by the last access time updates. The last access time attribute of the HTTP session is updated every time the servlet or JSP accesses the session. The servlet or JSP does not have to update the session, just access it. When persistent session management is enabled, the changed last access time will be written to the database. If write at “End of servlet service” mode is enabled, then WebSphere could be writing to the session database every time it process an HTTP request. This can be a significant overhead. By using the time based mode, these updates to the database would be deferred and done in a single transaction. Only the latest change to the last access time of the session will be written. This can significantly reduce the database I/O and CPU requirements. See “Reduce persistent sessions database I/O” on page 557 for more information.

Note: Time based writes requires session affinity for session data integrity.

Comparison between write frequency modes

Let’s consider an example where the Web browser accesses the application once every 5 seconds:

- In "End of servlet service” mode, the session would get written out every 5 seconds.
- In "Manual update" mode, the session gets written out whenever the servlet issues IBMSession.sync(). It is the responsibility of the servlet writer to use the IBMSession interface instead of the HttpSession Interface and the servlets/JSPs must be updated to issue the sync().
- In "Time based" mode, the servlet or JSP need not use the IBMSession class nor issue IBMSession.sync(). If the Write Interval is set to 120 seconds, then the session data gets written out at most every 120 seconds.

Time based write details

The following details apply to time based write:

- The expiration of the write interval does not necessitate a write to the database unless the session has been touched (that is, getAttribute/setAttribute/removeAttribute was called) since the last write.
- If a session write interval has expired and the session has only been retrieved (that is, request.getSession() was called since the last write) then the last access time will be written to the database regardless of the “Write contents” setting.
- If a session write interval has expired and the session properties have been either accessed or modified since the last write then the session properties will be written out in addition to the last access time. Which session properties get written out is dependent on the “Write contents” settings.
- Time based write allows the servlet or JSP to issue IBMSession.sync() to force the write of session data to the database.
- If the time between session servlet requests (for a particular session) is greater than the write interval then the session effectively gets written out after each service method invocation.
- The session cache should be large enough to hold all of the active sessions. Failure to do this will result in extra database writes since the receipt of a new session request may result in writing out the oldest cached session to the database. Or to put it another way, if the session manager has to remove the least recently used HttpSession from the cache during a “full cache” scenario, the session manager will write out that HttpSession (per the Write contents settings) upon removal from the cache.
- The session invalidation time must be at least twice the write interval to ensure that a session does not inadvertently get invalidated prior to getting written to the database.
- A newly created session will always get written to the database at the end of the service method.

15.7.6 Persistent sessions and non-serializable J2EE objects

In order for the WebSphere session manager to persist a session to the database, all of the Java objects in an HttpSession must be serializable (that is, implement the java.io.Serializable interface). The HttpSession can also contain the J2EE objects which are not serializable:

- javax.ejb.EJBObject
- javax.ejb.EJBHome
- javax.naming.Context
- javax.transaction.UserTransaction

The WebSphere session manager works around the problem of serializing these objects in the following manner:

- EJBObject and EJBHome each have Handle and HomeHandle object attributes that are serializable and can be used to reconstruct the EJBObject and EJBHome.
- Context is constructed with a hash table based environment, which is serializable. WebSphere will retrieve the environment, then wrapper it with an internal, serializable object, so that on re-entry it can check the object type and reconstruct the Context.
- UserTransaction has no serializable attributes. WebSphere provides two options:

  a. The Web developer can place the object in the HttpSession but WebSphere will not persist it outside the JVM.

  b. WebSphere has a new public wrapper object, com.ibm.websphere.servlet.session.UserTransactionWrapper, which is serializable and requires the InitialContext used to construct the UserTransaction. This will be persisted outside the JVM and be used to reconstruct the UserTransaction.

Note: As per J2EE, a Web component may only start a transaction in a service method. A transaction that is started by a servlet or JSP must be completed before the service method returns. That is, transactions may not span Web requests from a client. If there is an active transaction after returning from the service method, WebSphere will detect it and will abort the transaction.

In general, Web developers should consider making all other Java objects held by HttpSession serializable, even if immediate plans do not call for the use of persistent session management. If the Web site grows, and persistent session management becomes necessary, the transition between local and persistent management occurs transparently to the application if the sessions hold only serializable objects. If not, a switch to persistent session management requires coding changes to make the session contents serializable.

15.7.7 Support for single or multi-row schemas

When using the single-row schema, each user session maps to a single database row. This is WebSphere's default configuration for persistent session management. With this setup, there are hard limits to the amount of user-defined, application-specific data that WebSphere Application Server can access.

When using the multi-row schema, each user session maps to multiple database rows. In a multi-row schema, each session attribute maps to a database row.

In addition to allowing larger session records, using a multi-row schema can yield performance benefits, as discussed in “Multi-row persistent session management” on page 559.

It should be stressed that switching between multi-row and single-row is not a trivial proposition.

Switching from single-row to multi-row schema

To switch from single-row to multi-row schema for sessions:

- Modify the session manager properties to switch from single to multi-row schema. You need to check the **Use multirow sessions** box on the Database tab of the Session Manager Service window, shown in Figure 15-9 on page 535.
- Manually drop the database table or delete all the rows in the database table that the product uses to maintain HttpSession objects.

  To drop the table:

  - Determine which data source the session manager is using.
  - In the data source properties, look up the database name.
  - Use the database facilities to connect to the database.
  - Drop the SESSIONS table.
- Restart the application server or server group.

Design considerations

Consider configuring direct single-row usage to one database and multi-row usage to another database while you verify which option suits your application's specific needs. You can do this by switching the data source used, then monitor performance.

Table 15-1 Single vs. multi-row schemas

Programming issue	Application scenario
Reasons to use single-row	► You can read/write all values with just one record read/write. ► This takes up less space in a database, because you are guaranteed that each session is only one record long.

Programming issue	Application scenario
Reasons *not* to use single-row	2 MB limit of stored data per session. That is, the sum of sizes of all session attributes is limited to 2 MB.
Reasons to use multi-row	► The application can store an unlimited amount of data; that is, you are limited only by the size of the database and a 2 MB-per-record limit (so the size of each session attribute can be 2 MB). ► The application can read individual fields instead of the whole record. When large amounts of data are stored in the session but only small amounts are specifically accessed during a given servlet's processing of an HTTP request, multi-row sessions can improve performance by avoiding unneeded Java object serialization.
Reasons *not* to use multi-row	If data is small in size, you probably do not want the extra overhead of multiple row reads when everything could be stored in one row.

In the case of multi-row usage, design your application data objects so that they do not to have references to each other. This is to prevent circular references. For example, suppose you are storing two objects A and B in the session using HttpSession.put(..), and A contains a reference to B. In the multi-row case, because objects are stored in different rows of the database, when objects A and B are retrieved later, the object graph between A and B is different from stored. A and B behave as independent objects.

15.7.8 What is written to the persistent session database

As we saw in Figure 15-11 on page 537, WebSphere V4.0 supports two modes for writing persistent session contents to the database.

- **Only updated attributes**. This was the only mode available before WebSphere V4.0. It writes only the HttpSession properties that have been updated via setAttribute() and removeAttribute().
- **All session attributes**. This mode was added in WebSphere V4.0. It writes all the HttpSession properties to the database.

When a new session is initially created (with either of the above two options) the entire session is written to the database including any Java objects bound to the session. The behavior for subsequent servlet or JSP requests for this session varies depending on whether the single-row or multi-row database mode is in use.

- In single-row mode:
  - Only updated attributes: If any session attribute has been updated (via setAttribute or removeAttribute), then all of the objects bound to the session will be written to the database.
  - All session attributes: All bound session attributes will be written to the database.
- In multi-row mode:
  - Only updated attributes: Only the session attributes that were specified via setAttribute or removeAttribute will be written to the database.
  - All session attributes: All of the session attributes that reside in the cache will be written to the database. If the session has never left the cache, then this should contain all of the session attributes.

By using the “All session attributes” mode the servlet and JSP can change Java objects that are attributes of the HttpSession without having to call setAttribute() on the HttpSession for that Java object in order for the changes to be reflected in the database.

Adding the “All session attributes” mode provides some flexibility to the application programmer and protects against possible side effects of moving from local sessions to persistent sessions.

However, using “All session attributes” mode can potentially increase database activity and be a performance drain. Individual customers will have to evaluate the pros and cons for their installation. It should be noted that the combination of “All session attributes” mode with “Time based” write could greatly reduce the performance penalty and essentially give you the best of both worlds.

As shown in Example 15-8 and Example 15-9, the initial session creation contains a setAttribute but subsequent requests for that session do not need to use setAttribute.

Example 15-8 Initial servlet

```
HttpSession sess = request.getSession(true);
myClass myObject = new myClass();
myObject.someInt = 1;
sess.setAttribute("myObject", myObject);   // Bind object to the session
```

Example 15-9 Subsequent servlet

```
HttpSession sess = request.getSession(false);
myObject =  sess.getAttribute("myObject");   // get bound session object
myObject.someInt++;  // change the session object
// setAttribute() not needed with write "All session attributes" specified
```

Example 15-10 and Example 15-11 show the case where setAttribute is still required even though the write "All session attributes" option is enabled.

Example 15-10 Initial servlet

```
HttpSession sess = request.getSession(true);
String myString = new String("Initial Binding of Session Object");
sess.setAttribute("myString", myString);   // Bind object to the session
```

Example 15-11 Subsequent servlet

```
HttpSession sess = request.getSession(false);
String myString  = sess.getAttribute("myString"); // get bound session object
...
myString = new String("A totally new String");  // get a new String object
sess.setAttribute("myString", myString);   // Need to bind the object to the
session since a NEW Object is used
```

HttpSession set/getAttribute action summary

Table 15-2 summarizes the action of the HttpSession setAttribute and removeAttribute methods for various combinations of the row type, write contents, and write frequency session persistence options.

Table 15-2 Write contents vs. write frequency

Row type	Write contents	Write frequency	Action for setAttribute	Action for remove-Attribute
Single-row	Only updated attributes	End of servlet service / sync() call with Manual update	If any of the session data has changed then write all of this session's data from cache[1]	If any of the session data has changed then write all of this session's data from cache[1]

Row type	Write contents	Write frequency	Action for setAttribute	Action for remove-Attribute
Single-row	Only updated attributes	Time based	If any of the session data has changed then write all of this session's data from cache[1]	If any of the session data has changed then write all of this session's data from cache[1]
Single-row	All session attributes	End of servlet service / sync() call with Manual update	Always write all of this session's data from cache[2]	Always write all of this session's data from cache[2]
Single-row	All session attributes	Time based	Always write all of this session's data from cache	Always write all of this session's data from cache
Multi-row	Only updated attributes	End of servlet service / sync() call with Manual update	Write only thread specific data that has changed	Delete only thread specific data that has been removed
Multi-row	Only updated attributes	Time based	Write thread specific data that has changed for *all* threads using this session	Delete thread specific data that has been removed for *all* threads using this session
Multi-row	All session attributes	End of servlet service / sync() call with Manual update	Write all session data from cache	Delete thread specific data that has been removed for *all* threads using this session
Multi-row	All session attributes	Time based	Write all session data from cache	Delete thread specific data that has been removed for *all* threads using this session

Notes:

1. When a session is written to the database while using single-row mode, *all* of the session data is written. Therefore, no database deletes are necessary for properties that were removed via removeAttribute() since the write of the entire session will not include removed properties.
2. Multi-row mode has the notion of thread-specific data. Thread-specific data is defined as session data that was added or removed while executing under this thread. If "End of servlet service" or "Manual update" modes are used and "Only updated attributes" is enabled, then only the thread-specific data is written to the database.

15.8 Invalidating sessions

If the user no longer needs the session object because they went through the logoff process for the site, for example, the session becomes an excellent candidate for invalidation. Invalidating a session removes it from the session cache, as well as from the session database.

WebSphere offers three methods for invalidation session objects:

- Programmatically, by calling the invalidate() method on the session object. If the session object is accessed by multiple threads in a Web application, take care that none of the threads still have references to the session object.
- An invalidator thread scans for timed out sessions every n seconds where n is configurable from the administration console. Session invalidation timeout is set in the Session Manager Service window, Advanced tab, as seen in Figure 15-5 on page 527.
- For persistent sessions only, the administrator can specify times when the scan will be run. The session database cleanup schedule for invalidated sessions is set in the Configure Persistence Tuning window, as seen in Figure 15-11 on page 537. This feature has the following benefits when used with persistent session:
  - Database scans can be scheduled during periods that normally have low demand. This avoids slowing down online applications due to contention in the persistent session database.
  - When this setting is used with the "End of servlet service" write frequency option, WebSphere does not have to write out the last access time with every HTTP request. This is because WebSphere does not have to synchronize the invalidator thread's deletion with the HTTP request access.

Notes:

1. HttpSession timeouts are not enforced. Instead, all invalidation processing is handled at the configured invalidation times.
2. HttpSessionBindingListener processing is potentially delayed by this this configuration. It is not recommended if listeners are used.

15.8.1 Session listeners

As per the Servlet 2.2 specification, before terminating a session WebSphere notifies session attributes implementing the following interface:

```
javax.servlet.http.HttpSessionBindingListener
```

The unbound event will be called on the session attribute. Sessions time out (even in persistent sessions), so this interface allows the application to manage session information before the session timeout completes and the session data is destroyed. In a cluster environment with persistent sessions, if the session times out, this unbound event may be fired in any one of the application servers in the cluster.

15.9 Session security

WebSphere Application Server maintains the security of individual sessions.

When session manager integration with WebSphere security is enabled, the session manager checks the user ID of the HTTP request against the user ID of the session held within WebSphere. This check is done as part of the processing of the request.getSession() function. If the check fails, WebSphere throws an com.ibm.websphere.servlet.session.UnauthorizedSessionRequestException exception. If it succeeds the session data is returned to the calling servlet or JSP.

Session security checking works with the standard HttpSession. The identity or user name of a session can be accessed via the com.ibm.websphere.servlet.session.IBMSession interface. An unauthenticated identity is denoted by the user name "anonymous".

The session manager uses WebSphere's security infrastructure to determine the authenticated identity associated with a client HTTP request that either retrieves or creates a session. WebSphere security determines identity using certificates, LPTA, and other methods. See Chapter 7, "Security" on page 103 for more information on WebSphere security.

Security integration rules for HTTP sessions

Session management security has the following rules:

- Sessions in unsecured pages are treated as accesses by the "anonymous" user.
- Sessions created in unsecured pages are created under the identity of that "anonymous" user.
- Sessions in secured pages are treated as accesses by the authenticated user.
- Sessions created in secured pages are created under the identity of the authenticated user. They can only be accessed in other secured pages by the same user. To protect these sessions from use by unauthorized users, they cannot be accessed from an insecure page, that is, do not mix access to secure and insecure pages.
- Security integration in session manager is not supported in HTTP form-based login unless LPTA is used.

Table 15-3 lists possible scenarios when security integration is enabled, where outcomes depend on whether the HTTP request was authenticated and whether a valid session ID and user name was passed to the session manager.

Table 15-3 HTTP session security

Request session ID / user name	**Unauthenticated HTTP request is used to retrieve the session**	**Authenticated HTTP request is used to retrieve the session. The user ID in the HTTP request is "FRED".**
No session ID was passed in for this request, or the ID is for a session that is no longer valid.	A new session is created. The user name is "anonymous".	A new session is created. The user name is "FRED".
A valid session ID is received. The current session user name is "anonymous".	The session is returned.	The session is returned. The session manager changes the user name to "FRED".
A valid session ID is received. The current session user name is "FRED".	The session is not returned. UnauthorizedSession-RequestException is thrown[1].	The session is returned.

Request session ID / user name	Unauthenticated HTTP request is used to retrieve the session	Authenticated HTTP request is used to retrieve the session. The user ID in the HTTP request is "FRED".
A valid session ID is received. The current session user name is "BOB".	The session is not returned. UnauthorizedSession-RequestException is thrown[1].	The session is not returned. UnauthorizedSession-RequestException is thrown[1].

Notes:

1. com.ibm.websphere.servlet.session.UnauthorizedSessionRequestException is thrown to the servlet or JSP.

Enabling session security

The administrator can enable session manager integration with WebSphere security as followings:

1. Open the **Advanced** tab on the Session Manager Service window, as shown in Figure 15-12.
2. Check the **Integrate with WebSphere security** option.
3. Click **OK**.
4. Click **Apply** to save the application server changes.
5. Restart the application server or server group to make these changes take effect.

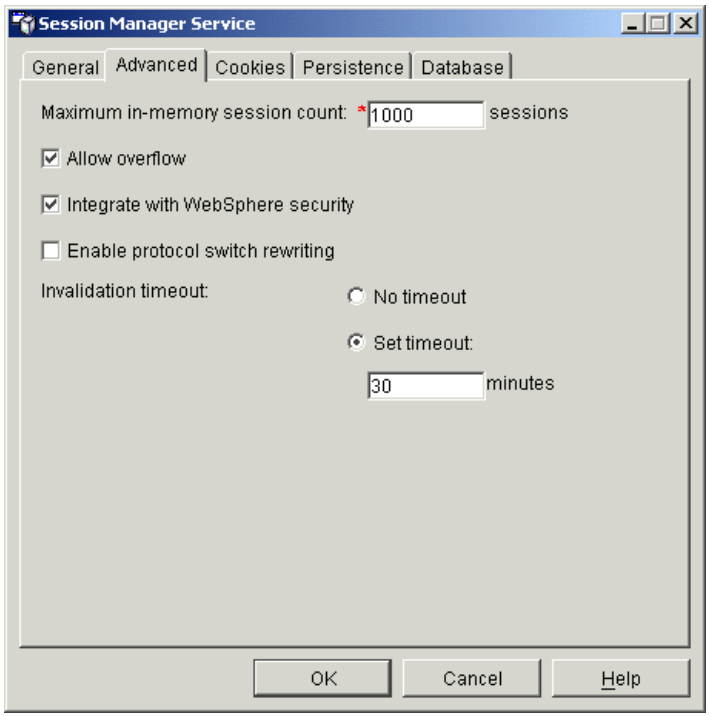


Figure 15-12 Session manager service - Integrate with WebSphere security option

15.9.1 Using larger DB2 page sizes

Previously, only the 4 KB DB2 page size was supported. The fastest binary column, varchar for bit data, used to store HttpSession attributes is only a little over 3 KB for this page size. WebSphere V4.0 now supports 4 KB, 8 KB, 16 KB, and 32 KB page sizes, and hence can have larger varchar for bit data columns of ~7 KB, 15 KB, or 31 KB. Using this performance feature, we see faster persistence for HttpSession of sizes of 7 KB to 31 KB in the single-row case, or attribute sizes of 4 KB to 31 KB in the multi-row case.

To enable this feature involves dropping any existing table created with a 4 KB buffer pool and table space. This also applies if you subsequently change between 4 KB, 8 KB, 16 KB, or 32 KB.

To use a page size other than the default (4 KB), do the following:

1. If the SESSIONS table already exists, drop it from the DB2 database:

```
DB2 connect to session
DB2 drop table sessions
```

2. Create a new DB2 buffer pool and tablespace, specifying the same page size (8 KB, 16 KB or 32 KB) for both, and assign the new buffer pool to this tablespace. Following are simple steps for creating an 8 KB page:

```
DB2 connect to session
DB2 CREATE BUFFERPOOL sessionBP SIZE 1000 PAGESIZE 8K
DB2 connect reset
DB2 connect to session
DB2 CREATE TABLESPACE sessionTS PAGESIZE 8K MANAGED BY SYSTEM USING
('D:\DB2\NODE0000\SQL00005\sessionTS.0') BUFFERPOOL sessionBP
DB2 connect reset
```

   Refer to DB2 product documentation for details.

3. Configure the correct tablespace name and page size in the Session Manager Service window, Database tab. As seen in Figure 15-9 on page 535, page size is referred to as “DB2 row size” in the session manager database properties.
4. Restart WebSphere. On startup, the session manager creates a new SESSIONS table based on the page size and table space name specified.

15.10 Session performance considerations

This section includes guidance for developing and administering scalable, high-performance Web applications using WebSphere Application Server session support.

15.10.1 Session size

Large session objects pose several problems for a Web application. If the site uses session caching, large sessions reduce the memory available in the WebSphere instance for other tasks, such as application execution.

For example, assume a given application stores 1 MB of information per user session object. If 100 users arrive over the course of 30 minutes, and assume the session timeout remains at 30 minutes, the application server instance must allocate 100 MB just to accommodate the newly arrived users in the session cache. Note this number does not include previously allocated sessions that have not timed out yet. The actual memory required by the session cache could be considerably higher than 100 MB.

1 MB per user session * 100 users = 100 MB

Web developers and administrators have several options for improving the performance of session management:

- Reduce the size of the session object
- Reduce the size of the session cache
- Add additional instances
- Invalidate unneeded sessions
- Increase the memory available
- Reduce the session timeout interval

Reduce session object size

Web developers must carefully consider the information kept by the session object:

- Removing information easily obtained or easily derived helps keep the session object small.
- Rigorous removal of unnecessary, unneeded, or obsolete data from the session.
- Consider whether it would be better to keep a certain piece of data in an application database rather than in the HTTP session. This gives the developer full control over when the data is fetched or stored and how it is combined with other application data. Web developers can leverage the power of SQL if the data is in an application database.

This becomes particularly important when persistent sessions are used. There is a significant performance overhead when WebSphere has to serialize a large amount of data and write it to the database. Even if the Write contents option is enabled, if the session object contains large Java objects or collections of objects that are updated regularly, there is a significant performance penalty in persisting these objects. This penalty can be reduced by using time base writes; see 15.7.5, “Time based writes to the session database” on page 540.

Notes: In general the best performance will be realized with session objects that are less than 2 KB in size. Once the session object starts to exceed 4-5 KB in size, a significant decrease in performance can be expected.

Even if session persistence is not an issue, minimizing the session object size will help to protect your Web application from scale-up disasters as user numbers increase. Large session objects will require more and more JVM memory, leaving no room to run servlets.

See also 15.9.1, “Using larger DB2 page sizes” on page 553 on how WebSphere V4.0 can provide faster persistence of larger session objects.

Session cache size

As discussed earlier in this chapter, the session manager allows administrators to change the session cache size to alter the cache's memory footprint. By default, the session cache holds 1000 session objects. By lowering the number of session objects in the cache, the administrator reduces the memory required by the cache.

However, if the user's session is not in the cache, WebSphere must retrieve it from either the overflow cache (for local caching) or the session database (for persistent sessions). If the session manager must retrieve persistent sessions frequently, the retrievals may impact overall application performance.

WebSphere maintains overflowed local sessions in memory, as discussed in 15.4, "Local sessions" on page 524. Local session management with cache overflow enabled allows an "unlimited" number of sessions in memory. In order to limit the cache footprint to the number of entries specified in session manager, the administrator should use persistent session management, or disable overflow.

Note: When using local session management without specifying the Allow overflow property, a full cache will result in the loss of user session objects.

Create additional application server clones

WebSphere also gives the administrator the option of creating additional application server instances, or clones. Creating additional clones spreads the demand for memory across more JVMs, thus reducing the memory burden on any particular instance. Depending on the memory and CPU capacity of the machines involved, the administrator may add additional instances within the same machine. Alternatively, the administrator may add additional machines to form a hardware cluster, and spread the instances across this cluster.

Note: When configuring a session cluster, session affinity routing provides the most efficient strategy for user distribution within the cluster, even with session persistence enabled. With clones, the WebSphere HTTP plug-in provides affinity routing among clone instances.

Invalidate unneeded sessions

If the user no longer needs the session object because they went through the logoff process for the site, for example, the session becomes an excellent candidate for invalidation. Invalidating a session removes it from the session cache, as well as from the session database. For more information see 15.8, "Invalidating sessions" on page 549.

Increase available memory

WebSphere allows the administrator to increase an application server's heap size. By default, WebSphere allocates 256 MB as the maximum heap size. Increasing this value allows the instance to obtain more memory from the system, and thus hold a larger session cache.

A practical limit exists, however, for an instance's heap size. The machine memory containing the instance needs to support the heap size requested. Also, if the heap size grows too large, the length of the garbage collection cycle with the JVM may impact overall application performance. This impact has been reduced with the introduction of multi-threaded garbage collection.

Reduce persistent sessions database I/O

Every time that a servlet or JSP accesses an HTTP session the last access time is updated on the session object. This is done to make sure that the session does not time out. The last access time is updated even if the servlet does not call getAttribute() or setAttribute(). It only has to call getSession().

When persistent session management is enabled the change to the last access time is written to the database. See 15.7.8, "What is written to the persistent session database" on page 545 for more on how much data is written to the database. When the "End of servlet service" Write frequency mode is enabled, the session is written at the end of the call to the service() method for the servlet or JSP. This can easily lead to a situation where WebSphere is writing the session to the database every time an HTTP request is processed. This is true even if the servlet or JSP only reads from the session.

For JSPs this is a particular concern. In compliance with the J2EE specification, JSPs access the session object each time they are executed by default. This means that the last access time is changed and written to the database each time WebSphere executes the JSP.

From a performance point of view, the Web developer should consider the following:

- Optimize the use of the HttpSession within a servlet. Only store the minimum amount of data required in HttpSession. Data that does not have to be recovered after a clone fails or is shut down may be best kept elsewhere, such as in a hash table. Recall that HTTP session is intended to be used as a *temporary* store for state information between browser invocations.
- JSPs that do not need to access the session object should use the JSP directive in Example 15-12. This tells the JSP container that you will not be accessing the session. This stops the last accessed time being updated.

By default this directive is set to true. This causes the last access time to be updated and written to the database every time WebSphere executes the JSP.

Example 15-12 Directive to stop a JSP updating the session last accessed time

```
<%@ page session="false"%>
```

- Use “Time based” Write frequency mode. This greatly reduces the amount of I/O as the database updates are deferred up to a configurable number of seconds. Using this mode, all of the outstanding updates for a Web application are written out periodically based on the configured write interval.
- Use the “Specify session database cleanup schedule for invalidated sessions” option. When using the "End of servlet service" Write frequency mode, WebSphere does not have to write out the last access time with every HTTP request. This is because WebSphere does not have to synchronize the invalidator thread's deletion with the HTTP request’s access.

Session timeout interval

By default, each user receives a 30-minute interval between requests before the session manager invalidates the user’s session. Not every site requires a session timeout interval this generous. By reducing this interval to match the requirements of the average site user, the session manager purges the session from the cache (and the persistent store, if enabled) more quickly.

Avoid setting this parameter too low and frustrating users. The administrator must take into account a reasonable time for an average user to interact with the site (read returned data, fill out forms, and so on) when setting the interval. Also, the interval must represent any increased response time during peak times on the site (such as heavy trading days on a brokerage site, for example).

Finally, in some cases where the persistent session database table contains a large number of entries, frequent execution of the timeout scanner reduces overall performance. In cases where the database contains many session entries, avoid setting the session timeout so low it triggers frequent, expensive scans of the persistent session database for timed-out sessions. Alternatively, the administrator should consider schedule based invalidation where scans for invalid object can be deferred to a time that normally has low demand. See 15.8, “Invalidating sessions” on page 549.

Multi-row persistent session management

When a session contains multiple objects accessed by different servlets or JSPs in the same Web application, multi-row session support provides a mechanism for improving performance. Multi-row session support stores session data in the persistent session database by Web application and value. Table 15-4 shows a simplified representation of a multi-row database table.

Table 15-4 Simplified multi-row session representation

Session ID	Web application	Property	Small value	Large value
DA32242SSGE2	ShoeStore	ShoeStore.First.Name	“Alice”	
DA32242SSGE2	ShoeStore	ShoeStore.Last.Name	“Smith”	
DA32242SSGE2	ShoeStore	ShoeStore.Big.String		“A big string....”

In this example, if the user visits the ShoeStore application, and the servlet involved needs the user’s first name, the servlet retrieves this information through the session API. The session manager brings into the session cache only the value requested. The ShoeStore.Big.String item remains in the persistent session database until the servlet requests it. This saves time by reducing both the data retrieved and the serialization overhead for data the application does not use.

After the session manager retrieves the items from the persistent session database, these items remain in the in-memory session cache. The cache accumulates the values from the persistent session database over time as the various servlets within the Web application request them. With WebSphere’s session affinity routing, the user returns to this same cached session instance repeatedly. This reduces the number of reads against the persistent session database, and gives the Web application better performance.

How session data is written to the persistent session database has been made configurable in WebSphere V4.0. For information on session updates using single and multi-row session support see 15.7.7, “Support for single or multi-row schemas” on page 543. Also see 15.7.8, “What is written to the persistent session database” on page 545.

Even with multi-row session support, Web applications perform best if the overall contents of the session objects remain small. Large values in session objects require more time to retrieve from the persistent session database, generate more network traffic in transit, and occupy more space in the session cache after retrieval.

Multi-row session support provides a good compromise for Web applications requiring larger sessions. However, single-row persistent session management remains the best choice for Web applications with small session objects. Single-row persistent session management requires less storage in the database, and requires fewer database interactions to retrieve a session's contents (all of the values in the session are written or read in one operation). This keeps the session object's memory footprint small, as well as reducing the network traffic between WebSphere and the persistent session database.

Note: Avoid circular references within sessions if using multi-row session support. The multi-row session support does not preserve circular references in retrieved sessions.

Managing your session database connection pool

When using persistent session management, the session manager interacts with the defined database through a WebSphere Application Server data source. Each data source controls a set of database connections known as a connection pool. By default, the data source opens a pool of no more than 10 connections. The maximum pool size represents the number of simultaneous accesses to the persistent session database available to the session manager.

For high-volume Web sites, the default settings for the persistent session data source may not be sufficient. If the number of concurrent session database accesses exceeds the connection pool size, the data source queues the excess requests until a connection becomes available. Data source queueing can impact the overall performance of the Web application (sometimes dramatically).

For best performance, the overhead of the connection pool used by the Session Manager needs to be balanced against the time that a client may spend waiting for an in-use connection to become available for use. By definition a connection pool is a *shared* resource, so in general the best performance will typically be realized with a connection pool that has significantly fewer connections than the number of simultaneous users.

A large connection pool does not necessarily improve application performance. Each connection represents memory overhead. A large pool decreases the memory available for WebSphere to execute applications. Also, if database connections are limited because of database licensing issues, the administrator must share a limited number of connections among other Web applications requiring database access as well. This is one area where performance tuning tests will be required to determine the optimal setting for a given application.

As discussed above, session affinity routing combined with session caching reduces database read activity for session persistence. Likewise, “Manual update” write frequency, “Time based” write frequency, and multi-row persistent session management reduce unnecessary writes to the persistent database. Incorporating these techniques may also reduce the size of the connection pool required to support session persistence for a given Web application.

Prepared statement caching is a connection pooling mechanism that can be used to further improve session database response times. A cache of previously prepared statements is available on a connection. When a new prepared statement is requested on a connection, the cached prepared statement is returned, if available. This caching reduces the number of costly prepared statements created, which improves response times.

In general, base the prepared statement cache size on:

- The smaller of:
  - Number of concurrent users
  - Connection pool size
- The number of different prepared statements

With 50 concurrent users, a connection pool size of 10, and each user using 2 statements (a select and an insert) the prepared statement cache size should be at least 10 x 2 = 20 statements. To find out more, see the InfoCenter "Prepared statement cache size" article, in the Tuning section.

Note: The persistent session database performs best if it is not shared with other databases, such as the WebSphere administrative database. This eliminates contention for resources, such as connections, which impacts performance.

15.10.2 Session database tuning

While the session manager implementation in WebSphere V4.0 provides for a number of parameters that can be tuned to improve performance of applications that utilize HTTP sessions, maximizing performance will require tuning the underlying session persistence table. WebSphere provides a "first step" in this regard by creating an index for the sessions table when creating the table. The index is comprised of the session ID, the property ID (for multi-row sessions) and the Web application name.

While most database managers provide a great deal of capability in tuning at the table or tablespace level, creation of a separate database or instance will afford the most flexibility in tuning. Proper tuning of the instance/database can improve performance by 5% (or more) over that which can be achieved by simply tuning the table or tablespace.

While the specifics will vary depending on the database and operating system in use, in general the database should be tuned and configured as appropriate for a database that experiences a great deal of I/O. The DBA should monitor and tune the database buffer pools, database log size and write frequency. Additionally, maximizing performance will require striping the database/instance across multiple disk drives and disk controllers, and utilizing any hardware or OS buffering that is available in order to reduce disk contention. There are any number of excellent references that cover the specifics of database tuning for the database product in use. An excellent reference for DB2 and Oracle is the redbook *Database Performance on AIX in DB2 UDB and Oracle Environments*, SG24-5511.

16

Configuring WebSphere resources

Resource providers are a class of objects that provide resources needed by running Java applications, and J2EE applications in particular. If your application requires data access, you are probably already familiar with JDBC data source providers.

Our discussion of application server resources includes the following resource providers:

- JDBC providers for obtaining relational database data sources
- JavaMail providers for obtaining mail sessions
- URL providers for obtaining URLs
- J2C providers for obtaining Enterprise Information Systems access
- JMS providers for obtaining JMS connection factories and destinations

We conclude with a brief look at configuring application client resources.

16.1 JDBC providers

The JDBC API provides a Java application programming interface to SQL databases. JDBC providers and data sources are used by Java components, such as servlets and enterprise beans, for database-independent connectivity to a variety of data stores.

The WebSphere administrator has an important role in establishing and maintaining database connections through JDBC providers and data sources:

- Configuring the JDBC providers and data sources used in connection pooling.

  JDBC providers and data sources need to be configured for each brand and version of database from which enterprise applications or individual resources will require connections.

- Adjusting connection pooling parameters for optimal performance.

  Connection pools provide a way to share the connection overhead among multiple requests. The optimal value for the pool size and other settings needs to be determined, based on environmental factors such as the operating system in use.

16.1.1 What are JDBC providers and data sources?

Database vendors supply software we call *JDBC providers* that enable Java applications to connect to JDBC-compliant databases through the use of data source objects. WebSphere uses JDBC provider configurations to specify the location of the JDBC data source code for connecting Java applications to a database.

WebSphere uses data source configurations to establish a pool of connections to a database. The data source configuration also specifies connection pooling parameters, such as the maximum number of connections to maintain in the pool. All entity beans require a data source, specified as either a property of the enterprise bean, or as a property of the enterprise bean container.

WebSphere creates a data source instance (and the associated connection pool) for every application server instance that uses the data source. The connection pool associated with a data source is in turn shared by all application components (servlets, JSP files, and enterprise beans) that are running in an application server.

16.1.2 Configuring JDBC providers and data sources

Use the WebSphere Administrative Console to administer JDBC providers and data sources. As shown in Figure 16-1, you can work with JDBC providers and data source resources in the tree view, under **Resources** -> **JDBC Providers** -> provider_name. Expand the tree further to see **Data Sources** -> data_source_name for each provider.

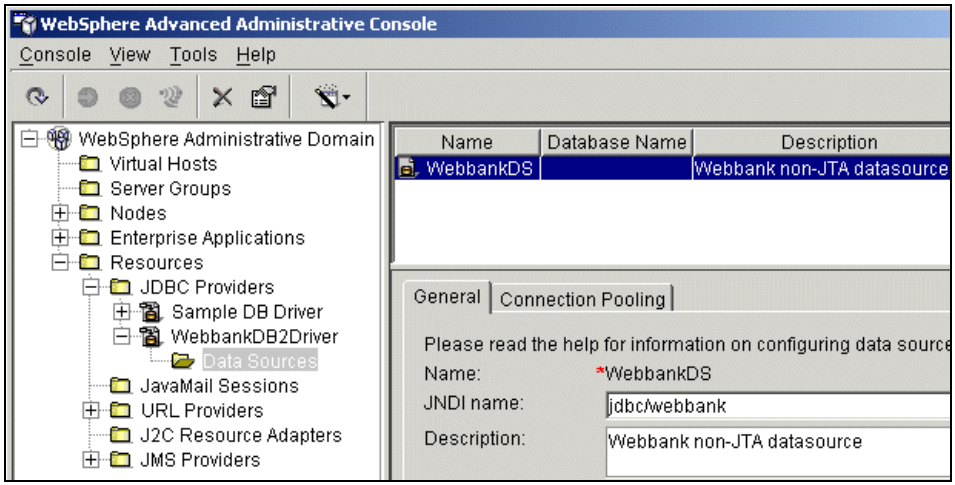


Figure 16-1 Working with JDBC providers and data sources

In this section we look at the properties used to configure JDBC providers and data sources.

Refer to 19.1.3, “Creating a JDBC provider and data source” on page 691 for an example of creating a JDBC provider and data source for the Webbank sample application.

Configuring JDBC providers

To create a new JDBC provider right-click **JDBC Providers** and select **New...** from the pop-up menu. To update an existing JDBC provider, right-click the required provider and select **Properties** from the pop-up menu.

As shown in Figure 16-2, the JDBC provider general properties are:

- Name

  A name for the provider. It is recommended you enter a name that is suggestive of the database product you are using, such as DB2JdbcDriver.

- Description

  A description of the provider, for your administrative records.

- Implementation class

  The name of the Java data source class provided by the database vendor. For example, the one-phase commit protocol classes for DB2 and Oracle are:

  – DB2:

    `COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource`

  – Oracle:

    `oracle.jdbc.pool.OracleConnectionPoolDataSource`

  See the full listing of Java data source classes by database vendor in the InfoCenter.

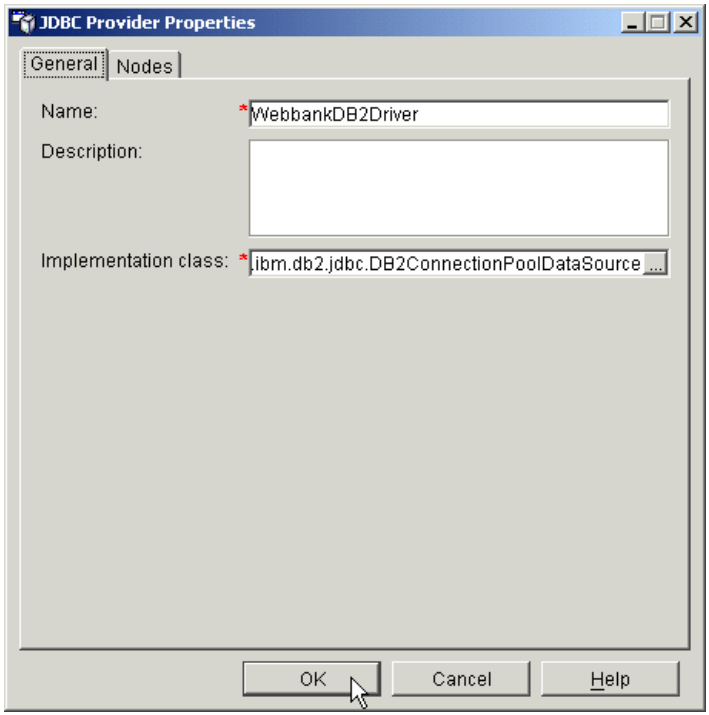


Figure 16-2 JDBC Provider Properties - General tab

You need to install a driver on each node where the JDBC provider is used. The nodes properties for the JDBC providers, seen in Figure 16-3, are:

- Node

  The node (or machine) on which to install the driver.

  Click the **Install New...** and **Uninstall** buttons on the Nodes tab to access windows for installing drivers on specific nodes and for uninstalling drivers.

- Classpath

  The path to the Java file containing the implementation code for the database driver, such as db2java.zip file for DB2.

  See the InfoCenter for details on locating JDBC providers on your operating system.

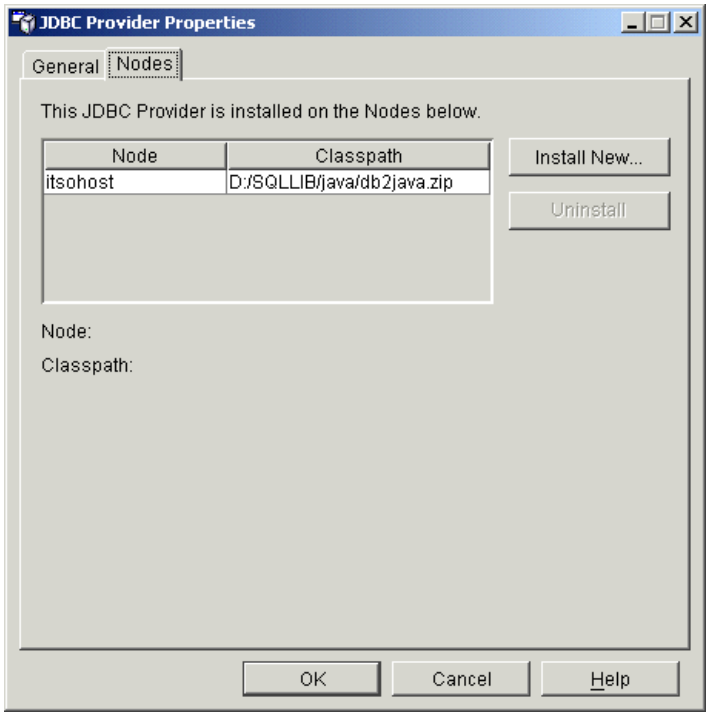


Figure 16-3 JDBC Provider Properties - Nodes tab

Configuring JDBC data sources

To create a new JDBC data source right-click **Data Sources** and select **New...** from the pop-up menu. To update an existing data source, right-click the required data source and select **Properties** from the pop-up menu.

Figure 16-4 shows the Data Source Properties window, with the General tab selected.

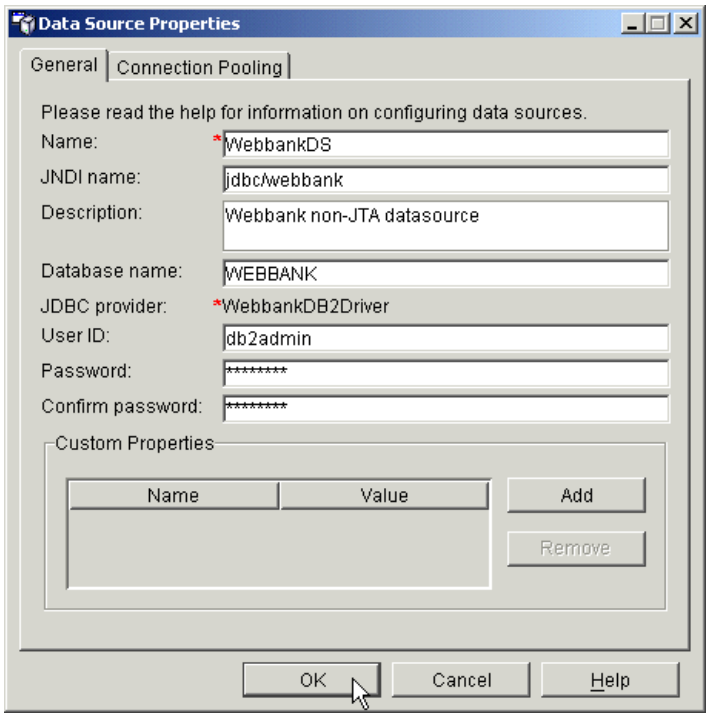


Figure 16-4 Data Source Properties - General tab

General properties of data sources are:

- Name

  A name by which to administer the data source.

  Use a name that is suggestive of the database you will use to store data, such as WASDataSource, where WAS is the database name.

- JNDI name

  The JNDI name is the location in the name space at which to bind the JDBC resource definition. When installing an application that contains modules with JDBC resource references, this JNDI name can be used to resolve one or more of the references.

  The default value for this property is the value of the Name property prefixed with "jdbc/" (such as "jdbc/DataSourceName").

- Description

  A description of the data source, for your administrative records.

- Database name

The name of the database used to store entity bean data.

This is required for DB2, and sometimes required for Sybase, Merant, and Informix (depending on your database configuration), and ignored for Oracle.

- JDBC provider

  The JDBC provider (also known as data source provider) with which this data source is associated. It is used to connect to a relational database.

- User ID

  The user name for connecting to the database when no user ID and password pair is specified by the application. If the user ID is specified, the password must also be specified.

- Password and Confirm password

  The password for connecting to the database when no user ID and password pair is specified by the application. If the password is specified, the user ID must also be specified.

- Custom Properties

  A set of custom name-value pairs describing properties of the data source.

  DB2 does not require custom properties. Oracle, for example, requires a URL property indicating the database from which the data source will obtain connections, such as "jdbc:oracle:thin:@myServer:1521:myDatabase," where "myServer" is the server name, "1521" is the port it is using for communication, and "myDatabase" is the database name.

  See the full listing of required properties for each database type in the InfoCenter.

Configuring connection pooling

Connection pooling can improve the response time of applications that require connections, especially Web-based applications. When a user makes a request over the Web to a resource, the resource accesses a data source. Most user requests do not incur the overhead of creating a new connection because the data source may locate and use an existing connection from the pool of connections.

When the request is satisfied and the response is returned to the user, the resource returns the connection to the connection pool for reuse. Again, the overhead of a disconnect is avoided. Each user request incurs a fraction of the cost of connection or disconnecting. After the initial resources are used to produce the connections in the pool, additional overhead is insignificant because the existing connections are reused.

To configure data source connection pooling, select the **Connection Pooling** tab of the Data Source Properties window, as shown in Figure 16-5.

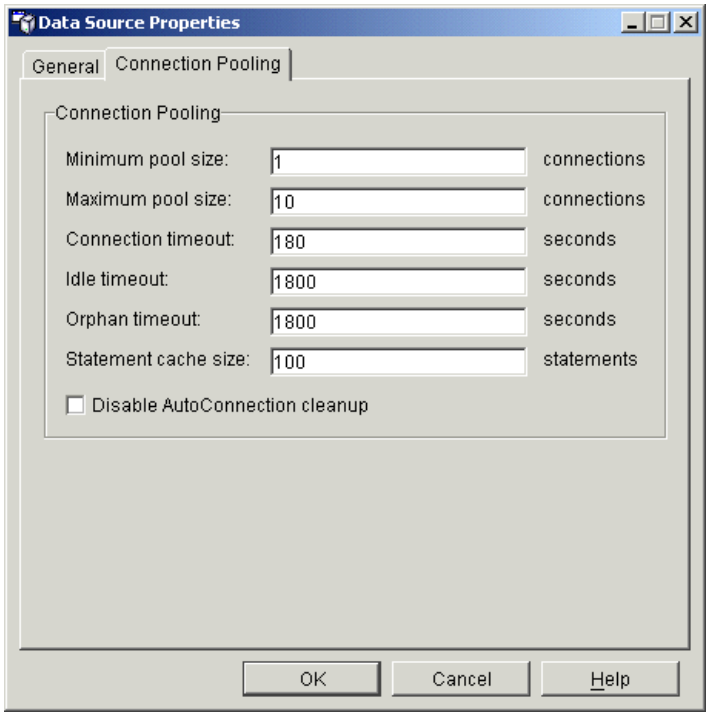


Figure 16-5 Data Source Properties - Connection Pooling tab

Connection pooling properties (values must be positive integers):

- Minimum pool size

  The minimum number of connections in the pool. It should be sufficiently large to address the startup performance penalty that can occur when the connection pool grows in response to an increase in load.

- Maximum pool size

  The maximum number of connections that can be in the pool. If the maximum number of connections is reached and all connections are in use, additional requests for a connection wait up to the number of seconds specified in the Connection timeout property.

  The maximum connection pool size should be large enough to provide a high probability that a new request has access to an open connection. The size of the pool needs to be balanced against the performance impact of creating too large a pool and the delay incurred when a request has to wait for an open connection. The specifics for any environment should be made through

performance experiments prior to production deployment, and confirmed using the Resource Analyzer during production. The maximum connection pool size should also ensure that the maximum number of connections does not exceed the available database licenses or network infrastructure capability.

- Connection timeout

  The maximum time in seconds that requests for a connection wait if the maximum number of connections is reached and all connections are in use.

  In general, the connection timeout should be less than the expected browser timeout value. This allows the connection timeout to occur, the exception to be caught by the application, and user-friendly error page (such as server busy, please retry) to be displayed to the user, rather than a browser timeout error page. In addition, in a system whose database or database connections are overloaded, a sufficiently small connection timeout value would prevent the database query from being performed when the results would (of necessity) be discarded by the application due to a browser timeout. The connection timeout should be large enough to allow for reasonable queueing delays to the database server.

- Idle timeout

  The maximum time in seconds that an idle (unallocated) connection can remain in the pool before being removed to free resources.

  The idle timeout should be configured to allow reasonable sharing of connection resources between the WebSphere processes and any other processes using the same database.

- Orphan timeout

  The maximum number of seconds that an application can hold a connection without using it before the connection can be returned to the pool.

  The orphan timeout is a provision to handle an application's failure to release connections or other similar failures. The orphan timeout should be large enough to ensure that all database requests made upon obtaining the connection can be completed (including queueing delays) in non-failure situations. The orphan timeout should be small enough so that it permits reasonable recovery time in the event of failures.

  Note that the actual amount of time before a connection is closed is approximately twice the orphan timeout value.

- Statement cache size

  The maximum number of prepared statements to cache for the data source. The limit is shared among all connections. The default value is 100. Consider increasing the cache size if an application uses more than 100 different prepared statements.

- Disable AutoConnection cleanup

  Keeps the connection pooling software from automatically closing connections from this data source at the end of a transaction. This behavior is needed if you want to reuse the same connection across multiple transactions. When this is set, you must be sure to close the connection programmatically when you are through using it.

16.1.3 More information

These documents and Web sites are also relevant as further information sources:

- WebSphere InfoCenter, “Accessing data” and “Administering database connections” articles.

  http://www.ibm.com/software/webservers/appserv/infocenter.html

- WebSphere Connection Pooling white paper

  http://www.ibm.com/software/webservers/appserv/whitepapers.html

- JDBC API documentation

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

16.2 JavaMail sessions

JavaMail models an e-mail and messaging service.The JavaMail APIs provide a platform, and protocol independent framework to build e-mail client applications based on Java technology. JavaMail requires the Java Activation Framework (JAF) as the underlying framework to deal with complex data types that are not plain text, like MIME (Multipurpose Internet Mail Extensions), URL (Uniform Resource Locator) pages, and file attachments.

To create platform-independent applications, a JavaMail program uses a resource factory reference. This resource factory reference is used to obtain a JavaMail session. A resource factory is an object that provides access to resources in a program's deployed environment using the naming conventions defined by JNDI (Java Naming and Directory Interface).

JavaMail Sessions are represented by the javax.mail.Session class. The "session" object validates a JavaMail user, and controls the user's access to the message storage and transport services.

WebSphere Application Server V4.0 supports JavaMail Version 1.1.3 and the JavaBeans Activation Framework Version 1.0.1. All Web components of WebSphere, namely servlets, JSPs, EJBs, and applications, support JavaMail. In addition to JAF, JavaMail requires service providers.

16.2.1 What are JavaMail service providers?

JavaMail service providers implement specific protocols. For example, Simple Mail Transfer Protocol (SMTP) is a transport protocol for sending mail. Post Office Protocol (POP3) is the standard protocol for receiving mail. Internet Message Access Protocol (IMAP) is an alternative protocol to POP3.

WebSphere comes equipped with two service providers, SMTP for sending mail and IMAP for receiving mail. To use other protocols, install the appropriate service providers for those protocols.

Figure 16-6 illustrates the relationship among the different JavaMail components:

- JavaMail APIs
- JavaBeans Activation Framework
- Service providers
- Mail protocols

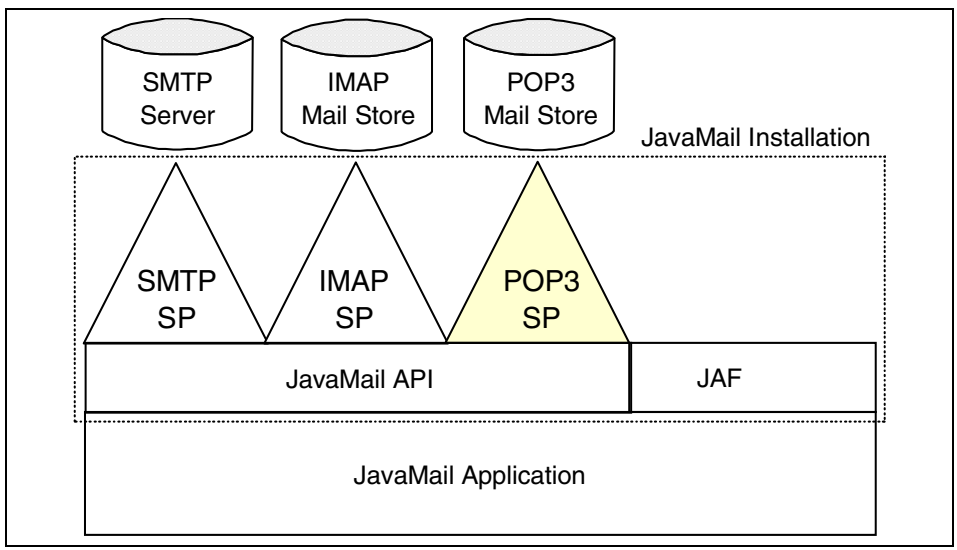


Figure 16-6 JavaMail components

The following Sun licensed packages are part of WebSphere Application Server:

- mail.jar - contains JavaMail APIs, the SMTP service provider, and the IMAP service provider
- activation.jar - contains the JavaBeans Activation Framework

These JAR files can be found in <WAS_HOME>/java/jre/lib/ext directory along with all the other Java extension packages.

Note: WebSphere does not include a POP3 service provider.

16.2.2 Configuring JavaMail sessions

Use the WebSphere Administrative Console to administer JavaMail sessions. As shown in Figure 16-7, you can work with JavaMail sessions in the tree view, by clicking **Resources** -> **JavaMail Sessions** -> session_name.

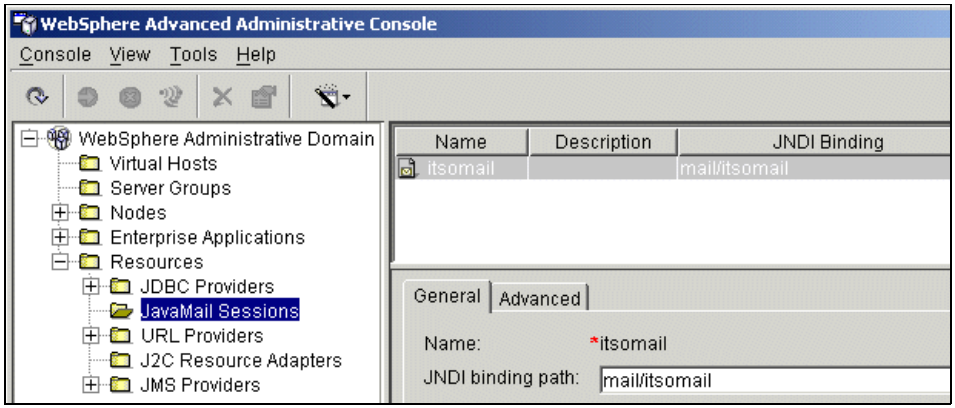


Figure 16-7 Working with JavaMail session

To create a new JavaMail session, right-click **JavaMail Sessions** and select **New...** from the pop-up menu. To update an existing JavaMail session, right-click the required JavaMail session object and select **Properties** from the pop-up menu.

As shown in Figure 16-8, the JavaMail session general properties are:

- Name

  Administrative name of the JavaMail session object.

- JNDI Binding Path

The JNDI binding path is the location in the name space at which to bind the JavaMail resource definition. When installing an application that contains modules with JavaMail resource references this JNDI name can be used to resolve one or more of the references.

The default value for this property is the value of the Name property prefixed with "mail/" (such as "mail/mailName").

- Protocol

  By default it is SMTP for outgoing mail.

- Server

  The server to which to connect when sending mail. Specify the fully qualified Internet host name of the mail server, also known as the SMTP server.

- Mail Originator

  Value of replyTo field in mail messages sent through the mail transport host. The Internet e-mail address that by default will be displayed in the received message as either the "From" or the "Reply-To" address. The recipient's reply will come to the specified address.

- User Name

  The user ID to provide when connecting to the mail transport host. This property is rarely used for the default SMTP protocol. This field can be left blank unless you use a transport protocol that requires a user ID and password.

- Password

  The password to provide when connecting to the mail transport host. Like the user name, this property is rarely used for the default SMTP protocol. This field can be left blank unless you use a transport protocol that requires a user ID and password.

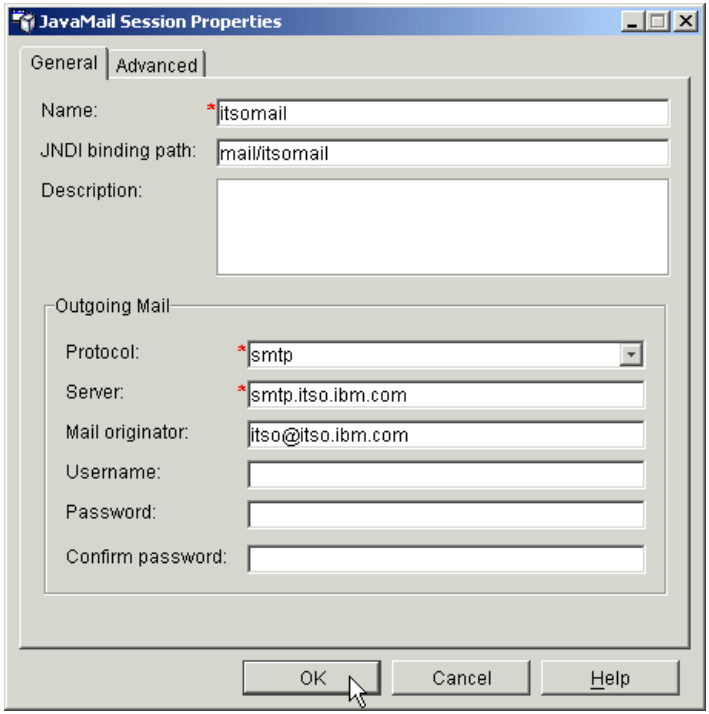


Figure 16-8 JavaMail Session Properties - General tab

Select the **Advanced** tab to enable mail store access. You need to do this only if the application has a retrieve mail functionality. The advanced properties for the JavaMail session, seen in Figure 16-9, are:

- Protocol

  The protocol to be used when reading mail. Currently only IMAP is supported.

- Host

  The server to which to connect when reading mail. This property combines with the mail store user ID and password to represent a valid mail account. For example, if the mail account is itso@itso.ibm.com, enter itso.ibm.com.

- Username

  The user ID to use when connecting to the mail store host. This property combines with the mail store user ID and password to represent a valid mail account. For example, if the mail account is itso@itso.ibm.com, enter itso.

- Password

  The password to use when connecting to the mail store host. This property combines with the mail store user ID and password to represent a valid mail account. For example, if the mail account is itso@itso.ibm.com, enter the password corresponding to itso.

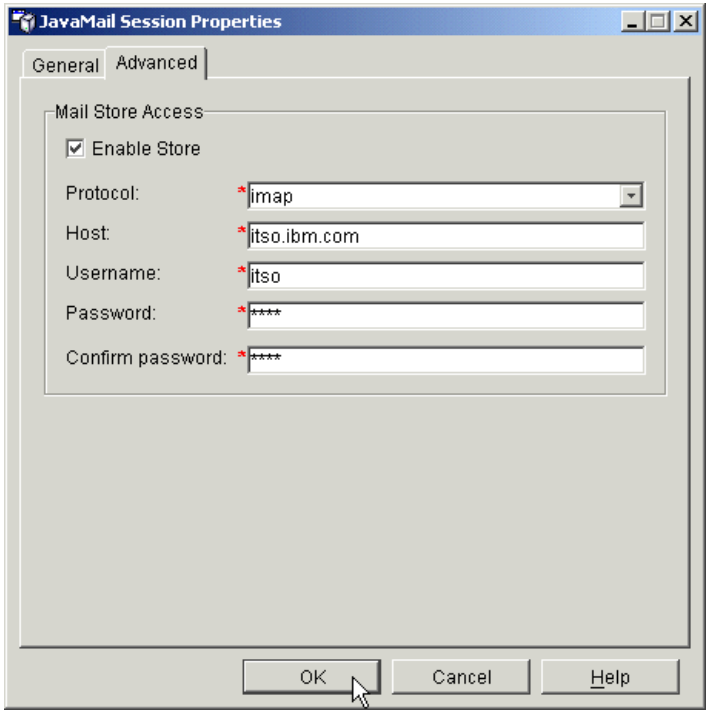


Figure 16-9 JavaMail Session Properties - Advanced tab

16.2.3 More information

These documents and Web sites are also relevant as further information sources:

- WebSphere InfoCenter, “Using JavaMail” and “Administering mail providers and mail sessions” articles.

  http://www.ibm.com/software/webservers/appserv/infocenter.html

- JavaMail API design specification

  http://java.sun.com/products/javamail/javamail-1_1_3.html

16.3 URL providers

A Uniform Resource Locator (URL) is an identifier that points to a resource that is accessible electronically, such as a file in a directory on a machine on a network or documents stored in databases.

URLs are in the format scheme:scheme_information. A scheme might be http, ftp, file, or another term that identifies the type of resource and the mechanism by which the resource can be accessed. The scheme_information commonly identifies the Internet machine making a resource available, the path to that resource, and the resource name.

For the URL `http://www.ibm.com/software/webservers/appserv/library.html`, the scheme is `http`, and the scheme_information is `//www.ibm.com/software/webservers/appserv/library.html`. From the scheme information we can identify the machine (`www.ibm.com`), the path (`/software/webservers/appserv/`), and the resource name (`library.html`).

In WebSphere V4.0, you can use URL resources to define application URLs in a way that is consistent with the J2EE model, where platform resources are allocated during application packaging or deployment. You can also use URL resources to implement custom URL protocols that allow applications to access other resources using standard Java URL APIs.

16.3.1 What are URL providers?

A URL provider implements the functionality for a particular URL protocol, such as HTTP. It is a pair of classes that extend java.net.URLStreamHandler and java.net.URLConnection.

A URL provider named *Default URL Provider* is included in the initial WebSphere configuration. It utilizes the URL support provided by the JDK. Any URL resources with protocols supported by the JDK (such as http, ftp, file, and mailto) should use the Default URL Provider.

Customers can also "plug-in" their own URL providers that implement other protocols not supported by the JDK (such as nntp for example). WebSphere allows customers to supply their own classes to associate application code lookups with URLs implemented by these custom URL providers.

16.3.2 Configuring URL providers and URLs

Use the WebSphere Administrative Console to administer URL providers and URLs. As shown in Figure 16-10, you can work with URL providers and URL resources in the tree view, by clicking **Resources** -> **URL Providers** -> provider_name. Expand the tree further to see URLs -> url_name for each provider.

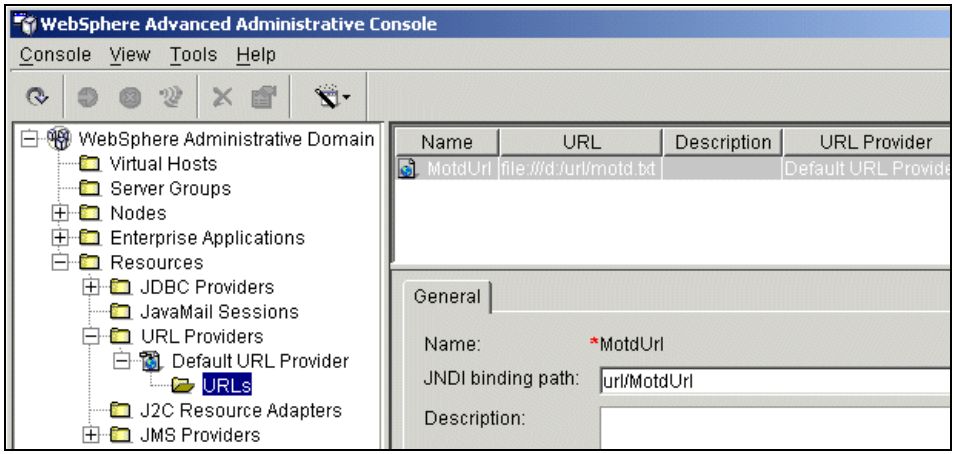


Figure 16-10 Working with URL providers and URLs

Configuring URL providers

To create a new URL provider, right-click **URL Providers** and select **New...** from the pop-up menu. To update an existing URL provider, right-click the required provider and select **Properties** from the pop-up menu.

As shown in Figure 16-11, the URL provider general properties are:

- Name

  Administrative name for the URL provider.

- Description

  Optional description of the URL provider, for your administrative records.

- Protocol

  The custom protocol supported by this stream handler. This field is set to “unused” for the Default URL Provider.

  For a custom URL provider for accessing online stock quote information, for example, you might set this to “quote”.

- Stream handler class

  The fully qualified name of Java class that implements the stream handler for the protocol specified by the Protocol property. This field is set to “unused” for the Default URL Provider.

  For a custom URL provider for accessing online stock quote information, for example, you might set this to “com.itso.QuoteURLStreamHandler”.

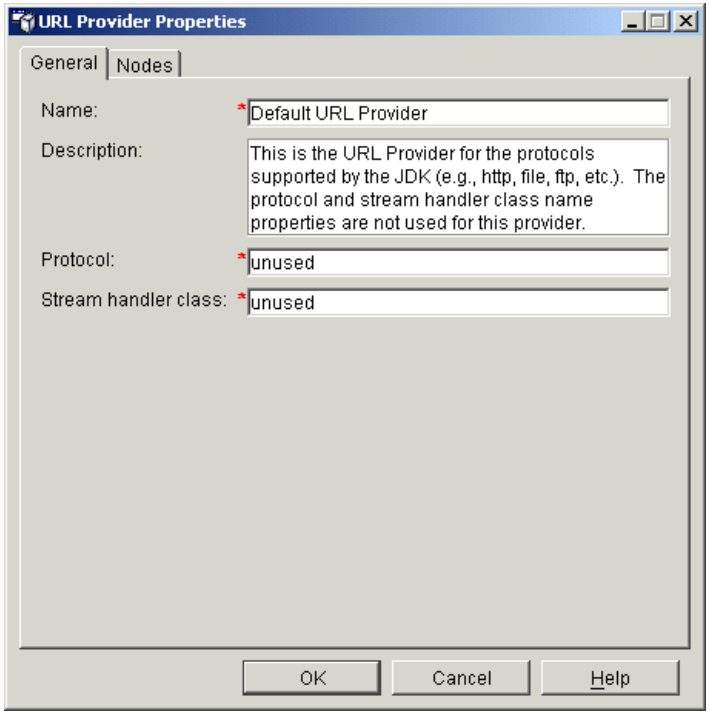


Figure 16-11 URL Provider Properties - General tab

You need to install a driver on each node where the URL provider is used. The nodes properties for the URL providers, seen in Figure 16-12, are:

- Node

  The node with which the URL provider is associated.

  Click the **Install New...** and **Uninstall** buttons on the Nodes tab to access windows for installing drivers on specific nodes and for uninstalling drivers.

- Classpath

  The path to the Java file containing the implementation classes for the URL provider. This field is set to “unused” for the Default URL Provider.

For a custom URL provider for accessing online stock quote information, for example, you might set this to “D:\myclasses\quote.jar”.

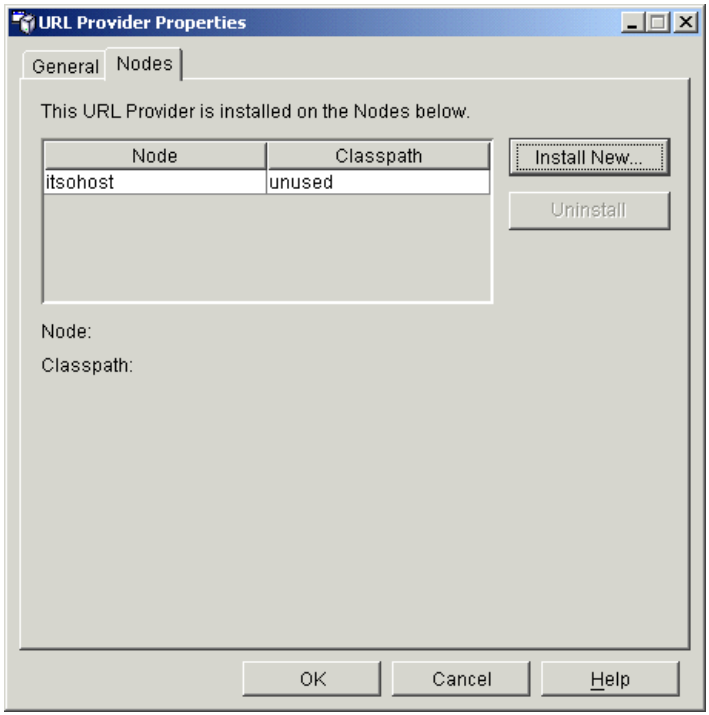


Figure 16-12 URL Provider Properties - Nodes tab

Configuring URLs

To create a new URL, right-click **URLs** and select **New...** from the pop-up menu. To update an existing URL, right-click the required URL and select **Properties** from the pop-up menu.

As shown in Figure 16-13, the properties of URLs are:

- Name

  Administrative name for the URL.

- JNDI binding path

  The JNDI binding path is the location in the name space at which to bind the URL resource definition. When installing an application which contains modules with URL resource references this JNDI name can be used to resolve one or more of the references.

  The default value for this property is the value of the Name property prefixed with "url/" (such as "url/UrlName").

- Description

  Optional description of the URL, for your administrative records.

- URL Provider

  The URL provider that implements the protocol for this URL.

- URL

  The string from which to form a URL.

  For a custom URL provider for accessing online stock quote information, for example, you might set this to “quote://IBM”.

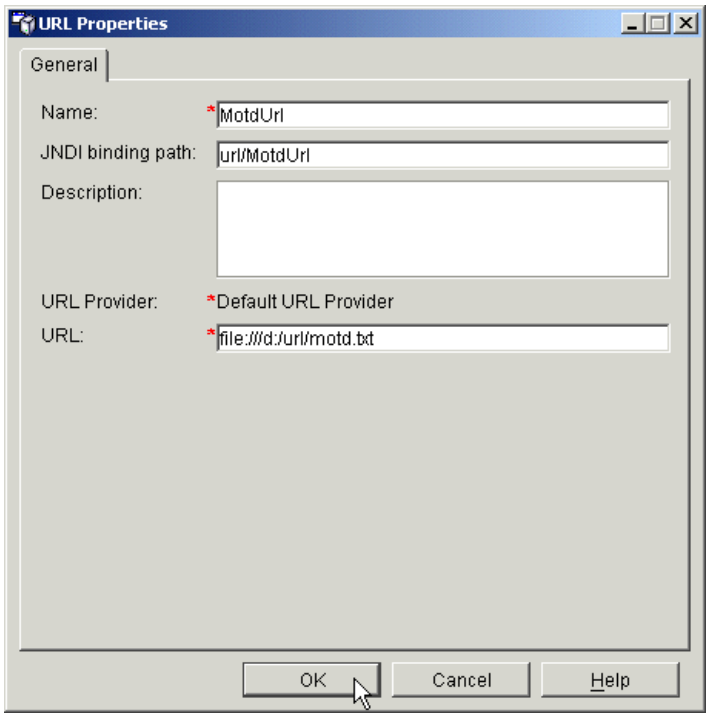


Figure 16-13 URL Properties

16.3.3 URL provider sample

Example 16-1 provides a sample making use of the URL provider and URL resources described in 16.3.2, “Configuring URL providers and URLs” on page 579.

Note that the Web module resource reference, myHttpUrl, is bound to the URL resource JNDI name, url/MotdUrl, during application assembly or deployment.

Example 16-1 HTTP URL provider sample

```
<%
   javax.naming.InitialContext ctx = new javax.naming.InitialContext();
   javax.naming.Context env =
      (javax.naming.Context) ctx.lookup("java:comp/env");
   java.net.URL url = (java.net.URL) env.lookup("myHttpUrl");
   java.io.InputStream ins = url.openStream();
   int c;
   while ((c = ins.read()) != -1) {
      out.write(c);
      }
%>
```

In our case, we inserted the Example 16-1 code into a JSP, added the JSP to a Web module, added a URL resource reference to the Web module, and deployed the Web module. Then we checked that the contents of the file specified in the MotdUrl URL resource, "file:///d:/url/motd.txt", were included in the JSP's output.

Similarly, a stock quote custom URL provider could be accessed as shown in Example 16-2. The Web module resource reference, myQuoteUrl, is bound to a URL resource with JNDI name, url/QuoteUrl, and URL "quote://IBM". The custom URL provider will access an online stock quote for IBM.

Example 16-2 Quote URL provider sample

```
<%
   javax.naming.InitialContext ctx = new javax.naming.InitialContext();
   javax.naming.Context env =
      (javax.naming.Context) ctx.lookup("java:comp/env");
   java.net.URL url = (java.net.URL) env.lookup("myQuoteUrl");
   out.println("The stock price is "+url.getContent());
%>
```

Note: Each application server's name space is initialized on startup. This means application servers must be restarted to load a modified resource property, such as URL string.

16.3.4 More information

These documents and Web sites are also relevant as further information sources:

- WebSphere InfoCenter, "What are URLs and URL providers" and "Administering URL providers and URLs" articles.

16.4 J2C resource adapters

The Connector Architecture (technology preview) for WebSphere Application Server (J2C) provides a J2EE-compliant way for Java applications to interact with non-relational back-end systems such as CICS and IMS.

As shown in Figure 16-14, J2C uses resource adapters that J2EE applications can access via a common client interface API. The common client interface API is provided with the Connector Architecture (technology preview) for WebSphere Application Server. Resource adapters are supplied by enterprise information system (EIS) vendors or by third-party vendors.

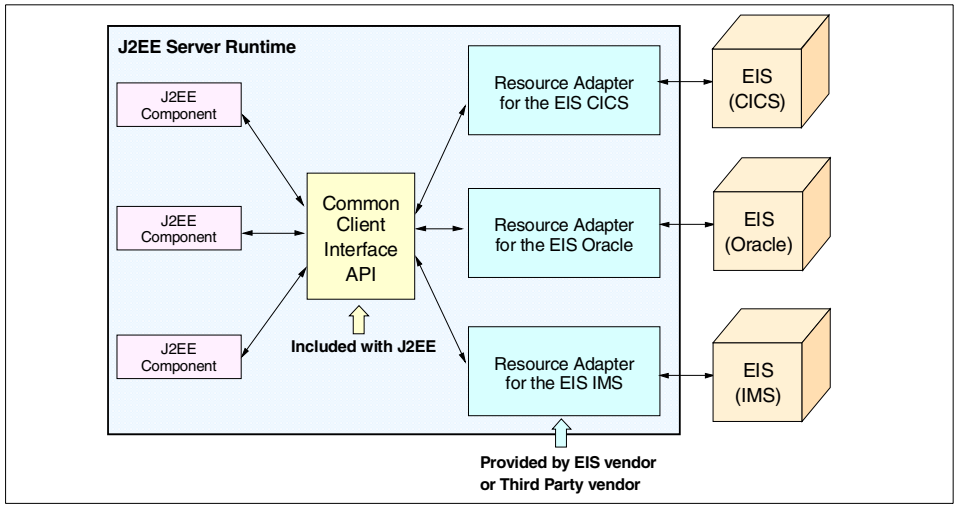


Figure 16-14 J2C architecture

Benefits of the J2C connector architecture include:

- The common client interface simplifies application integration with diverse EISs.
- Each EIS requires just one implementation of the resource adapter, as there is no need to custom develop an adapter for every application.

16.4.1 What are J2C resource adapters and connection factories?

A J2C resource adapter represents a library that supports connections from an application to a non-relational back-end system. That is, a J2C adapter supports connections to an enterprise information system (EIS) resource.

A Resource Adapter Archive (RAR) file is a Java archive (JAR) file used to package a resource adapter for the Connector Architecture for WebSphere Application Server.

A RAR file can contain the following:

- EIS-supplied resource adapter implementation code in the form of JAR files or other executables, such as DLLs.
- Utility classes.
- Static documents, such as HTML files, images, and sound.
- Connector architecture (J2C) common client interfaces, such as cci.jar.

The standard file extension of a RAR file is .rar.

A J2C connection factory is one set of connection configuration values. Application components, such as enterprise beans, have resource-ref descriptors that refer to a specific connection factory, not to the resource adapter. You can think of the connection factory as a holder of a list of configuration properties.

In addition to the arbitrary set of configuration properties defined by the vendor of the resource adapter, there are several standard configuration properties that apply to the connection factory. These standard properties are used by the J2C connection pool manager in the application server runtime and are not known by the vendor-supplied resource adapter code.

16.4.2 Installing the J2C runtime

The Connector Architecture (technology preview) for WebSphere Application Server is not installed with the base WebSphere installation. After installing the base WebSphere Application Server V4.0, Advanced Edition product, you then need to install the J2C runtime.

To install the Connector Architecture (technology preview):

1. Stop all WebSphere processes, including the administrative server and Web server.
2. Run the Connector Architecture (technology preview) installation batch file found in the <CD_root>/<os>/connector directory:

```
installJ2C
```

Note: You can also download the Connector Architecture (technology preview) from `http://www7b.boulder.ibm.com/wsdd/downloads/jca.html`.

3. Follow the instructions provided by the installation program.
4. When the installation program completes, verify that <WAS_HOME>/lib directory contains the j2c.jar, jca.jar, eablib.jar, recjava.jar, and ccf.jar files.

The Connector Architecture (technology preview) installation program will place its uninstall program in the <WAS_HOME> directory.

16.4.3 Configuring J2C resource adapters and connection factories

Use the WebSphere Administrative Console to administer J2C resource adapters and connection factories. As shown in Figure 16-15, you can work with J2C resource adapters and connection factories in the tree view, under **Resources** -> **J2C Resource Adapters** -> adapter_name. For each adapter, expand the tree further by clicking **J2C Connection Factories** -> connection_name.

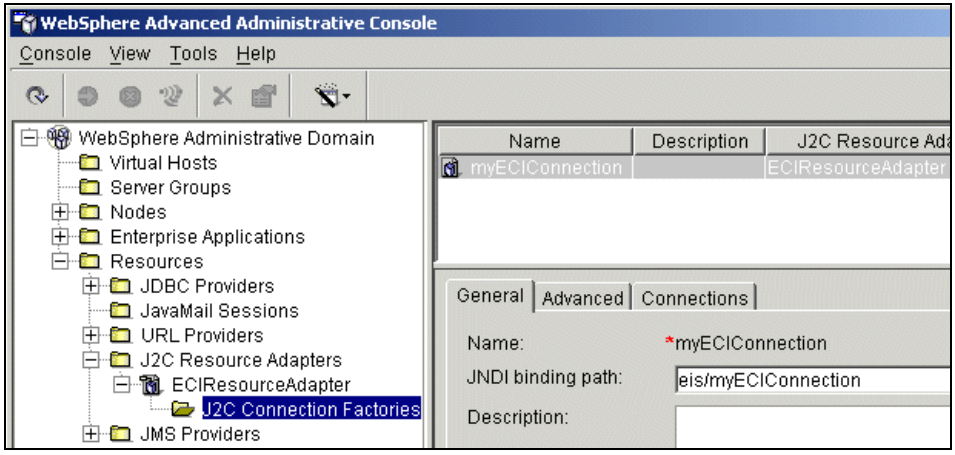


Figure 16-15 Working with J2C resource adapters and connection factories

Configuring J2C resource adapters

To create a new J2C resource adapter right-click **J2C Resource Adapters** and select **New...** from the pop-up menu. To update an existing J2C resource adapter, right-click the required adapter and select **Properties** from the pop-up menu.

As shown in Figure 16-16, the J2C resource adapter general properties are:

- Name

  Administrative name for the resource adapter.

- Description

  Optional description of the resource adapter, for your administrative records.

- Archive file name

  Path to the Resource Adapter Archive (RAR) file for the resource adapter.

  Use the **[...]** button on the right to access windows for selecting .rar files on specific nodes.

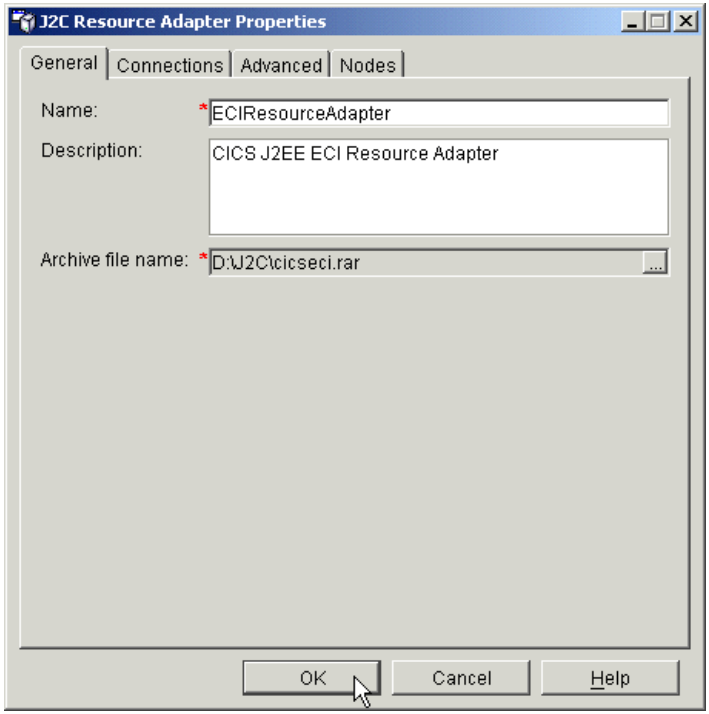


Figure 16-16 J2C Resource Adapter Properties - General tab

As shown in Figure 16-17, the J2C resource adapter connections properties are:

- Custom connection properties

  Connection properties that are specific to the given J2C adapter, including a name, description, data type, and value for each property.

  These properties are available for information only here.

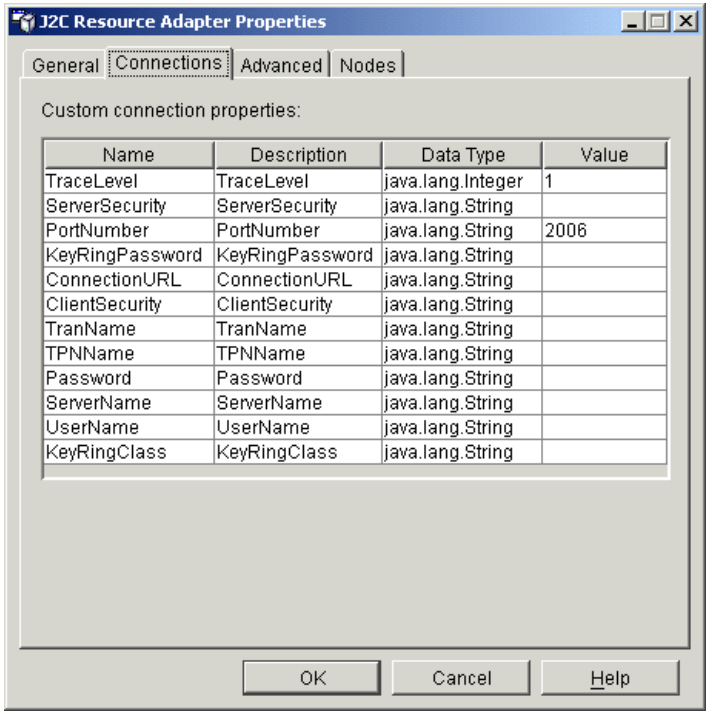


Figure 16-17 J2C Resource Adapter Properties - Connections tab

As shown in Figure 16-18, the J2C resource adapter advanced properties are:

- Vendor properties

  Connection properties that are specific to the given J2C adapter. See your adapter documentation for further details on these properties.

  These properties are available for information only here.

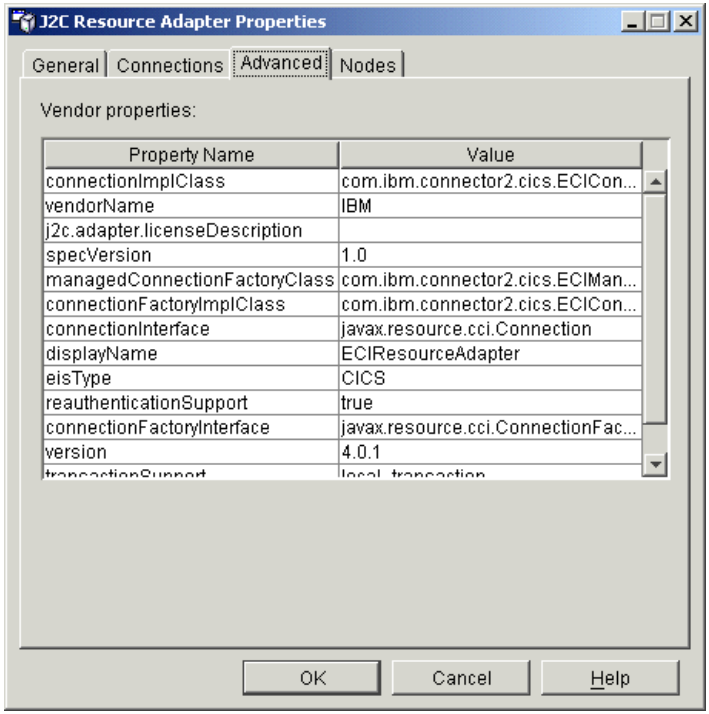


Figure 16-18 J2C Resource Adapter Properties - Advanced tab

You need to install a driver on each node where the J2C resource adapter is used. The nodes properties for the J2C resource adapter, seen in Figure 16-19, are:

- Node

  The administrative node on which the J2C resource adapter is installed.

  Click the **Install New...** and **Uninstall** buttons on the Nodes tab to access windows for installing drivers on specific nodes and for uninstalling drivers.

- Classpath

  The path to the Java file containing the implementation classes for the J2C resource adapter.

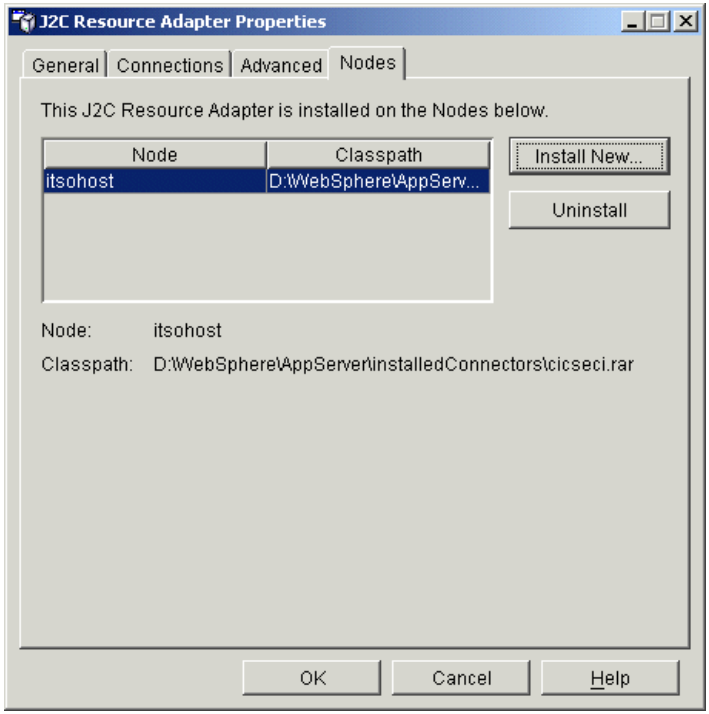


Figure 16-19 J2C Resource Adapter Properties - Nodes tab

Configuring J2C connection factories

To create a new J2C connection factory right-click **J2C Connection Factories** and select **New...** from the pop-up menu. To update an existing J2C connection factory, right-click the required connection factory and select **Properties** from the pop-up menu.

As shown in Figure 16-20 the general properties of J2C connection factories are:

- Name

  Administrative name for the J2C connection factory.

- JNDI binding path

  The JNDI binding path is the location in the name space at which to bind the J2C resource definition. When installing an application that contains modules with J2C resource references this JNDI name can be used to resolve one or more of the references.

  The default value for this property is the value of the Name property prefixed with "eis/" (such as "eis/connectionName").

- Description

  Optional description of the J2C connection factory, for your administrative records.

- J2C Resource Adapter

  The J2C resource adapter on which this J2C connection factory is based.

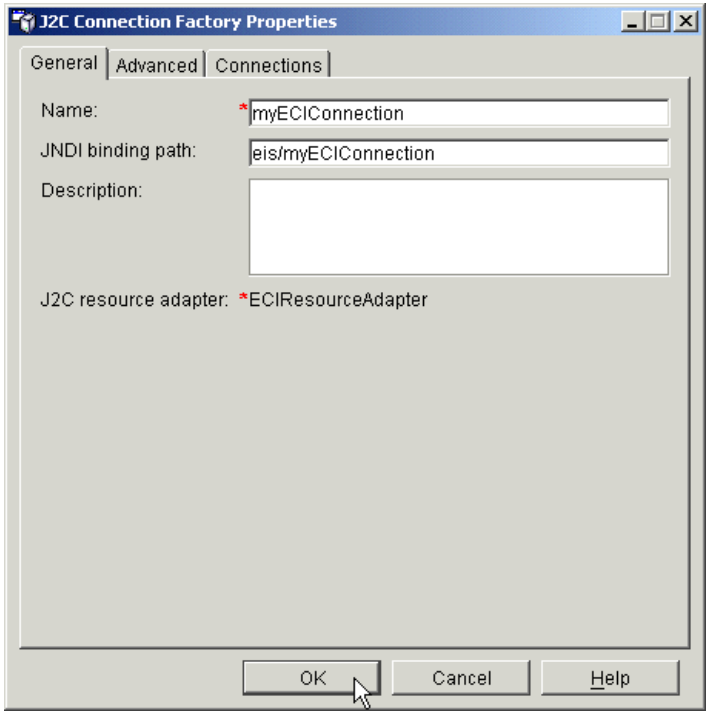


Figure 16-20 J2C Connection Factory Properties - General tab

As shown in Figure 16-21 the advanced properties of J2C connection factories are:

- Connection Timeout

  If the maximum number of connections has been reached, the connection will wait for the specified number of milliseconds before aborting and throwing a ResourceAllocationException. If the maximum number of connections has not been reached, or is set to 0, Connection Timeout will not be used.

  If Connection Timeout is 0, the pool manager will wait indefinitely.

- Maximum Connections

  The maximum number of managed connections that can be created by a particular ManagedConnectionFactory. After this number is reached, no new connections are created and either the requester waits for the Connection Timeout or a ResourceAllocationException is thrown. If Maximum Connections is set to 0, the number of connections can grow indefinitely. Maximum Connections must be larger than or equal to Minimum Connections.

- Minimum Connections

  The minimum number of managed connections to maintain. If this number is reached, the garbage collector will not discard any managed connections. Note, if the actual number of connections is lower than the value specified by the minimum connections settings, no attempt will be made to increase the number of connections to the minimum. Minimum Connections must be less than or equal to Maximum Connections.

- Reap Time

  Number of seconds between runs of the garbage collector. The garbage collector discards all connections that have been unused for the value specified by the unused timeout.

  To disable the garbage collector, set the reap time to 0 or set the unused timeout to 0.

- Unused Timeout

  Number of milliseconds after which an unused connection is discarded. Setting this value to 0 disables the garbage collector.

- Pool Name

  The name used by the pool manager to group managed connections created by different ManagedConnectionFactories.

- Subpool Name

  The name used by the pool manager to group managed connections created by different ManagedConnectionFactories within a particular pool.

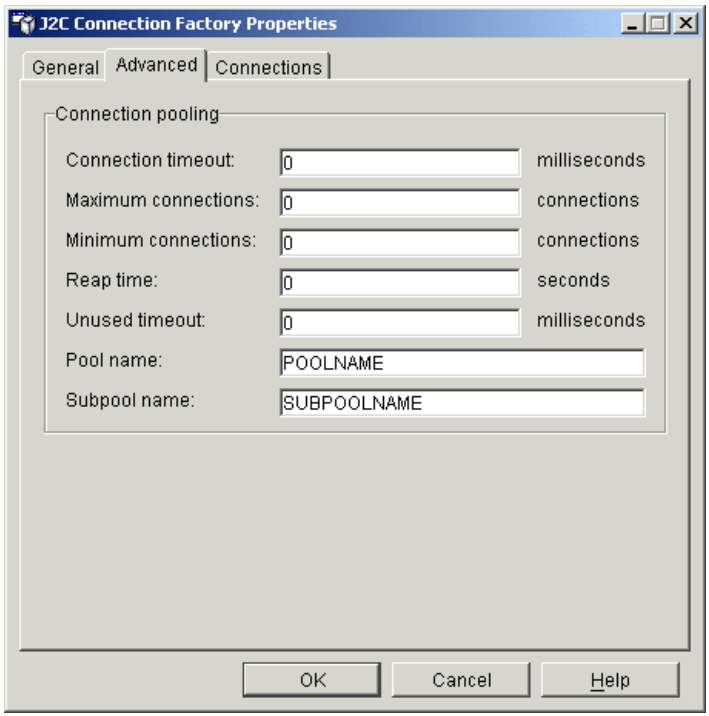


Figure 16-21 J2C Connection Factory Properties - Advanced tab

As shown in Figure 16-22 the connections properties of J2C connection factories specify the set of connection configuration values. You can see that one of the configuration values of our particular resource adapter, the ConnectionURL, defines the URL of the CICS Transaction Gateway.

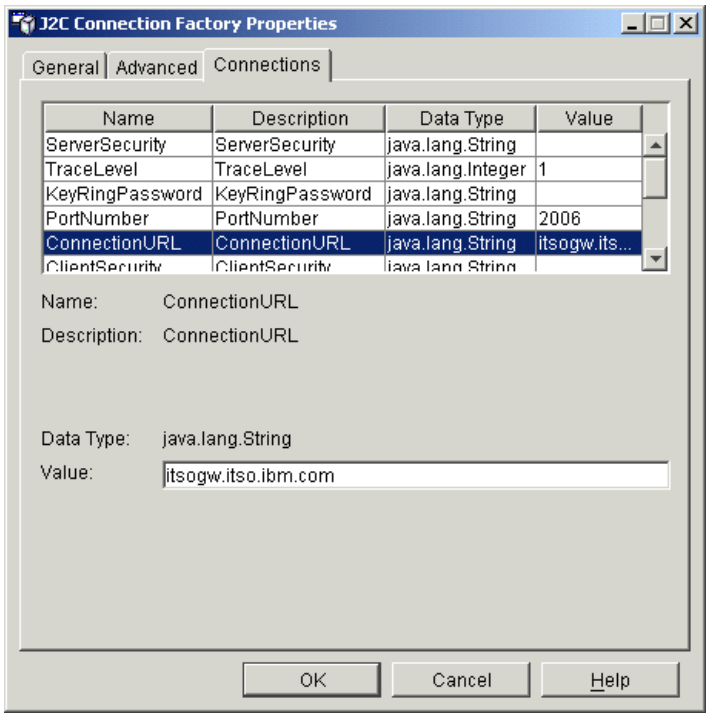


Figure 16-22 J2C Connection Factory Properties - Connections tab

16.4.4 More information

These documents and Web sites are also relevant as further information sources:

- WebSphere InfoCenter, “What are J2C adapters” and “Administering J2C related administrative resources” articles.

  `http://www.ibm.com/software/webservers/appserv/infocenter.html`

- Connector Architecture for WebSphere Application Server (technology preview) download page

  `http://www7b.boulder.ibm.com/wsdd/downloads/jca.html`

- J2EE Connector Architecture documentation

  `http://java.sun.com/j2ee/connector`

- Information on CICS and the CICS Transaction Gateway (CTG) V4.0

  `http://www.ibm.com/software/ts/cics`

16.5 JMS providers

The Java Messaging Service (JMS) supports the development of message-based applications in the Java programming language, allowing for the asynchronous exchange of data and events throughout an enterprise.

Messaging systems map well to business process flows and are well suited to both time-independent and parallel processing applications. Using assured "once-only" delivery mechanisms, these systems can also be made more tolerant of failures (such as network outages).

16.5.1 What are JMS providers?

A JMS provider is a base messaging system and related Java classes that implement the JMS API. With WebSphere Application Server, JMS provider resources offer the following:

- Connection factories to help an application create connections to specific JMS providers. Each factory has defined parameters that configure a connection.
- Destination objects that an application uses to specify the target of messages it produces and the source of messages it consumes. A JMS destination provides a specific endpoint for messages.

16.5.2 Installing a JMS provider

IBM supplies a JMS provider implementation with MQSeries. MQSeries JMS support is available for IBM MQSeries Server V5.2 in two support packs:

- MA88 MQSeries classes for Java and MQSeries classes for Java Message Service
- MA0C MQSeries - Publish/Subscribe

These support packs are available at:

http://www.ibm.com/software/ts/mqseries/txppacs

MQ JMS support provides a tool named JMSAdmin for administering JMS objects (such as connection factories, queues, and topics) and binding them to a JNDI name space.

To configure JMSAdmin to use the WebSphere JNDI name space, modifying the following properties in JMSAdmin.config:

```
INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory
PROVIDER_URL=iiop://localhost:900
```

You can then use JMSAdmin to define the required JMS connection factory and queue objects, and bind them to the WebSphere JNDI name space.
Example 16-3 shows how we defined the QCF connection factory and Q1 queue JMS objects used in 16.5.3, “Configuring JMS resources” on page 597.

Example 16-3 Defining JMS connection factory and queue objects using JMSAdmin

```
InitCtx> define QCF(QCF)
InitCtx> define Q(Q1) QUEUE(SYSTEM.DEFAULT.LOCAL.QUEUE)
InitCtx> display ctx
Contents of InitCtx
      ...
   a  QCF                       com.ibm.mq.jms.MQQueueConnectionFactory
      ...
   a  Q1                        com.ibm.mq.jms.MQQueue
      ...
InitCtx>
```

The WebSphere JMS provider resource provides a configurable link between the Web/EJB module resource references and the required MQ JMS JNDI names. It also adds the MQ JMS support classes to classpath to all application servers on the node.

As shown in Figure 16-23, application resource references are mapped to the JMS resource in the application deployment descriptor. The JMS resource properties then map to the "external" JNDI name that map to our external MQ JMS provider objects.

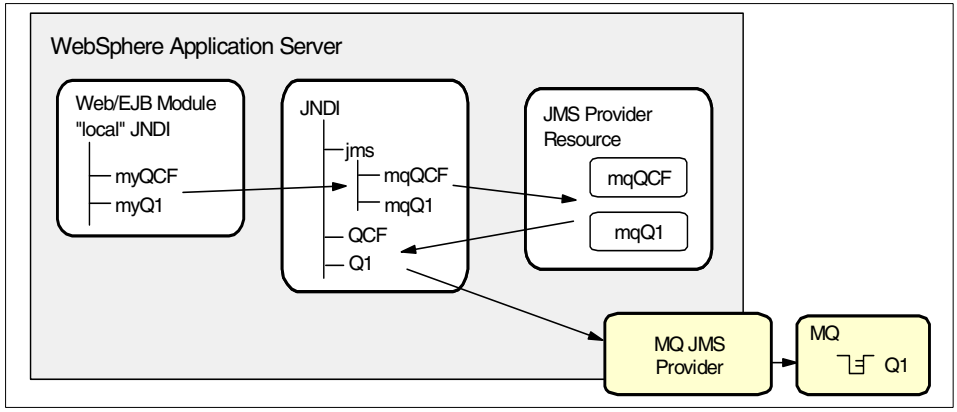


Figure 16-23 MQSeries access using the WebSphere JMS provider resources

16.5.3 Configuring JMS resources

Use the WebSphere Administrative Console to administer JMS resources. As shown in Figure 16-24, you can work with JMS providers, connection factories, and destinations in the tree view, under **Resources** -> **JMS Providers** -> provider_name. For each provider, expand the tree further by clicking **JMS Connection Factories** -> factory_name and **JMS Destinations** -> dest_name.

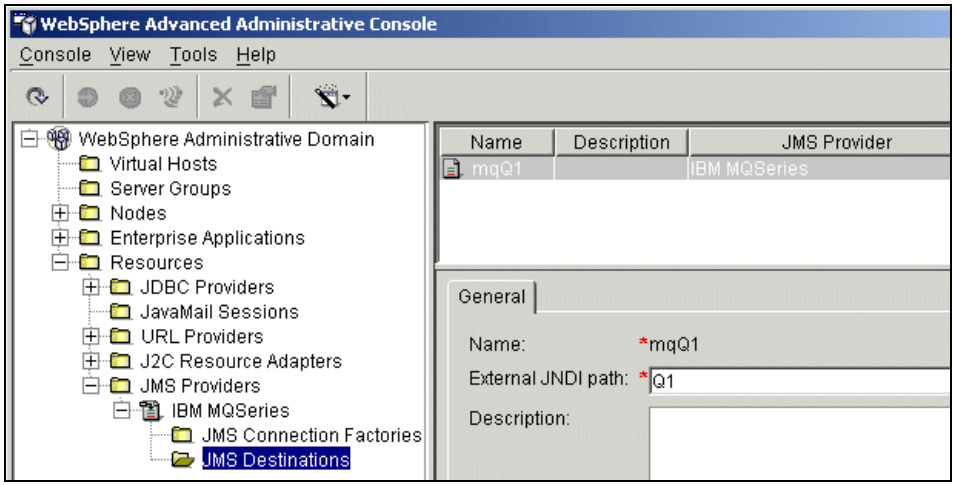


Figure 16-24 Working with JMS providers, connection factories, and destinations

Configuring JMS providers

To create a new JMS provider, right-click **JMS Providers** and select **New...** from the pop-up menu. To update an existing JMS provider, right-click the required provider and select **Properties** from the pop-up menu.

As shown in Figure 16-25, the JMS provider general properties are:

- Name

  Administrative name for the JMS provider.

- Description

  Optional description of the JMS provider, for your administrative records.

- Context factory classname

  The Java class name of the JMS provider's initial context factory.

- Provider URL

  JMS provider URL for external JNDI lookups.

- Binding classname

  Java classname to be used for name space binding. This value is required only for providers with non-standard binding requirements.

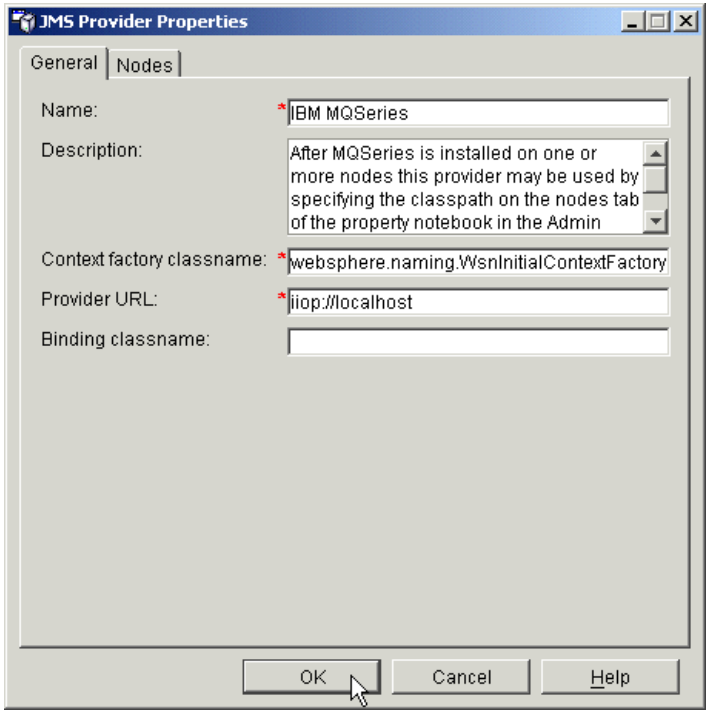


Figure 16-25 JMS Provider Properties - General tab

You need to install a driver on each node where the JMS provider is used. The nodes properties for the JMS providers, seen in Figure 16-26, are:

- Node

  The node with which the JMS provider is associated.

  Click the **Install New...** and **Uninstall** buttons on the Nodes tab to access windows for installing drivers on specific nodes and for uninstalling drivers.

- Classpath

  The class path that identifies the location of the implementation classes for the JMS provider. For the MQ JMS support pack, we specified the following paths on a single line, separated by semi-colons (use colons on UNIX).

```
D:\IBM\MQSeries\Java\lib;
D:\IBM\MQSeries\Java\lib\com.ibm.mq.jar;
D:\IBM\MQSeries\Java\lib\com.ibm.mqjms.jar;
D:\IBM\MQSeries\Java\lib\fscontext.jar;
```

`D:\IBM\MQSeries\Java\lib\providerutil.jar`

Substitute `D:\IBM\MQSeries\Java` with the %MQ_JAVA_INSTALL_PATH% directory where you installed the MQ JMS support pack.

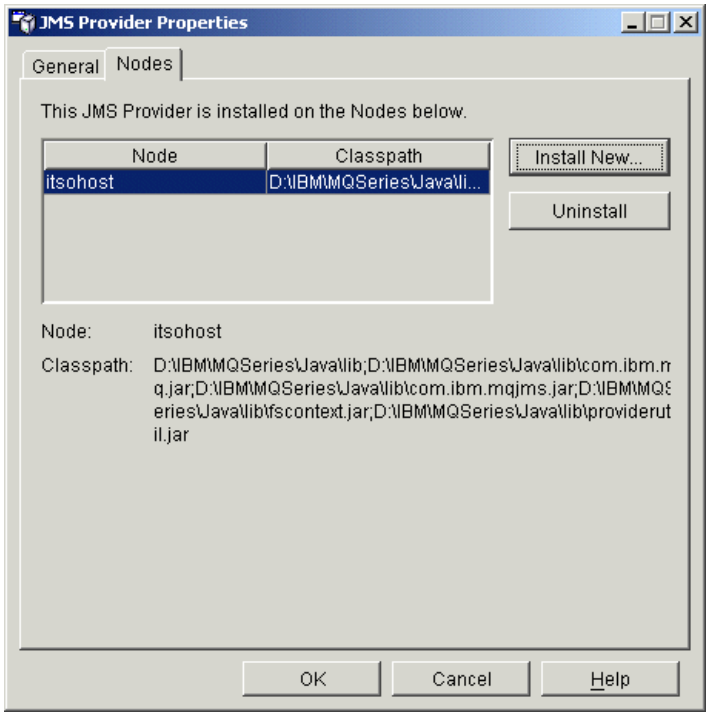


Figure 16-26 JMS Provider Properties - Nodes tab

Configuring JMS connection factories

To create a new JMS connection factory, right-click **JMS Connection Factories** and select **New...** from the pop-up menu. To update an existing JMS connection factory, right-click the required connection factory and select **Properties** from the pop-up menu.

As shown in Figure 16-27, the properties of JMS connection factories are:

- Name

  Administrative name for the JMS connection factory.

- External JNDI path

  The JNDI binding path of external JMS connection factory.

- Description

  Optional description of the JMS connection factory, for your administrative records.

- JMS Provider

  The JMS provider associated with this connection factory.

Note: WebSphere automatically assigns an internal JNDI path to each WebSphere connection factory resource. The value assigned is the Name property prefixed with "jms/" (such as "jms/jmsName"). This name is the location in the name space at which to bind the JMS resource definition. When installing an application which contains modules with JMS resource references this JNDI name can be used to resolve one or more of the references.

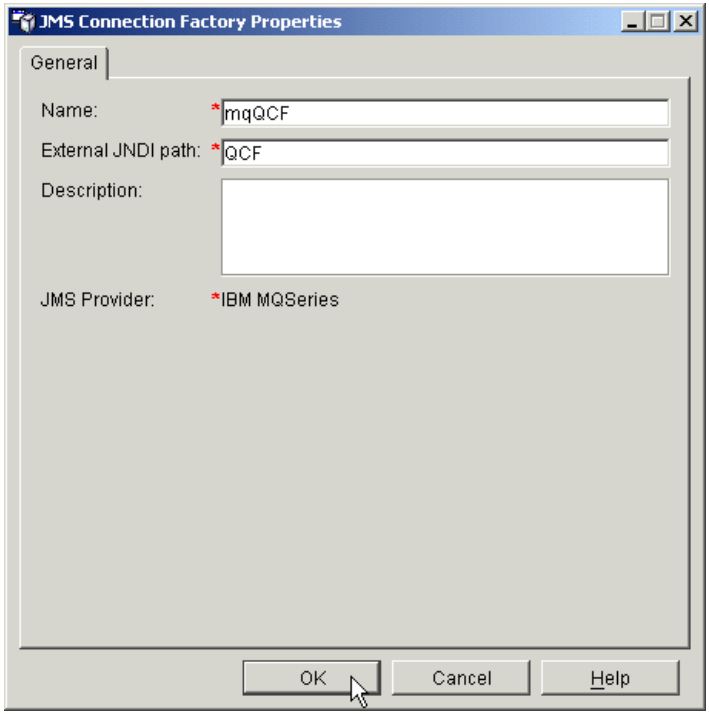


Figure 16-27 JMS Connection Factory Properties

Configuring JMS destinations

To create a new JMS destination, right-click **JMS Destinations** and select **New...** from the pop-up menu. To update an existing JMS destination, right-click the required JMS destination and select **Properties** from the pop-up menu.

As shown in Figure 16-28 the properties of JMS destinations are:

- Name

  Administrative name for the JMS destination.

- External JNDI path

  The JNDI binding path of external JMS destination.

- Description

  Optional description of the JMS destination, for your administrative records.

- JMS Provider

  The JMS provider associated with this JMS destination.

Note: WebSphere automatically assigns an internal JNDI path to each WebSphere JMS destination resource. See the note in “Configuring JMS connection factories” on page 599 for details.

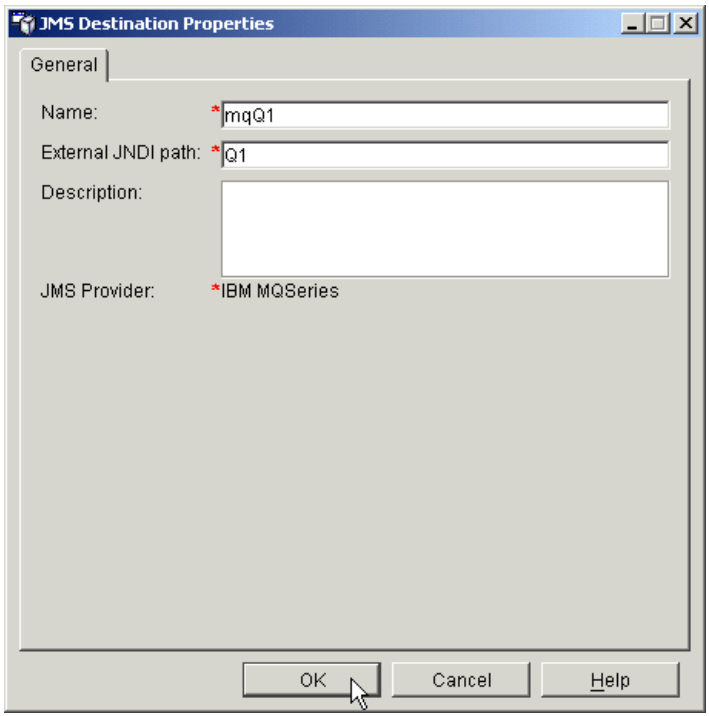


Figure 16-28 JMS Destination Properties

16.5.4 JMS provider sample

Example 16-4 provides a simple sample making use of the JMS provider and JMS resources described in 16.5.2, “Installing a JMS provider” on page 595 and 16.5.3, “Configuring JMS resources” on page 597.

Note that the Web module resource references, myQCF and myQ, are bound to the JMS resource JNDI names, jms/mqQCF and jms/mqQ, during application assembly or deployment.

Example 16-4 JMS provider sample

```
<%

   // get JNDI context
   javax.naming.InitialContext ctx = new javax.naming.InitialContext();

   // get local JNDI environment
   javax.naming.Context env =
      (javax.naming.Context) ctx.lookup("java:comp/env");
```

```
    // get queue connection factory
    javax.jms.QueueConnectionFactory qcf =
       (javax.jms.QueueConnectionFactory) env.lookup("myQCF");

    // create a connection
    javax.jms.QueueConnection qc = qcf.createQueueConnection();

    // create a session
    javax.jms.QueueSession qSession =
       qc.createQueueSession(false, javax.jms.Session.AUTO_ACKNOWLEDGE);

    // get queue
    javax.jms.Queue q = (javax.jms.Queue) env.lookup("myQ");

    // create a queue sender for the queue and send a text message
    javax.jms.QueueSender sender = qSession.createSender(q);
    sender.send(qSession.createTextMessage("Hello world!"));
    sender.close();

%>
```

In our case, we inserted the Example 16-4 code into a JSP, added the JSP to a Web module, added JMS resource references to the Web module, and deployed the Web module. Then using MQSeries Explorer to browse messages, we checked that our “Hello World!” test message was sent to SYSTEM.DEFAULT.LOCAL.QUEUE when the JSP was accessed.

Note: Each application server’s name space is initialized on startup. This means application servers must be restarted to load a modified resource property, such an external JNDI path.

16.5.5 More information

These documents and Web sites are also relevant as further information sources:

- WebSphere InfoCenter, “What are JMS and JMS providers” and “Administering messaging and JMS providers” articles.

  http://www.ibm.com/software/webservers/appserv/infocenter.html

- Java Message Service API documentation

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

- MA88: MQSeries classes for Java and MQSeries classes for Java Message Service.

  http://www.ibm.com/software/ts/mqseries/txppacs/ma88.html

- WebSphere Developer Domain Library, WebSphere MQ JMS support articles.

  http://www7b.boulder.ibm.com/wsdd/library/techtip/0111_cox.html

  http://www7b.boulder.ibm.com/wsdd/library/techtip/0112_cox.html

16.6 Configuring application client resources

As already discussed, application server resources are configured using the WebSphere Administrative Console. To configure application client resources, use the Application Client Resource Configuration Tool, shown in Figure 16-29. These configurations are stored in the client application EAR file, and are used by the WebSphere application client runtime for resolving and creating an instance of the resources for the client application.

You can launch this GUI using the <WAS_HOME>/bin/clientConfig script.

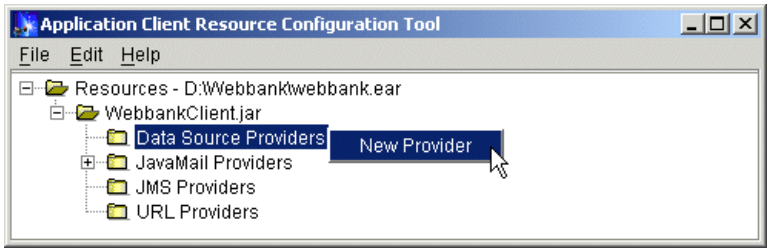


Figure 16-29 Application Client Resource Configuration Tool

Refer to the InfoCenter for more information on using the Application Client Resource Configuration Tool.

17

Server groups and workload management

This chapter discusses workload management and how it is implemented in WebSphere using server groups and clones. Session management implications (HttpSession affinity and HttpSession persistence) are discussed in Chapter 15, “Configuring session management” on page 513.

17.1 Managing workloads

Workload management (WLM) is the process of spreading multiple requests for work over the resources that can do the work. It optimizes the distribution of processing tasks in the WebSphere Application Server environment. Incoming work requests are distributed to the application servers and other objects that can most effectively process the requests.

Workload management is also a procedure for improving performance, scalability and reliability of an application. It provides failover when servers are not available.

Workload management is most effective when used in systems that contain servers on multiple machines. It can also be used in systems that contain multiple servers on a single, high-capacity machine. In either case, it enables the system to make the most effective use of the available computing resources.

Workload management provides the following benefits to WebSphere applications:

- It balances client processing requests, allowing incoming work requests to be distributed according to a configured WLM selection policy.
- It provides failover capability by redirecting client requests to a running server when one or more servers are unavailable. This improves the availability of applications and administrative services.
- It enables systems to be scaled up to serve a higher client load than provided by the basic configuration. With server groups and clones additional instances of servers can easily be added to the configuration. See 17.2, “Server groups and clones” on page 607 for details.
- It enables servers to be transparently maintained and upgraded while applications remain available for users.
- It centralizes administration of application servers and other objects.

As shown in Figure 17-1 on page 607, two types of requests can be workload managed in WebSphere Application Server V4.0, Advanced Edition:

- Servlet requests. Can be distributed across multiple Web containers. We will refer to it as “Plug-in WLM”.
- EJB requests. Can be distributed across multiple EJB containers. We will refer to it as “EJS WLM”.

Note: All resources that participate in workload management must be running under the same version of WebSphere Application Server.

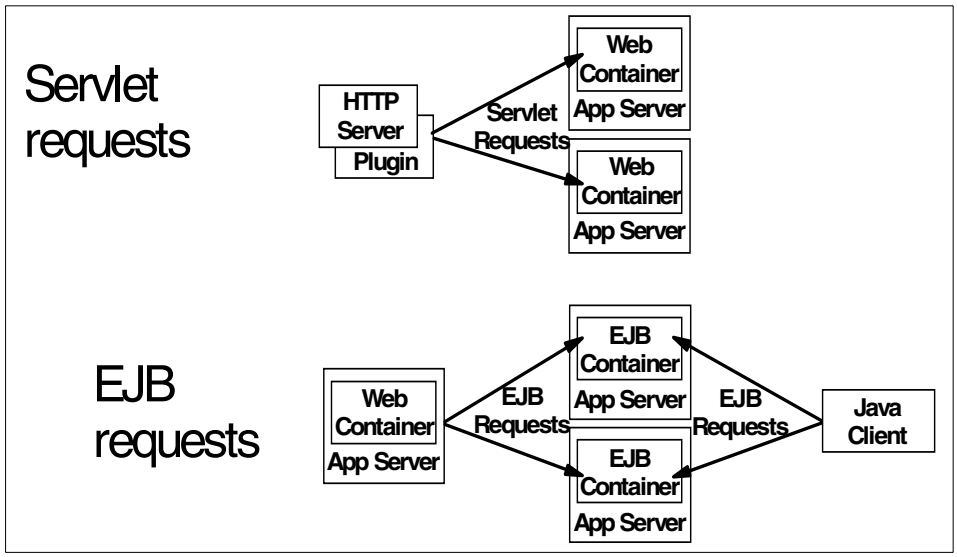


Figure 17-1 Requests that can be workload managed

17.2 Server groups and clones

This section describes how workload management is implemented in WebSphere Application Server V4.0, Advanced Edition, by using server groups and clones. Administrative servers can also participate in workload management.

A *server group* is a template of an application server (and its contents). It is based upon a server that you've set up and it is used for creating and managing copies of that server. The copies are called *clones*. The act of creating the clones is called *cloning*.

The server group is a logical representation of the application server. It has the same structure and attributes as the real application server, but it is not associated with any node, and does not correspond to any real server process running on any node. As with the real application server, the server group will contain Web containers, EJB containers, servlets, EJBs, and so on, and you will be able to modify any properties of these logical objects.

Note: Server groups and clones can only be created for application servers. Therefore cloning a server group automatically enables WLM for the servlets and EJBs that it contains.

On the other hand, clones created from this server group represent real application server processes running on real nodes.

Clones are identical in every way to the server group from which they were created, allowing identical copies of application servers to be created. These copies can be used for workload management, since a request for a server resource can be handled by any of the server clones.

Starting or stopping the server group will automatically start or stop all the clones. Changes to a server group are propagated to its clones when the server group is restarted.

Clones can be distributed across different machines.

17.2.1 Creating server groups and clones

A high-level description of the procedure for creating server groups and clones follows:

1. Create the original instance of the application server that you want to clone. Configure it exactly as you would like it. For example, you can deploy enterprise beans into the application server's container and configure the application server to meet specific performance goals.
2. Create a server group from the application server by using the WebSphere Administrative Console.

   You don't always need to create server groups from existent application servers though. As an alternative:

   a. Go the administrative console.

   b. Select **Server Groups**.

   c. Right-click **Server Groups** and select **New...** from the pop-up menu.
3. Create clones of the server group.
4. Regenerate the Web server plug-in configuration file.
5. When changes are necessary, apply them to the server group, which in turn modifies the original instance (which is now a clone) and the other clones.
6. Install the enterprise application on the local node.
7. Expand the enterprise application archive (EAR) on each of the remote nodes.

How to create server groups and clones as per the first method described above is detailed in 17.7.1, "Create server group from application server" on page 628 and 17.7.2, "Create clone from server group" on page 630.

Tip: If you plan to create a clone on another node, make sure you have configured a JDBC driver on that node prior to creating the clone.

17.2.2 Administering server groups

Administrators can manage clones efficiently by managing the server group on which they are based.

After cloning an application server, modifying the application server group automatically propagates the same changes to all of the application server clones.

Note: If an application server is used as a base to create a server group, its modules will be taken away from it and put into server group context. Do not be alarmed if the modules seem to be missing from the original application server!

It is important to remember that to perform an administrative action on a clone, such as modifying the clone's properties, the administrator must perform the action on the associated server group instead.

Note: If you change a property in a clone and later change the same property in the server group associated with the clone, the change you made directly to the clone could be overwritten when you modify the server group. It is best to avoid modifying a clone directly.

17.2.3 Workload management with server groups and clones

Server groups and clones provide necessary support for workload management, failover, and scalability.

Workload management

When you modify a server group, the change is propagated to all its clones. Besides making it easy to administer several servers as one logical server, this keeps the servers identical so that requests can be routed to any of them with the same results.

This ability to route a request to any server in a group of identical servers allows the servers to share work, improving throughput of client requests. Requests can be evenly distributed to servers to prevent workload imbalances in which one or more servers have idle or low activity while others are overburdened. This load balancing activity is a benefit of workload management.

Failover

With several clones available to handle requests, it is more likely that failures will not damage throughput and reliability. With clones distributed to various nodes, an entire machine can fail without any application downtime. Requests can be routed to other nodes if one node fails.

Scaling

Cloning is an effective way to perform vertical and horizontal scaling of application servers.

- In **vertical scaling**, shown in Figure 17-2, multiple server clones of an application server are defined on the same physical machine, or node, may allow the machine's processing power to be more efficiently allocated.

  Even if a single JVM can fully utilize the processing power of the machine, you may still want to have more than one clone on the machine for other reasons. One such reason is using vertical clones for software failover. If a JVM reaches a table/memory limit (or some similar problem) you can have another process to go to for failover.

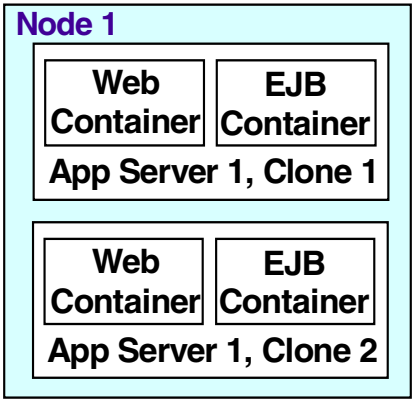


Figure 17-2 Vertical scaling

We recommend that you avoid using "rules of thumb" when determining the number of clones needed for a given machine. The only way to determine what is correct for your environment and application(s) is to tune a single instance of an application server for throughput and performance, then

incrementally add clones. Test performance and throughput as each clone is added. Always monitor memory usage when you are configuring a vertical scaling topology and do not exceed the available physical memory on a machine.

In general, if one is able to utilize 85% (or more) of the CPU on a large server, then there is little, if any, performance benefit to be realized from adding additional clones.

- In **horizontal scaling**, shown in Figure 17-3, clones of an application server are created on multiple physical machines. This allows a single WebSphere application to run on several machines while still presenting a single system image, making the most effective use of the resources of a distributed computing environment. Horizontal scaling is especially effective in environments that contain many smaller, less powerful machines. Client requests that overwhelm a single machine can be distributed over several machines in the system. Failover is another benefit of horizontal scaling. If a machine becomes unavailable, its workload can be routed to other machines containing clones.

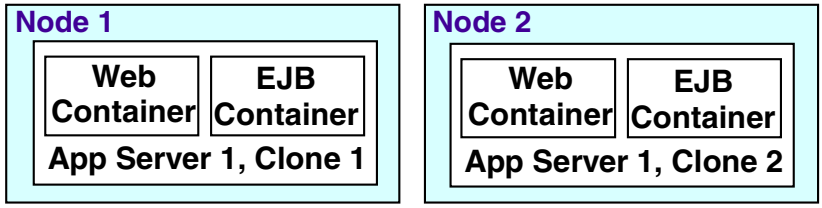


Figure 17-3 Horizontal scaling

Horizontal scaling can handle application server process failure and hardware failure without significant interruption to client service.

Note: Horizontal cloning on different platforms is not supported.

- WebSphere applications can combine horizontal and vertical scaling to reap the benefits of both scaling techniques, shown in Figure 17-4.

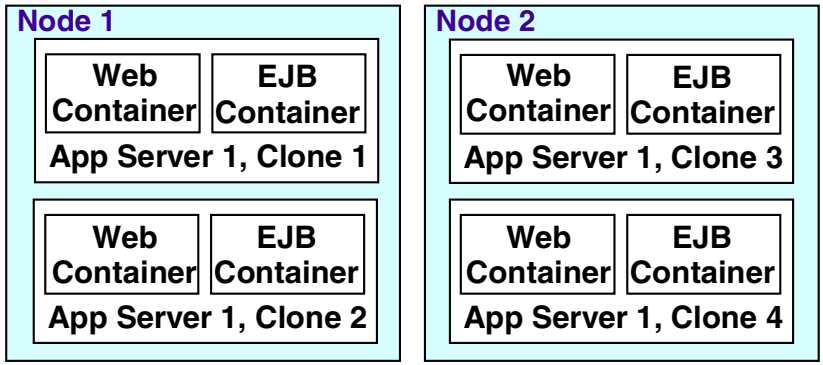


Figure 17-4 Vertical and horizontal scaling

17.3 Plug-in workload management

In this section we discuss plug-in workload management where the Web container and EJB container are on the same application server. The application server is cloned.

Cloning application servers that host Web containers (a Web container works with a Web server to handle requests for servlets and other Web resources, such as HTML files and JavaServer Page files) automatically enables plug-in workload management for the application servers and the servlets they host.

17.3.1 Servlet clustering architecture

In the simplest case the application server clones are configured on a single machine, where the Web server process also runs, as shown in Figure 17-5.

Random or round-robin routing of servlet requests occurs between the Web server plug-in and the cloned application servers using HTTP.

Note: There is a configuration parameter called LoadBalance, settable via the HTTP plug-in configuration file plugin-cfg.xml, that allows you to set load balancing to Random or Round Robin. The default is Round Robin. Refer to Appendix C, “The plugin-cfg.xml file definitions” on page 1071 for details.

The Web server plug-in will temporarily route around unavailable clones. When the servlet is invoked, the servlet only makes in-process calls to EJBs in the *same* application server (see the *process affinity* definition given in 17.5.7, “EJB server selection policy” on page 621).

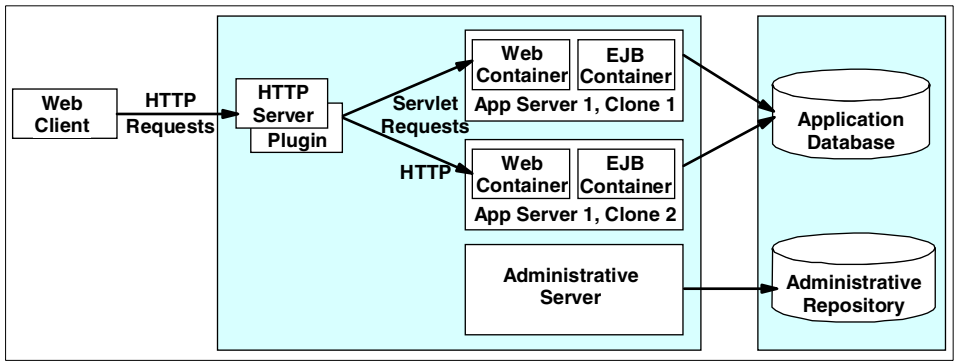


Figure 17-5 Plug-in workload management

Alternatively, the Web server can reside on a separate machine from the application server, typically across one or more firewalls. Figure 17-6 shows this topology.

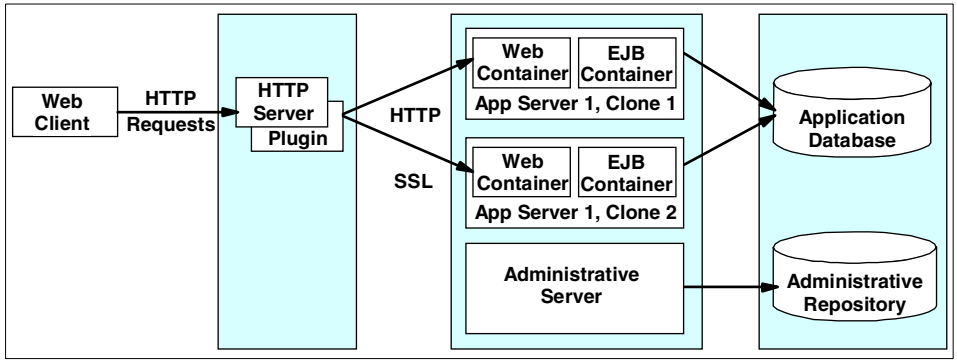


Figure 17-6 Remote HTTP server with clones

The benefits of having a remote Web server include:

- Physical separation of the Web server from the application server processes, usually across one or more firewalls.
- Very thin WebSphere install on the Web server machine.
- System resources contention is avoided.

It is a good practice to configure the WebSphere administrative repository on a separate database server.

Note: The OSE Web server plug-in and the servlet redirector are not supported in WebSphere V4.0.

17.4 Enterprise Java Services workload management

In this section we discuss Enterprise Java Services Workload Management (EJS WLM) where the Web container and EJB container are on different application servers. The application server hosting the EJB container is cloned.

Configuring the Web container in a separate application server from the Enterprise JavaBean container (an EJB container handles requests for both session and entity beans) enables distribution of EJB requests between the EJB container clones, as seen in Figure 17-7.

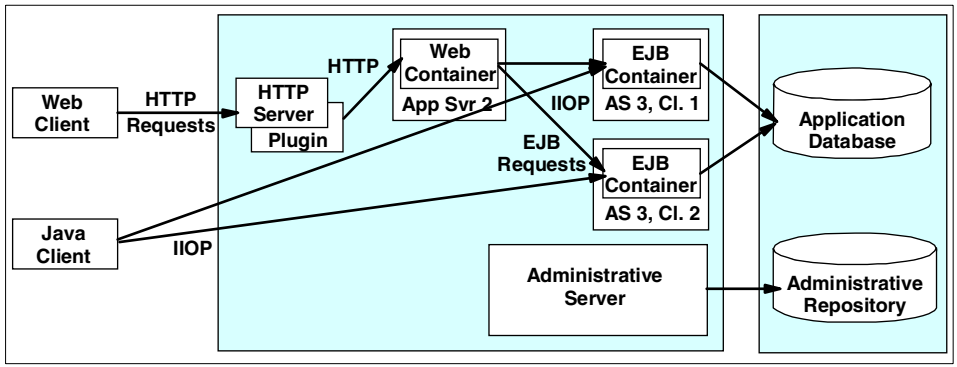


Figure 17-7 EJS workload management

In this configuration, EJB client requests are routed to available EJB container clones based on the workload management selection policy (Random, Round Robin, Random Prefer Local, Round Robin Prefer Local).

Note: This is the WLM EJB selection policy between EJB clients (such as servlets, stand-alone Java programs, or other EJBs) and EJB servers. It is not the WLM selection policy between the Web server and the Web containers, as noted in 17.3.1, “Servlet clustering architecture” on page 612.

The EJB clients can be servlets operating within a Web container, stand-alone Java programs using RMI/IIOP, or other EJBs.

Note: Because this configuration requires communicating using out-of-process RMI/IIOP, it generally performs slower than having both the Web container and the EJB container in the same application server.

17.5 Plug-in and EJS workload management

In this section we discuss plug-in and EJS workload management where the Web container and EJB container are on different application servers. Both application servers are cloned.

Figure 17-8 shows a configuration in which the Web container and the EJB container are implemented on different application servers. Both application servers have been cloned and they reside on different physical machines, allowing for firewall protection of business logic.

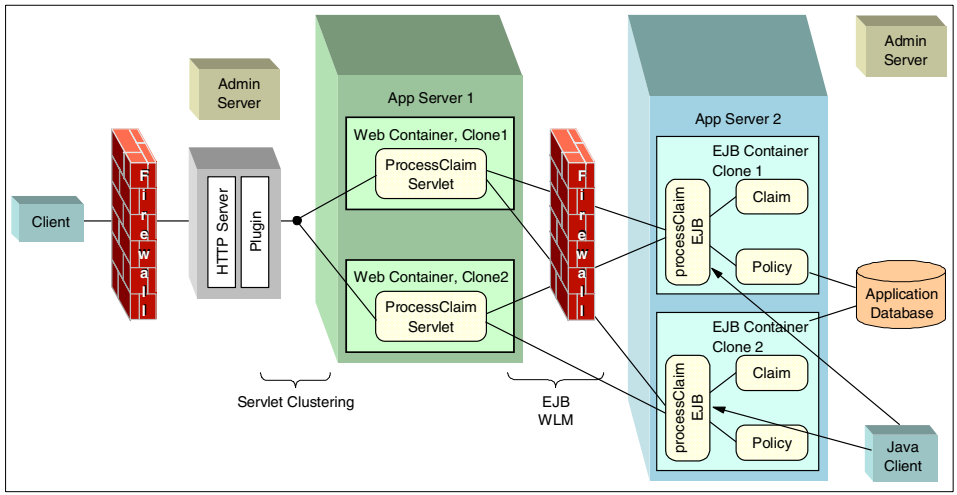


Figure 17-8 Servlet and EJB clustering

Note: In this configuration, in addition to the cost of going out of process (RMI/IIOP), there is the cost associated with being in separate machines (network stacks on the machines involved and network latency). It therefore performs slower than having both the Web container and the EJB container in the same application server and same machine.

17.5.1 Migrating workload-managed EJBs from V3.5 to V4.0

In WebSphere Application Server V3.5, workload management for enterprise beans was enabled by using stub code that allowed EJB clients to access enterprise beans through the workload management service. The wlmjar utility was used to generate this stub code and create workload management-enabled Java archive (JAR) files. This approach has been deprecated for WebSphere Application Server V4.0. Cloned enterprise beans now automatically participate in workload management.

You do not need to make any changes to enterprise beans that participated in workload management under WebSphere Application Server V3.5. The workload management service simply ignores the existing stub code and workload management-enabled JAR files. However, you must remove the name of the workload management-enabled JAR file from the CLASSPATH environment variable. Replace it with the name of the original EJB JAR file.

17.5.2 How EJBs participate in WLM

As mentioned in 17.5.1, “Migrating workload-managed EJBs from V3.5 to V4.0” on page 616, in WebSphere Application Server V4.0 the standard stubs generated during deployment don’t need to be modified in order to have workload management-enabled EJBs. The WLM client runtime is enabled as an ORB (Object Request Broker) plug-in that directs request to EJB clones.

As shown in Figure 17-9 on page 617, client requests for EJBs go through the ORB:

1. The EJB client makes a request. Requests can come from servlets, stand-alone Java clients, or other EJBs.
2. The EJB request goes through the ORB.
3. The ORB asks the WLM plug-in for a reference to a clone from the server group. This reference is called an Interoperable Object Reference (IOR).
4. The WLM plug-in determines whether or not it has seen an IOR for this server group yet.
5. If not, the WLM plug-in contacts its WebSphere administrative server to get server group information.
6. The WLM plug-in uses this server group information (such as EJB server selection policy, list of server clones available, and so on) to select a clone.
7. The plug-in returns the IOR for the selected clone. The IOR refers directly to the clone; no Location Service Daemon lookup is needed.
8. The ORB then directs the EJB request to the selected clone.

9. Subsequent requests are either dispatched to the same clone, if the call is in context of a transaction, or to another clone.
10. When server group information changes (clones started or stopped), the administrative server pushes the new server group information to the clones.
11. The clients get updated when they send a regular (application) method request to one of the clones.

    Along with the method request, the WLM plug-in on the client sends the version of its server group information (via an epoch number). The server clone compares that version to the version it received from the administrative server. If the client's version is stale, the server clone will send new server group information to the client on the reply of the method request.

    From that point on, the client will have current information to use (until the information is changed again).

    There are situations where server clones become unavailable but the information is not yet reflected in the server group information (propagated from the administrative server). In this case, the client could fail in connecting to a server clone.The WLM client plug-in will mark that server clone as bad. No threads of execution in the client application will attempt to use that server clone until two things happen:

    – The unusable interval time expires (using the configurable system property setting com.ibm.ejs.wlm.unusableInterval), in which case the WLM client plug-in can attempt to use that server clone again.
    – New server group information is received, in which case a clone will be selected based on the new list.

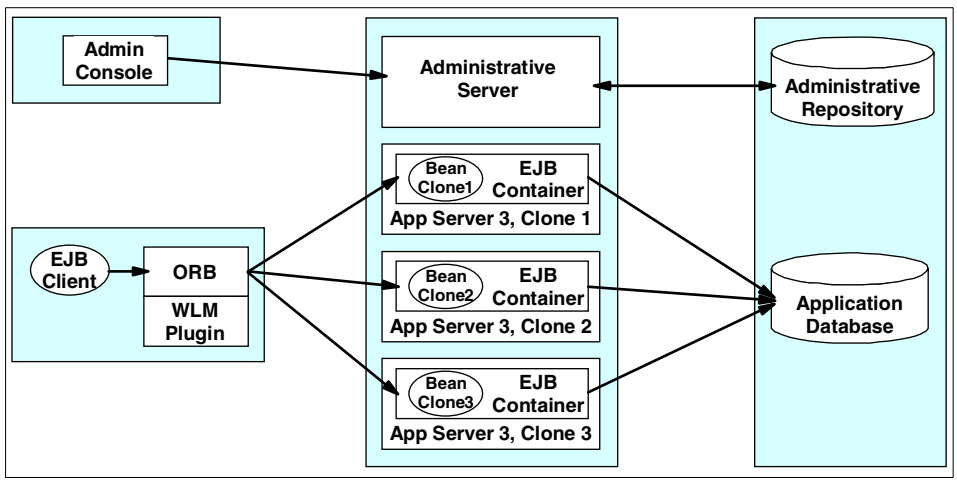


Figure 17-9 ORB directs requests to EJB clones

17.5.3 What is workload managed with EJBs?

The workload management service provides load balancing for the following types of enterprise beans:

- Homes of entity or session beans
- Instances of entity beans
- Instances of stateless session beans

As shown in Figure 17-10, the only type of EJB references not subject to load-distribution through EJS WLM are instances of a given stateful session bean.

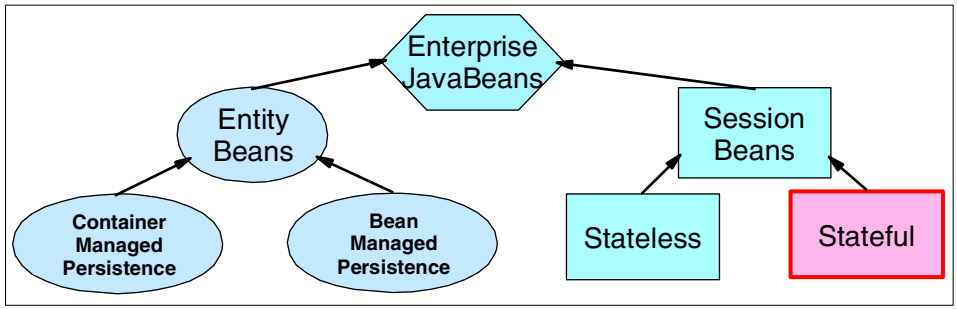


Figure 17-10 Enterprise beans that participate in workload management

17.5.4 Stateless session bean

By definition, a stateless session bean maintains no state information. Each client request directed to a stateless session bean is independent of the previous requests that were directed to the bean. The container maintains a pool of instances of stateless session beans, and provides an arbitrary instance of the appropriate stateless session bean when a client request is received. Requests can be handled by any stateless session bean instance in any clone of the application server, regardless of whether the bean instance handled the previous client requests.

17.5.5 Stateful session beans

A stateful session bean is used to capture state information that must be shared across multiple consecutive client requests that are part of a logical sequence of operations. The client must obtain an EJB object reference to a stateful session bean to ensure that it is always accessing the same instance of the bean.

WebSphere Application Server currently supports the cloning of stateful session bean home objects among multiple application servers. However, it does not support the cloning of a specific instance of a stateful session bean. Each instance of a particular stateful session bean can exist in just one application server and can be accessed only by directing requests to that particular application server, as seen in Figure 17-11. State information for a stateful session bean cannot be maintained across multiple application server clones.

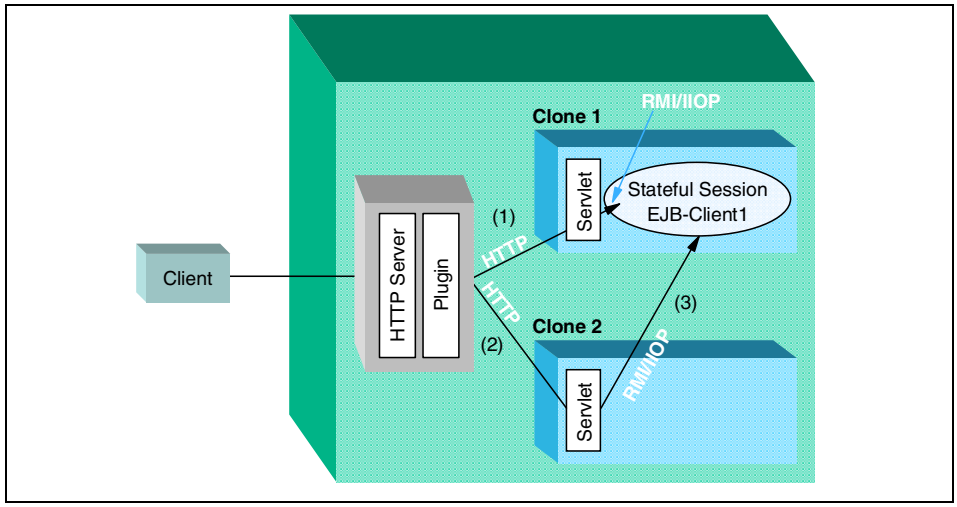


Figure 17-11 Stateful session beans

Note: Even though stateful session beans are not workload managed themselves, WLM can still be achieved when the homes are evenly distributed. It is only after the bean is created that everything will be done on the same clone.

17.5.6 Entity beans

An entity bean represents persistent data. Most external clients access entity beans by using session beans, but it is possible for an external client to access an entity bean directly. The information contained in an entity bean is not usually associated with a session or with the handling of one client request or series of client requests. However, it is common for a client to make a succession of requests targeted at the same entity bean instance. It is also possible for more than one client to independently access the same entity bean instance within a short time interval. The state of an entity bean must therefore be kept consistent across multiple client requests.

For entity beans, the concept of a session is replaced by the concept of a transaction. An entity bean is instantiated in a container for the duration of the client transaction in which it participates. All subsequent accesses to that entity bean within the context of that transaction are performed against that instance of the bean in that particular container. The container needs to maintain state information only within the context of that transaction. The workload management service uses the concept of *transaction affinity* (as defined in 17.5.7, “EJB server selection policy” on page 621) to direct client requests. After an EJB server entity bean is selected, client requests are directed towards it for the duration of the transaction.

Between transactions, the state of the entity bean can be cached. The WebSphere V4.0 EJB container supports option A, option B, and option C caching.

- With option A caching, WebSphere Application Server assumes that the entity bean is used within a single container. Clients of that bean must direct their requests to the bean instance within that container. The entity bean has exclusive access to the underlying database, which means that the bean cannot be cloned or participate in workload management if option A caching is used.
- With option B caching, the bean instance remains active (so it is not guaranteed to be passivated at the end of each transaction), but it is always reloaded from the database at the start of each transaction. A client can attempt to access the bean and start a new transaction on any container that has been configured to host that bean, as seen in Figure 17-12 on page 621.
- With option C caching (the default), the entity bean is always reloaded from the database at the start of each transaction and passivated at the end of each transaction. A client can attempt to access the bean and start a new transaction on any container that has been configured to host that bean, as seen in Figure 17-12 on page 621.

Entity beans can participate in workload management as long as the server reloads the data into the bean at the start of each transaction, assuming that transactional affinity is in place. Guaranteed passivation at the end of each transaction is not a requirement for a bean to participate in workload management. Hence, option B and option C caching are both compatible with workload management, but option A caching is not.

Table 17-1 provides a summary of the EJB caching options.

Table 17-1 EJB caching options

Option	Activate bean instance	Call ejbLoad() on bean instance
A	Once	Once

Option	Activate bean instance	Call ejbLoad() on bean instance
B	Once	At start of transaction
C	At start of transaction	At start of transaction

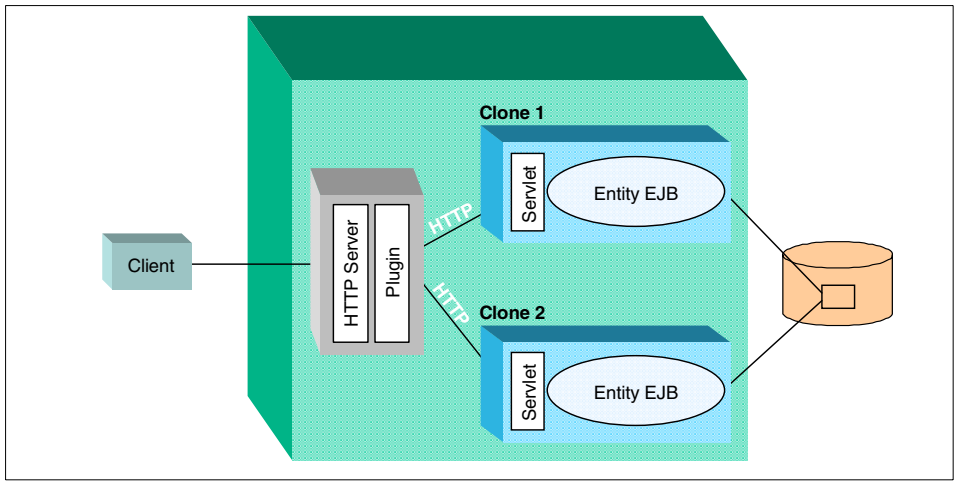


Figure 17-12 Cloning entity EJBs with option B and option C caching

See 18.11.1, “EJB container caching option for entity beans” on page 673 for details on how to set the EJB caching option.

17.5.7 EJB server selection policy

An EJB server selection policy defines how clients (such as servlets, stand-alone Java clients, or other EJBs) choose among EJB server clones (instances). EJS workload management offers the following selection policies:

- **Random**. EJB server instances are selected randomly from within the group of clones associated with the server group.
- **Round Robin**. An EJB server instance is initially selected at random from an ordered list of clones associated with the server group. Other EJB server instances are selected from the ordered list in turn, until the initially selected EJB server is selected again. The sequence is repeated.

  If a particular EJB server instance is stopped or otherwise unavailable, that instance is skipped (no attempt is made to select it) until it can be verified as being back in service.
- **Random Prefer Local**. EJB server instances on the same host as the client are selected according to the random server selection policy. If no local EJB

server instances are available, server instances on remote hosts are selected according to the random server selection policy.

- **Round Robin Prefer Local (default)**. EJB server instances on the same host as the client are selected according to the round-robin server selection policy. If no local EJB server instances are available, server instances on remote hosts are selected according to the round-robin server selection policy as shown in Figure 17-13. Note the solid lines for Prefer Local and the dotted lines for Round Robin/Random.

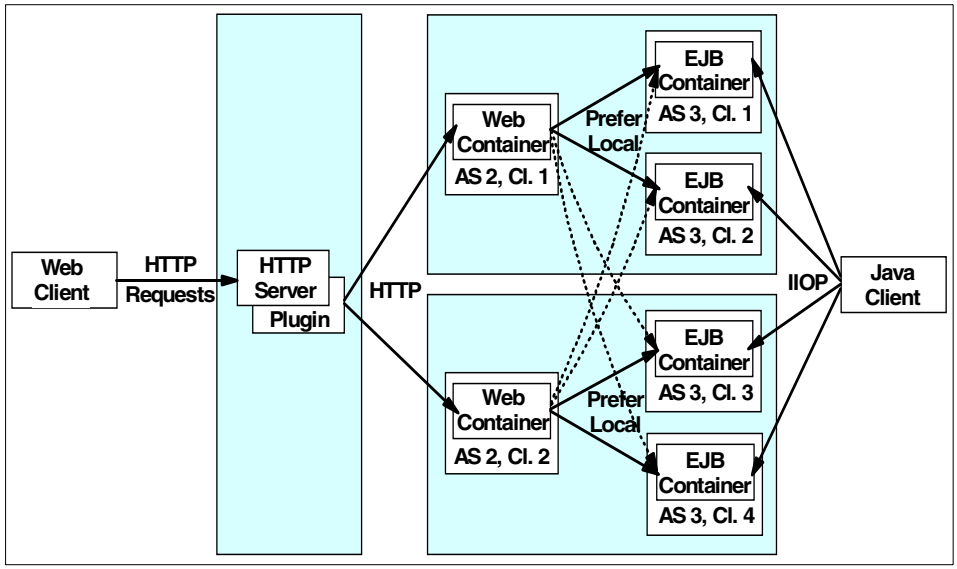


Figure 17-13 Round Robin Prefer Local

These policies, however, are overridden by *transaction affinity* and *process affinity.*

- **Transaction affinity**. Within a transaction, the first time an EJB server (entity bean) is picked, the prevailing selection policy for the server group is applied. After the EJB server is selected, it remains bound for the duration of the transaction.
- **Process affinity**. Where the Web container and the EJB container are configured in the same application server, the Web container will never route EJB requests to a container in a separate application server, as shown in Figure 17-14.

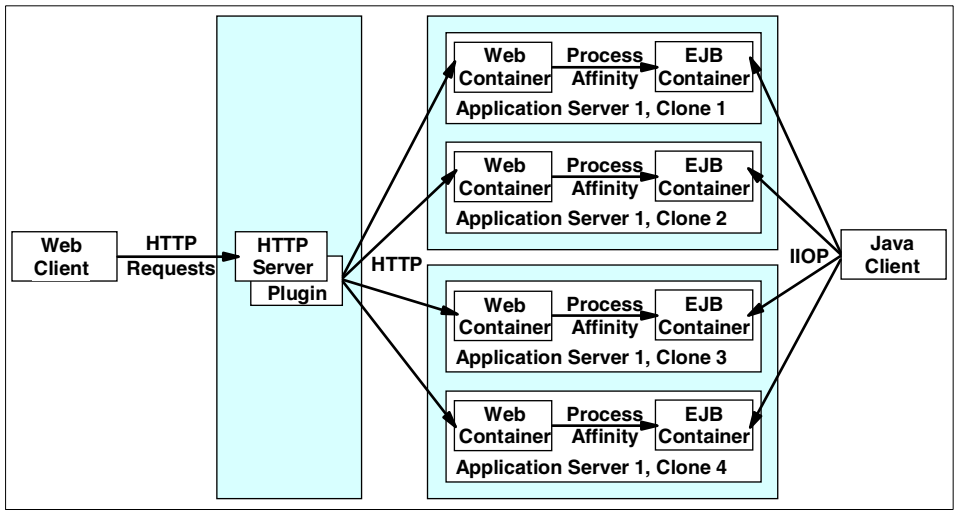


Figure 17-14 Process affinity

Note: Process affinity applies to servlet and EJB clients only. It does not affect stand-alone Java clients, because their EJB requests always come from outside of the application server process.

The choice of policy depends very much on your environment.

Round Robin and Random should give the same results. Round Robin is more systematic (meaning you know what the next server is, while using Random you are depending on an algorithm to determine that).

The Prefer Local option is based on topology and pre-production test results. Naturally the local host call will be quicker and if you can put your clients (usually Web containers) on the same machines as your servers, then the Prefer Local option is a good thing. If you have clients only on a subset of your machines, then you will need to analyze the load distribution, since you will have client requests coming from remote machines as well as the local machine.

17.5.8 EJS WLM failover processing of exceptions

The workload management service can throw the following exceptions if it encounters problems.

- `org.omg.CORBA.NO_IMPLEMENT`. This exception is thrown if the workload management service cannot contact any of the EJB servers that participate in workload management.

- `org.omg.CORBA.INTERNAL`. This exception is thrown when an internal software failure occurs.
- `org.omg.CORBA.COMM_FAILURE`. This exception is thrown by the ORB when a communications failure occurs. Any current transactions are rolled back and non-transactional requests are redone.
- `org.omg.CORBA.NO_RESPONSE`. This exception is thrown by the ORB when a communication failure occurs.

The WebSphere Application Server client can catch these exceptions and then implement its own strategies to handle the situation.

The workload management service uses the following failover strategies:

- If the workload management service cannot contact a clone, it automatically redirects the request to another clone, providing automatic failover.
- If the application throws an exception, automatic failover does not occur. The workload management service does not retry the request because it cannot know whether the request was completed.
- If an `org.omg.CORBA.NO_IMPLEMENT` exception is thrown, the workload management service has attempted repeatedly to contact all the servers without success. Workload management resumes when servers become available again.
- If an `org.omg.CORBA.INTERNAL` exception is thrown, the workload management service is no longer operating properly and no failover occurs.
- If the `org.omg.CORBA.COMM_FAILURE` or `org.omg.CORBA.NO_RESPONSE` exceptions are thrown, their return value determine whether automatic failover occurs:
  - With a `COMPLETION_STATUS` of `COMPLETED_NO`, automatic failover occurs because the request was not completed.
  - With a `COMPLETION_STATUS` of `COMPLETED_YES`, failover does not occur because the request was successfully completed.
  - With a `COMPLETION_STATUS` of `MAYBE` automatic failover does not occur. The workload management service cannot verify whether the request was completed. In this situation, the client application must anticipate a failure and retry the request. The workload management service then attempts to direct the request to a surviving clone.

17.5.9 WLM for Java clients

Remote Java clients that are not run on a machine with a WebSphere administrative server must have the following properties set on the Java command line:

For Java thin clients:

```
-Dcom.ibm.ejs.wlm.BootstrapNode=<admin_server_node>
-Dcom.ibm.CORBA.BootstrapHost=<admin_server_node>
-Dcom.ibm.CORBA.BootstrapPort=<admin_server_bootstrap_port (defaults to
900)>
```

where the `admin_server_node` is the host name of the machine where the WebSphere administrative server is located. For example:

```
java -Dcom.ibm.ejs.wlm.BootstrapNode=itsohost
-Dcom.ibm.CORBA.BootstrapHost=itsohost -Dcom.ibm.CORBA.BootstrapPort=900
WlmApp
```

For J2EE clients use:

```
launchClient <user_app.ear> -CCBootstrapHost=<admin_server_node>
-CCBootstrapPort=<admin_server_bootstrap_port (defaults to 900)>
```

where the `admin_server_node` is the host name of the machine where the WebSphere administrative server is located. For example:

```
launchClient webbank.ear -CCBootstrapHost=itsohost -CCBootstrapPort=900
```

17.6 Administrative server WLM

Administrative servers can be configured to provide failover capability, improving the availability of administrative and naming services, as shown in Figure 17-15 on page 626.

This behavior is enabled by default in the <WAS_HOME>/bin/admin.config file:

```
# Enable WLM of AdminServers
com.ibm.ejs.sm.adminServer.wlm=true
```

To achieve failover capabilities, you must enable administrative servers failover for every node in a domain.

Important: The administrative server only participates in the failover portion of workload management. Administrative server requests are not distributed based on a WLM selection policy. The only time requests are redirected to a different administrative server is in the failover case.

The following conditions apply:

1. Once a client has established a session with one of the administrative servers, it will stay with the same administrative server for the duration of the client session, unless the administrative server becomes unavailable, in

which case one of the other administrative servers will provide failover capabilities.

2. If there is an administrative server in the machine where the client runs, the client will always go to its local administrative server. Only when that administrative server is unavailable will it go to a remote administrative server. It will stay on the same remote administrative server for the duration of the client session, so it will not bounce to other remote administrative servers. This is true even if the local administrative server comes back up, since the client will not go back to the local administrative server.
3. There is a single point of failure on the first call a client makes to a domain. The client has to make a bootstrap call, which is usually an InitialContext call. If that call is successful, then the client will be able to fail over to other administrative servers because the administrative server group information is returned on the InitialContext call.

Requests can come from EJB clients and from the WebSphere Administrative Console.

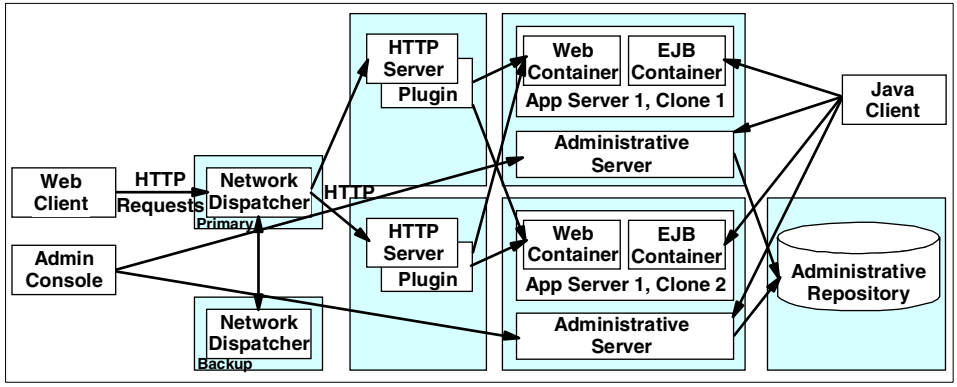


Figure 17-15 Administrative server WLM - Network Dispatcher used to distribute load between Web servers

Note: Workload management must be enabled or disabled for all administrative servers in a domain.

When an administrative server participates in workload management, an exception is thrown if the administrative server fails during an administrative task. Subsequent requests are redirected to the other administrative servers in the domain, minimizing the disruption to administrative operations.

For example, a command issued through the WebSphere Administrative Console can fail if an administrative server goes offline while the command is being executed. If workload management is enabled, any subsequent attempts to execute the command are redirected to another administrative server. This allows the command to be successfully reissued, possibly with a delay for the initial redirection. Subsequent requests will not be noticeably slower. The original administrative server will pick up its share of administrative requests when it comes back online.

There are two ways to enable workload management for administrative servers:

- By setting the following property in the <WAS_HOME>/bin/admin.config file (recommended):

  ```
  com.ibm.ejs.sm.adminServer.wlm=true
  ```

- By specifying the `-wlm` argument when starting an administrative server from the command line. For instance:

  ```
  java com.ibm.ejs.sm.server.Adminserver -wlm...
  ```

By default, workload management is enabled when you start the administrative servers in a domain.

To discontinue workload management, stop all administrative servers in the domain and restart them with workload management disabled.

Disable workload management by setting the following property in the admin.config file:

```
com.ibm.ejs.sm.adminServer.wlm=false
```

17.7 Using WLM: a sample procedure

In this section we take you through a sample procedure that shows how you can implement workload management by cloning an application server. This procedure consists of the following steps:

1. First decide which application server you are going to clone. When the application is fully and correctly configured you are ready to start.
2. Create a server group from your application server. The original application server instance becomes a clone and it is administered through the server group.
3. Select the workload management selection policy (used for EJS workload management).
4. Create one or more clones of the server group.

5. Start all of the application servers by starting the server group.
6. Regenerate the Web server plug-in configuration file.

Workload management automatically begins when you start the clones on the application server.

Note: You can only create server groups of entire application servers. Only application servers can be cloned, not components.

17.7.1 Create server group from application server

In this section we show how to create a server group from an application server:

1. Configure the application server on which you want to base the server group. Make sure you also configure its contents, such as Web and EJB modules.
2. Test the application server to make sure it is working properly before creating the server group based on it.
3. From the administrative console, stop the application server you want to clone by selecting the application server and clicking the **Stop** button.
4. Right-click the application server that you want to clone and select **Create Server Group**, as shown in Figure 17-16. This will display the Server Group Properties window.

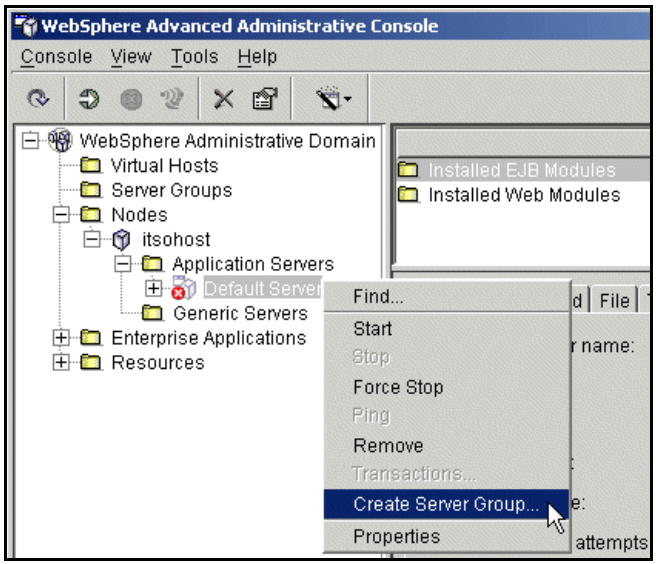


Figure 17-16 Create Server Group

5. Use the Server Group Properties window to specify settings for the server group. Specify the Server Group Name, as seen in Figure 17-17.

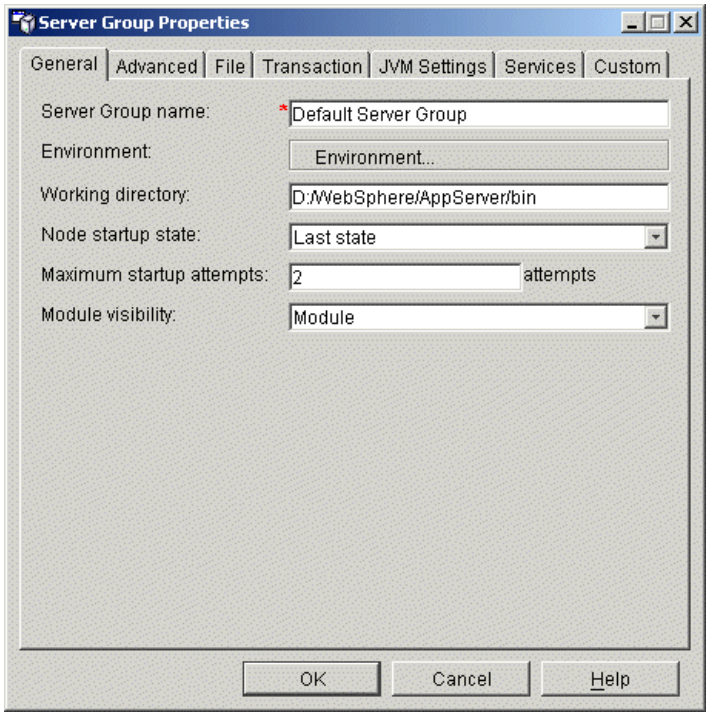


Figure 17-17 Server Group Properties

6. From the Advanced tab select the EJS WLM Selection policy. The default is Round Robin Prefer Local (Figure 17-18).

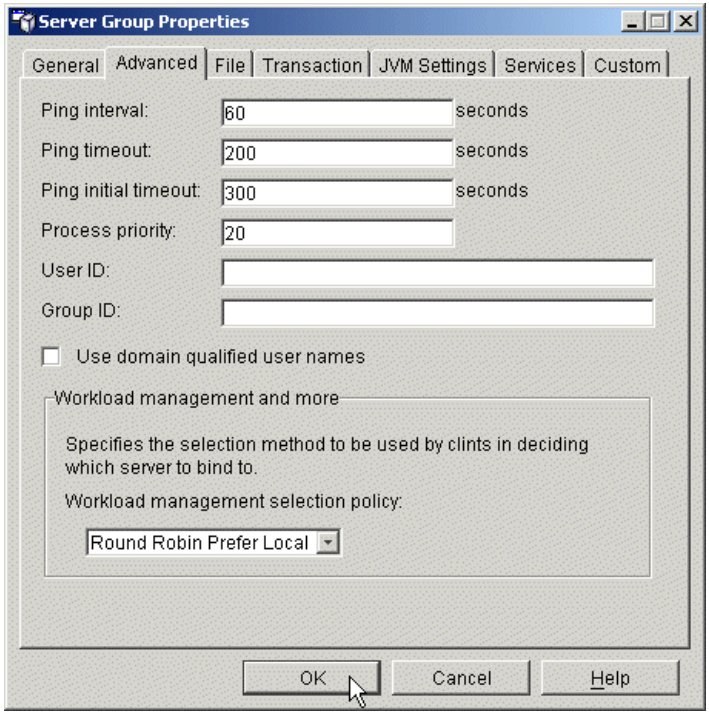


Figure 17-18 Selection of EJS WLM policy

7. Click the **OK** button to create the server group. Notice that the original application server has become the initial clone in the new server group.

   Notice also that the Web and EJB modules now appear on the server group rather than the original application server.

17.7.2 Create clone from server group

After creating a server group, you can create clones from the server group. The steps below show how to create a vertical clone (same machine).

1. Double click to expand your server groups.
2. Right-click the server group that you want to clone and select **New... -> Clone...** from the pop-up menu (Figure 17-19).

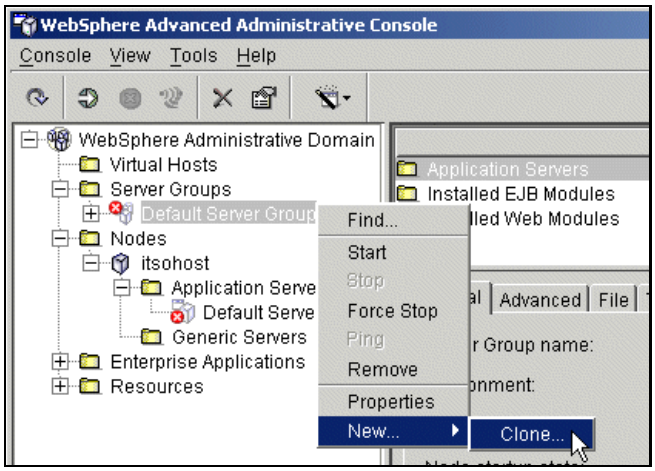


Figure 17-19 Create clone from server group

3. In the Create Clone window, shown in Figure 17-20, specify a name for the clone, and select the node on which the clone will run. It could be any node.

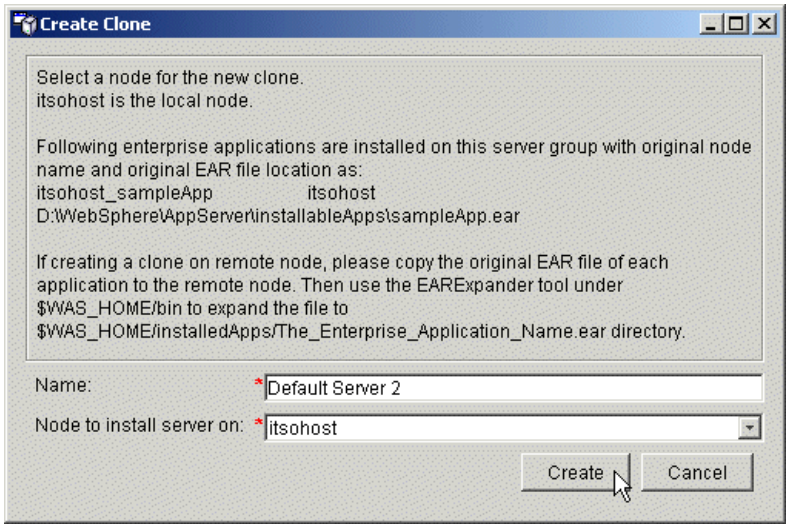


Figure 17-20 Create Clone

4. Click the **Create** button to create the clone.
5. Check the HTTP transport port on the Web Container Service to avoid port conflicts. Each application server has an embedded HTTP server, which by default is listening on port 9080. You cannot have more than one application server listening on port 9080.

a. Right-click the application server clone and select **Properties** from the pop-up menu, as seen in Figure 17-21.

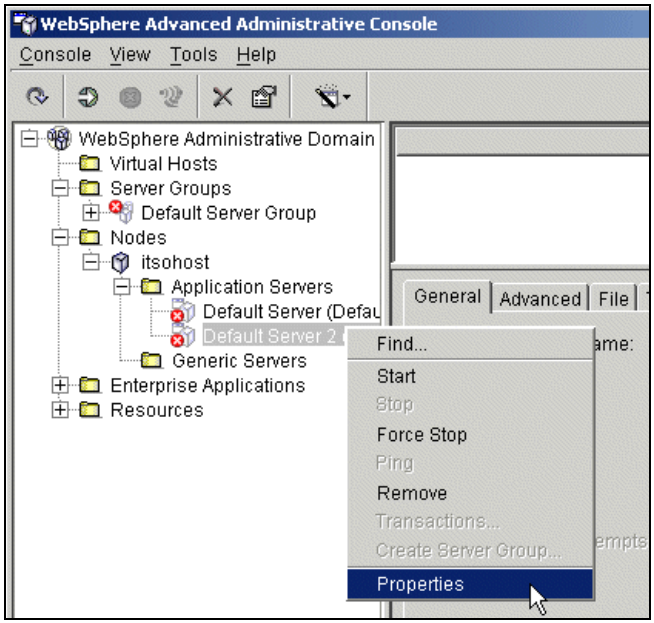


Figure 17-21 Clone Properties

b. From the Services tab, select **Web Container Service** and click **Edit Properties...**, as seen in Figure 17-22:

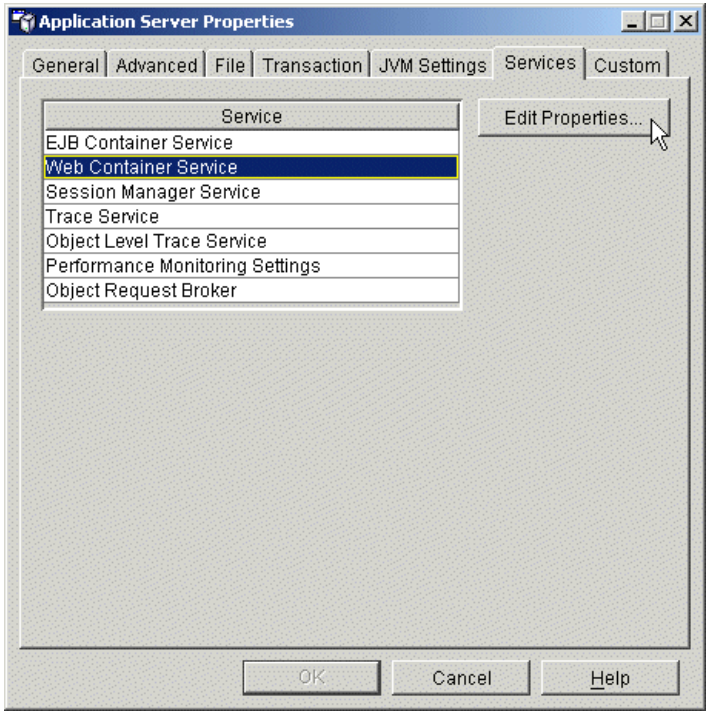


Figure 17-22 Web Container Service

c. From the Transport tab, shown in Figure 17-23 on page 634, select the only entry and click **Edit**. Change the default port if necessary. Click **OK**.

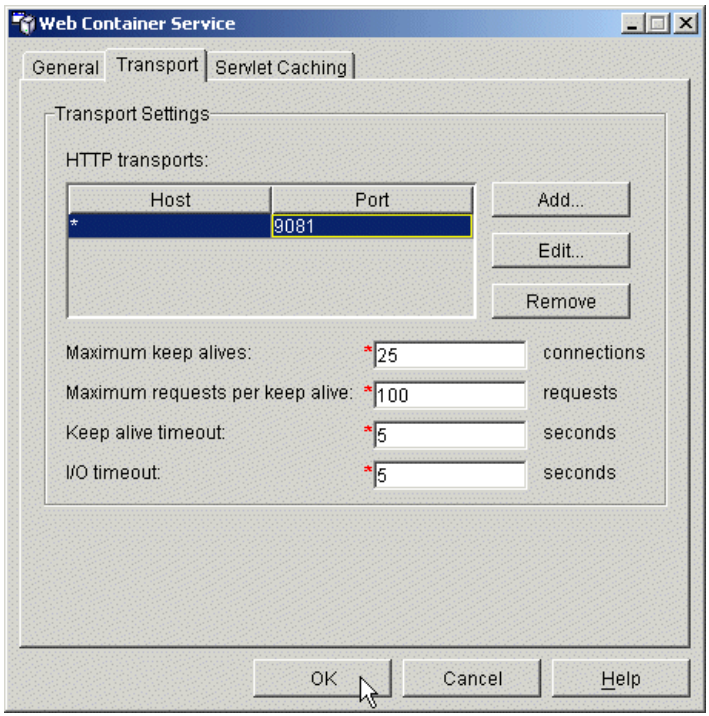


Figure 17-23 HTTP listener port associated with the Web container

6. Start the new server group by right-clicking it and selecting **Start** from the pop-up menu, as shown in Figure 17-24. This will start all the clones in the server group.

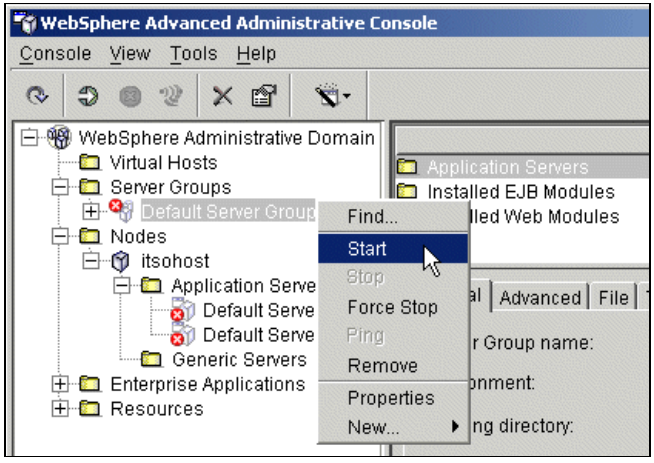


Figure 17-24 Start the server group

7. Regenerate the Web server plug-in configuration file to recognize the new configuration. When the WebSphere configuration changes, the Web server plug-in configuration information must be updated. Right-click the **Node** and select **Regen Webserver Plugin**. Alternatively run the command line script, GenPluginCfg.

   The default plug-in configuration file <WAS_HOME>\config\plugin-cfg.xml, as shown in Example 17-1, shows the clone entries for each of the server groups. Note the presence of a Server CloneID attribute. With this defined the plug-in can enforce HttpSession affinity. Simply removing the CloneID attribute/value pairs from the Server element will disable HttpSession affinity.

   See Chapter 15, “Configuring session management” on page 513 for more information on HttpSession affinity and HttpSession persistence.

Example 17-1 plugin-cfg.xml file

```
<ServerGroup Name="MyBankServerGroup">
   <Server CloneID="svgqgub2" Name="MyBank">
      <Transport Hostname="itsohost" Port="9090" Protocol="http"/>
   </Server>
   <Server CloneID="svgtncsn" Name="MyBankFirstClone">
      <Transport Hostname="itsohost" Port="9087" Protocol="http"/>
   </Server>
</ServerGroup>
```

If you have cloned the Default Server application server as described here, and the WebSphere Application Server examples are installed in the Default Server, then you can try out WLM using the BeenThere servlet WLM demonstration.

Use the following URL to access this demonstration:

```
http://<your.server>/webapp/examples/BeenThere
```

This demonstration shows workload management in action. You can see that the Servlet Server Name for each access to BeenThere changes between Default Server and Default Server 2, as shown in Figure 17-25 on page 636 and Figure 17-26 on page 636.

Click the **README** button to find out more about the BeenThere demonstration.

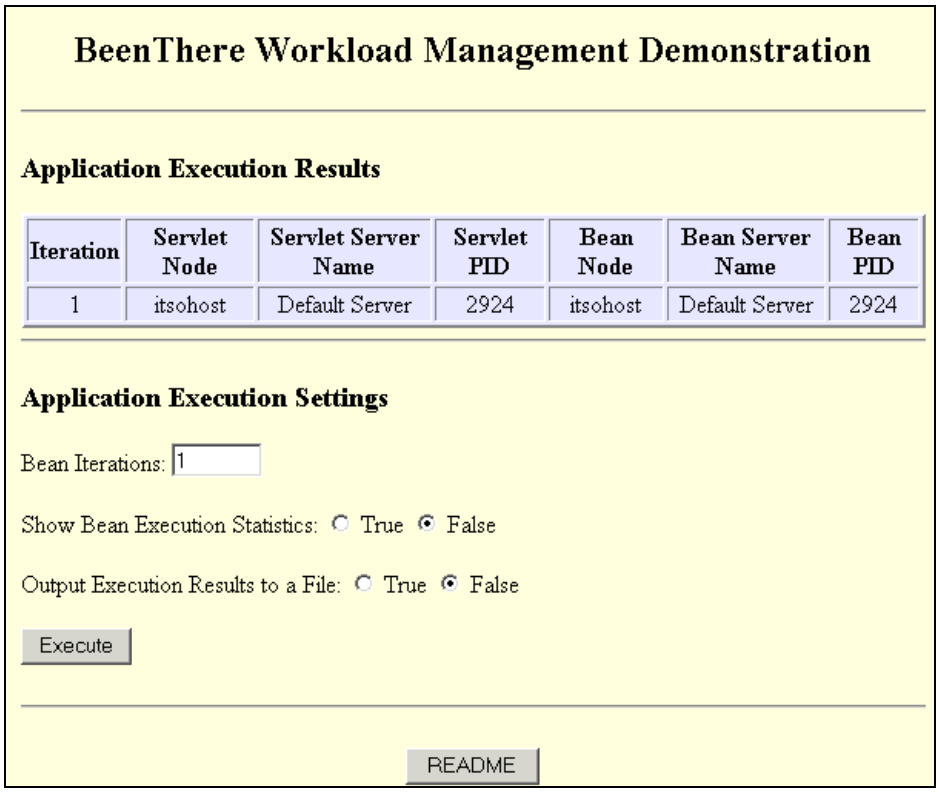


Figure 17-25 First BeenThere access

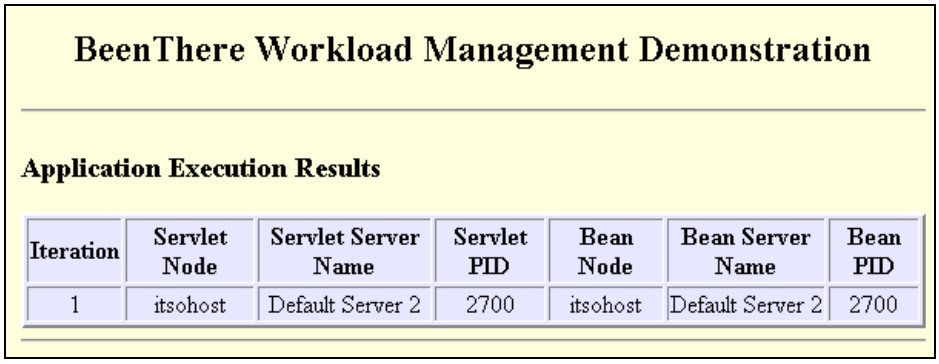


Figure 17-26 Next BeenThere access

The installation of an enterprise application to the server group installs the application on only one machine, typically the local machine (the one that the administrative console is connected to).

If you are planning to clone on another node, then you need to install the application on all the other nodes in the cluster. This is accomplished with the EARExpander (bat/sh) script, located in <WAS_HOME>/bin. Make sure you have configured a JDBC driver on that node.

You can create a clone on any node in the administrative domain.

Tip: If you want to test WLM for both “plug-in WLM” (servlet requests) and “EJS WLM” (EJB requests), you can use the BeenThere servlet that gets installed with the SampleApp. View the readme file that comes with it, found at `http://localhost/webapp/examples/beenthere.html`.

17.8 TCP/IP spraying

Though it is not explained in this book, *Network Dispatcher*, which is part of *WebSphere Edge Server*, can also be used for workload management. Placed in front of Web servers, client requests come into the Network Dispatcher machine, which then forwards the requests to the HTTP servers using one of several routing algorithms. To the client, Network Dispatcher looks like a Web server. Network Dispatcher can be configured for high availability, as shown in Figure 17-27.

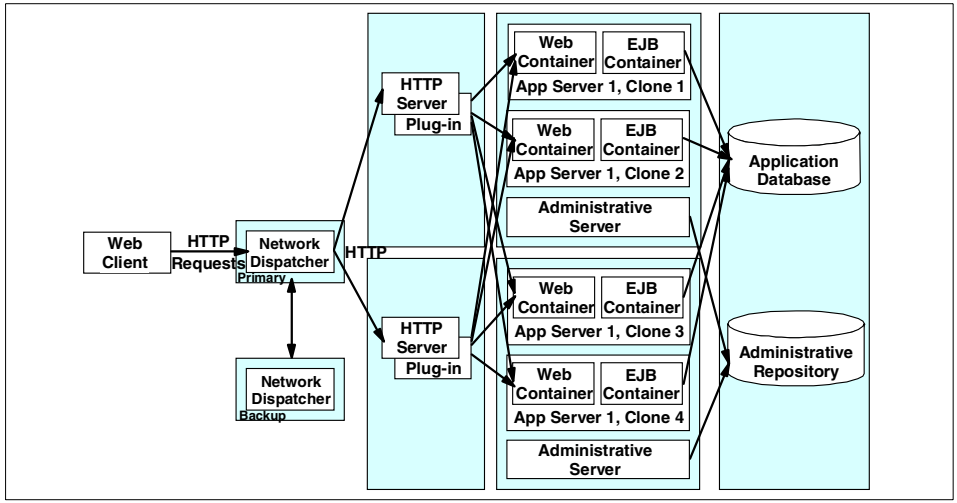


Figure 17-27 Combining horizontal and vertical scaling with Network Dispatcher

The only single point of failure in a WebSphere domain/cluster like the one shown in Figure 17-27 on page 637 is the database server where the WebSphere administrative database resides. It is on the database server that any hardware-based high availability (HA) solutions such as HACMP, Sun Cluster, or MC/ServiceGuard should be configured.

18

Packaging an application

In this chapter, we provide practical steps to packaging a J2EE application. If you are not familiar with J2EE applications packaging, we strongly recommend you read Chapter 3, “The Java 2 platform” on page 29 first.

We start by describing the steps to package applications using the Application Assembly Tool (AAT). First, we show how to package individual EJB, Web, and client modules, and then assemble them in an enterprise application. We then cover advanced packaging options which are common to EJB, Web, and client modules such as EJB references, or resources references. Finally, we cover all IBM extensions to the standard J2EE deployment descriptors, such as EJB caching options.

This chapter only covers how to package an application using the tools provided with the WebSphere Application Server product. Application development tools, such as IBM VisualAge for Java or IBM WebSphere Studio Application Developer are also capable of packaging J2EE applications. Please refer to redbook *WebSphere Version 4 Application Development Handbook*, SG24-6134 for more details.

18.1 Application Assembly Tool overview

The Application Assembly Tool (AAT) is a tool delivered with WebSphere Application Server that will help you package your application according to the J2EE specification. As shown in Figure 18-1 below, it lets you create EJB modules, Web modules, and application client modules. You can start the AAT from the command line using the **assembly.bat/assembly.sh** commands. You can find more information on the AAT in the InfoCenter.

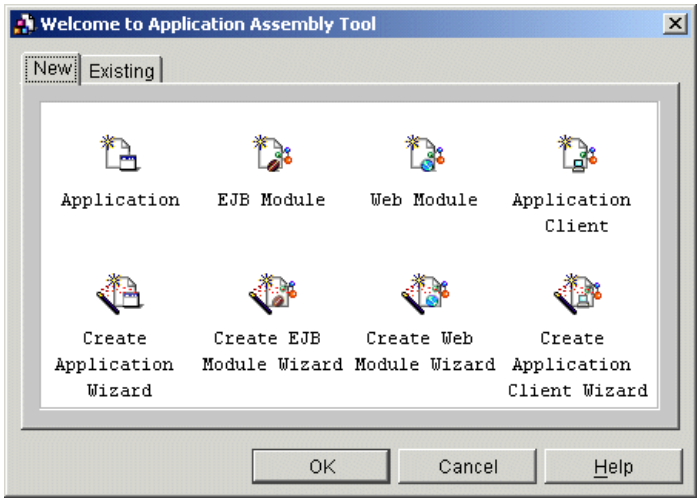


Figure 18-1 Application Assembly Tool welcome window

In this chapter, we explain how to create an EJB module, assuming we only have the necessary implementation classes. Then, we create a Web module, then an application client module. Finally, we use the Create Application wizard to assemble those modules in an enterprise application (since it requires the modules to exist). This is probably not the way you would do it in practice, but it allows us to cover all the wizards.

Note: You can have multiple Web modules, EJB modules, or application files opened at the same time in the AAT. Use the Window menu to switch between them.

18.2 Webbank application overview

The Webbank application is a sample that uses servlets, JSP pages, and enterprise beans. It also comes with stand-alone clients. We deliberately chose a simple application to illustrate the concepts.

Note: It is worth mentioning that the Webbank application has originally been developed for teaching purposes, mainly to demonstrate EJB transactional behavior. Some design choices (such as implementing the Transfer bean as a stateful EJB or using bean-managed transactions) have been governed by education purposes. As an example, the Transfer bean has no valid reason to be a stateful EJB.

Using the stand-alone clients or the Web interface, you can make transfers between a customer account and a branch account. Entity CMP beans have been used to access the customer and branch accounts, which are stored in a relational database. The Transfer session bean can be used to make transfers between accounts, while the Consultation session bean can be used to obtain the accounts balance. The sample architecture is shown in Figure 18-2. Instructions for finding the Webbank sample application are in Appendix D, “Additional material” on page 1083.

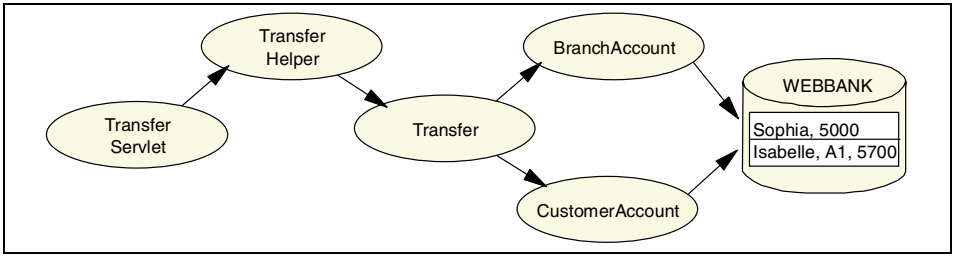


Figure 18-2 Webbank Application Architecture

For the sake of demonstrating how to package an application from scratch, we assume the following:

1. The enterprise beans have been developed, and you only have the class files available on the file system, that is, the J2EE deployment descriptor (ejb-jar.xml) has not been created. All enterprise beans class files are stored under the webbank\ejbModule folder.
2. Servlet code is stored under the webbank\webModule folder, along with the JSP files and the static files of the application (HTML and images).
3. Common classes, which are used by either enterprise beans or servlets, are packaged in a webbankCommon.jar file, stored in the webbank folder.
4. Finally, the stand-alone client class files are stored in the webbank\appClientModule folder.

The sections below assume that you have unpacked the webbank.zip file under D:\. Instructions for finding webbank.zip are in Appendix D, “Additional material” on page 1083. Using the Application Assembly Tool (AAT), you will now create the deployment descriptors of the Webbank enterprise beans and package them in a JAR file.

18.3 Packaging an EJB module

The AAT lets you group enterprise beans classes in a so-called EJB module. An EJB module is represented by a JAR file that contains the enterprise bean classes/interfaces and the bean deployment descriptors. The standard J2EE deployment descriptor is stored in an ejb-jar.xml file. IBM deployment descriptor extensions are stored in an ibm-ejb-jar-ext.xmi file. In this chapter, we refer to this file as the IBM extensions file. Additionally, bindings definitions (such as enterprise bean JNDI names) are stored in an ibm-ejb-jar-bnd.xmi file. In this chapter, we refer to this file as the IBM bindings file. Bindings can also be defined at deployment time using the administrative console. This topic is covered in Chapter 19, “Deploying an application” on page 687.

The AAT lets you manipulate these files using a GUI interface. Most of the time, you will not change these files manually. We recommend that you always use the AAT to update these files. However, we have documented the contents of the generated files to help you maintain these files using a text editor.

18.3.1 Creating the EJB module

To create an EJB module, you can either select the EJB Module option in the AAT welcome window when it starts up, or select **File** -> **New** -> **EJB Module** from the AAT main menu. An EJB module with a default name (New_EJB_JarX.jar) appears on the left-hand side of the AAT window.

As shown in Figure 18-3, the following General properties can be set for an EJB module:

- Display name

  The name of the EJB module, for use in any GUI tool that manipulates the EJB module, such as the AAT or the administrative console.

- Description

  A meaningful description of the contents of the module, such as EJBs for the Webbank application.

- EJB client jar

  The JAR file containing the classes that a client application would need.

- Classpath

  The list of JAR/ZIP files and classes that the EJBs contained in the EJB module depend on. If you need to specify multiple JAR files, they must separated by a *space*. Setting this property adds a Class-Path entry to the EJB JAR manifest file, like this:

```
Manifest-Version: 1.0
Class-Path: webbankCommon.jar
```

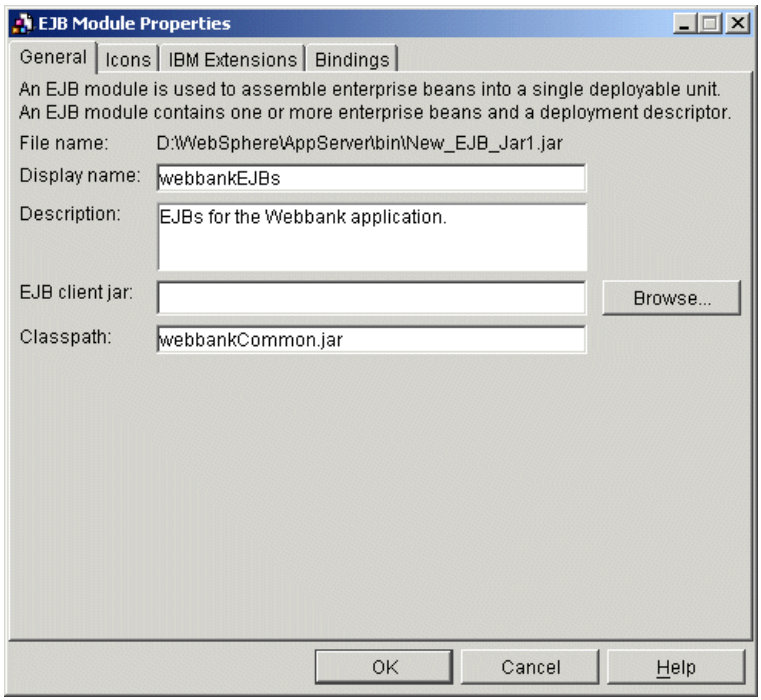


Figure 18-3 Setting the EJB module properties

Detailed information on the manifest Class-Path entry

The Webbank enterprise beans use classes from the webbankCommon.jar file. You *must* reference this JAR file to successfully:

1. Use the Java reflection mechanism on the EJB classes. AAT uses Java reflection to list all methods from the remote and home interfaces. If one of the methods throws an exception of type itso.webbank.common.InvalidAmount, then AAT cannot reflect this method unless the InvalidAmount class is available on the classpath.
2. Find the webbankCommon.jar file at runtime. By setting the Classpath property, you basically say that your EJB module depends on this webbankCommon.jar, which will then be loaded automatically. Please refer to

18.14.1, “Where should common classes go?” on page 683 for more details on how WebSphere finds and loads classes at runtime.

Tip: You will not see any errors appearing in the AAT GUI if classes cannot be found. If you select the method extensions entry, and cannot see your methods in there, have a look at the command window from which the AAT was started. You should then see errors similar to this one:

```
Could not reflect methods for itso.webbank.ejbs.CustomerAccount
because one of the methods references a type that could not be
loaded. itso/webbank/common/InvalidAmount.
```

You must reference the webbankCommon.jar file using a relative path (absolute paths, such as D:\webbank\webbankCommon.jar are ignored). You must make sure the file is accessible to the AAT, by including it in the appropriate place in the enterprise application module, or by having it in the AAT working directory. The AAT working directory is the directory where the module currently being edited is saved. Since we don’t have an enterprise application module yet, we will be using the AAT working directory option, and will save our EJB module in the same directory as webbankCommon.jar.

18.3.2 Adding files to the EJB module

Before adding any enterprise beans to the EJB module, you need to add all the files you need to the module, that is, the contents of the ejbModule folder. An ejbModule contains all the class files for the EJB plus any dependent classes.

To add the EJB class files to the EJB module:

1. Right-click the **Files** entry, and select **Add Files** from the pop-up menu.
2. Click **Browse...** and select the D:\webbank\ejbModule folder. It should now appear in the left pane.

Important: You have to select the *top* of the package hierarchy. In our sample, the EJB classes are located in the itso.webbank.ejbs package, which is represented by the itso/webbank/ejbs folder hierarchy on the file system. If you were selecting the itso folder instead of the ejbModule one, the AAT would think that it is the top of the hierarchy, and conclude that the package name is webbank.ejbs, which is obviously wrong. If you select the ejbModule folder, then the correct itso.webbank.ejbs package name is used for all classes.

3. Extend the tree view under the D:\webbank\ejbModule folder until you reach the ejbs folder, and select it.

4. Select all classes in the right pane, and click **Add**.
5. Click **OK**.

18.3.3 Adding a session bean to the EJB module

Now that all necessary files are available, you can start adding enterprise beans to the EJB module. Follow these steps to add the Consultation bean to the Webbank EJB module:

1. Select the **Session Beans** entry, right-click, and choose **New** from the pop-up menu. A window similar to Figure 18-4 on page 646 appears.
2. Click **Browse...** next to the Home interface field. The window that opens lets you select the directory/JAR File on the file system where enterprise beans classes are located. It also shows you the list of files that have already been added to the EJB module. You should therefore see all the classes you have added to the EJB module in the previous step.
3. Expand the tree view under webbankEJBs.jar until you can see the ejbs folder.
4. Select the ConsultationHome.class file in the right pane, and click **OK**.

   As you can see, most of the required information (marked with a red star) has been automatically filled in. Since the AAT has been able to find the remote interface as well as the bean class, those fields are already filled. Similarly, the ejb-name field is filled in; this is mainly because we have respected some naming conventions. Usually, if your EJB name is Foo, then the remote interface is named Foo, the home interface is named FooHome, and the EJB class is named FooBean.
5. Optionally, you can provide a Display name for the Consultation bean. The name is used in any GUI tool that manipulates the EJB, such as the AAT or the administrative console. If this name is not provided, the ejb-name is used instead.
6. Next, you must select the session EJB type, in our case **Stateless**.
7. Finally, make sure that the transaction type is set to **Container**. This ensures that all transactions started when invoking methods on the enterprise bean are container-demarcated (handled by the EJB container).
8. Click **OK**.
9. Repeat steps 1 to 8 for the Transfer session EJB. You must set the Session type to **Stateful** and the Transaction type to **Bean** (the Transfer EJB manages the transaction demarcation).

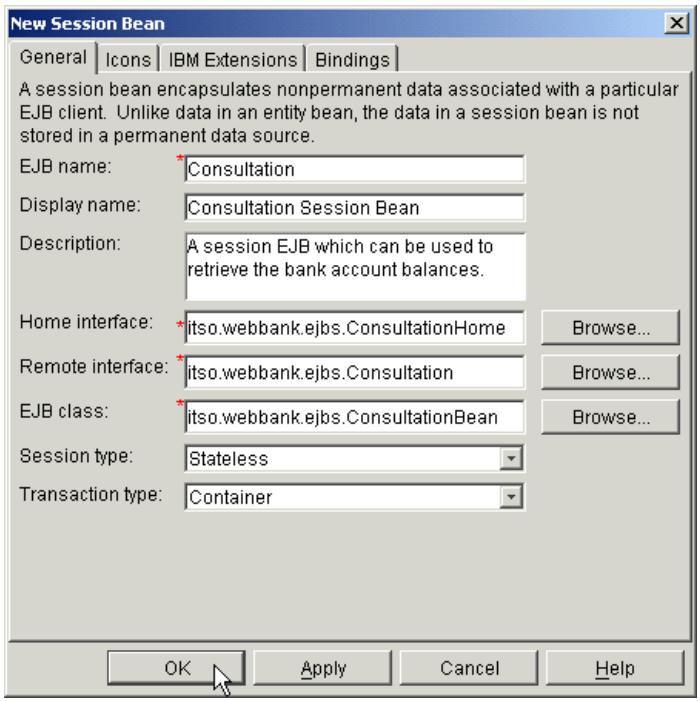


Figure 18-4 New Session Bean window

Now we need to save the Webbank EJB module in the same directory as webbankCommon.jar. Save the webbankEJBs.jar as follows:

1. Select **File** -> **Save** from the main menu.
2. Select the directory where you want to save this JAR, and enter a name for it. For our example, we are using D:\webbank\webbankEJBs.jar.

> **Important:** To enable AAT to reflect EJB methods, you must save the Webbank EJB module in the same directory as webbankCommon.jar, which is D:\webbank in our case. See “Detailed information on the manifest Class-Path entry” on page 643.

3. Click **Save**.

Table 18-1 contains all the session EJB settings that can be set from the general page, as well as their corresponding entry in the ejb-jar.xml file. Additional properties (such as environment entries or EJB references) are described later in this chapter.

Table 18-1 Session bean settings

Property name	Details
Generic enterprise beans settings	
EJB name	The name of the EJB, such as "Consultation". This name must be unique within the EJB module. This corresponds to the <ejb-name> entry.
Display name	This name is used in the AAT GUI, such as "Consultation Session Bean". This corresponds to the <display-name> entry.
Description	A meaningful description of this EJB, such as “A session EJB which can be used to retrieve the bank account balances.” This corresponds to the <description> entry.
Home interface	The class file that contains the enterprise bean home interface, such as "itso.webbank.ejbs.ConsultationHome". This corresponds to the <home> entry.
Remote interface	The class file that contains the enterprise bean remote interface, such as "itso.webbank.ejbs.Consultation". This corresponds to the <remote> entry.
EJB class	The class file that contains the definition of the enterprise bean class, such as "itso.webbank.ejbs.ConsultationBean". This corresponds to the <ejb-class> entry.
Small icon	A small image (gif/jpg) used by graphical tools. This corresponds to the <small-icon> entry.
Large icon	A larger image (gif/jpg) used by graphical tools. This corresponds to the <large-icon> entry.
Specific session bean settings	
Session type	Sets whether this session EJB is Stateless or Stateful. This corresponds to the <session-type> entry.
Transaction type	Sets whether transactions are handled by the Container or by the Bean. This corresponds to the <transaction-type> entry.

We have now added the session beans of our application to the EJB module. The next step is to add the entity beans.

18.3.4 Adding an entity bean to the EJB module

Adding entity beans to an EJB module is very similar to adding sessions beans. You simply need to supply different parameters. To add the BranchAccount entity bean to the EJB module:

1. Select the **Entity Beans** entry, right-click, and choose **New** -> **CMP** from the pop-up menu. By choosing this option, you set the `<persistence-type>` entry to `Container`. The alternative value is `Bean` (for Bean Managed Persistence).
2. Select the home interface, remote interface, as well as the enterprise bean class for the BranchAccount enterprise bean (located under itso/webbank/ejbs).
3. Additionally, select the primary key class. All the Webbank entity beans have specific primary key classes. Therefore, you must click the **Browse...** button, and select the BranchAccountKey class. If you had an enterprise bean with a primary key built from a single CMP field, you would type in the primary key field type instead.

Tip: If you do not know the type of your primary key class yet, just type in a dummy value (otherwise, AAT will not let you apply changes). Once you have defined all CMP fields, you will be able to choose a primary key field in the drop-down menu. This will set the correct value of the primary key class.

4. Since the BranchAccount bean has a primary key class, the Compound key value should be selected. If you have a single field key, you have first to describe the CMP fields of your entity bean to be able to select one of them in the drop-down list.
5. Click **OK** to save the changes.

Next, you must define the entity bean CMP fields, that is the persistent fields handled by the EJB container.

6. Right-click the **CMP Fields** entry under the BranchAccount enterprise bean, and select **New**.
7. Pick one CMP field name from the drop-down menu. This drop-down is filled with all the fields defined in the enterprise bean class, regardless of whether they should be persistent or not (AAT has no way to know this).
8. Repeat step 6 and 7 for all CMP fields.
9. Repeat steps 1 to 8 for the CustomerAccount enterprise bean.

10. Save the EJB module!

Table 18-2 Entity beans specific properties

Property name	Property explanation
Primary Key Class	The type of the entity bean primary key class. This should be used only when the primary key maps to multiple CMP fields. This corresponds to the `<prim-key-class>` entry.
Primary Key Field	The CMP field used as a primary key. If the primary key is composed of multiple fields, then CompoundKey is shown in this field. This corresponds to the `<prim-key>` entry.
Reentrant	Sets whether this entity bean is reentrant or not. Most of times, this value must be set to false. Be careful when using this flag, since as it allows loop-back calls (entity instance A calling instance B calling instance A again). This corresponds to the `<reentrant>` entry.

CMP field	A persistent field that the EJB container manages. Each CMP field has a *unique* field ID. This ID is used by the AAT or other database mapping tools to map the field to the actual column in a table. The ID is generated automatically by AAT from the enterprise name and the CMP field name. Each CMP field corresponds to an entry such as:
	`<cmp-field id="CustomerAccount_accountNumber">`
	`  <field-name>accountNumber</field-name>`
	`</cmp-field>`

So far, we have seen how to add enterprise beans to a module when you only have the classes available. This is a good learning exercise, but most application development tools are able to generate a basic deployment descriptor, which you can then customize with the AAT. The next section covers how to import packaged EJBs into an existing module.

18.3.5 Importing an EJB 1.1 JAR into an existing module

AAT lets you import enterprise beans that have already been packaged in an EJB JAR file. Importing an EJB into an existing module can be done by:

1. Selecting the Sessions Beans or Entity Beans entry, and selecting **Import** from the pop-up menu.
2. Click the **Browse...** button, and navigate to the JAR file that contains your enterprise beans and click **Open**. The top pane should now contain a list of beans (session *or* entity) found by AAT in the JAR file, as shown in Figure 18-5.

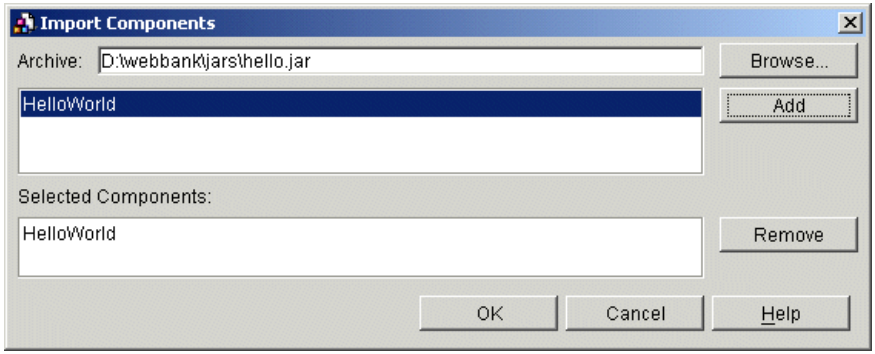


Figure 18-5 Importing the HelloWorld session EJB

3. Select one or multiple enterprise beans to import (hold down the Ctrl key to select multiple beans) and click the **Add** button.
4. Click **OK** to import the enterprise beans you have selected.

We have now reviewed the basic packaging settings for enterprise beans. The next section describes how to declare finder methods for entity beans.

18.3.6 Declaring finder methods for entity beans

Finder methods are declared on the entity bean home interface. The BranchAccount entity bean has a finder method called findByNameLike, which returns a list of all branch accounts for a specific branch. There is no standard way (in the Enterprise JavaBeans specification) to tell an EJB container that an SQL query should be executed when the finder method is invoked.

In WebSphere V3.5, you had to create a public interface (<BeanNameFinderHelper>) and declare a string that contained either the full query or part of the query you would want to execute, as shown in Example 18-1 on page 651. This method is still supported for WebSphere V4.0; however it is highly recommended that you use deployment descriptors extensions to describe finder methods.

Four types of finders are supported:

- User

  Normally, the EJB container is responsible for generating the code for a finder method based on information you provide, such as the FullQueryString or the WhereClauseString (see below). If you select the User mode, you tell the container that you are taking responsibility for writing this code. To support this, you will have to develop what is known as method custom finders. Please refer to the article *Implementing custom finder helpers for CMP entity beans* in the InfoCenter for more details.

- EJB SQL

  The EJB query language is the preferred way of defining finder methods in WebSphere V4.0, although using SQL is still supported. It is based on the EJB 2.0 specification, with some modifications required to work with the EJB 1.1 entity beans.

- Full Select

  This option lets you specify the full query string, such as `SELECT * FROM BRANCHACCOUNT WHERE BRANCHNAME='?'`. This option is only available for compatibility with previous editions of WebSphere. It must not be used for new development.

- Where Clause

  This option lets you specify the WHERE part of an SQL query, such as:

  ```
  T1.BRANCHNAME LIKE CONCAT(CONCAT('%',?),'%')
  ```

Example 18-1 Specify a finder method SQL query string in WebSphere V3.5

```
public interface BranchAccountBeanFinderHelper {
   public static final String findByNameLikeWhereClause =
      "T1.BRANCHNAME LIKE CONCAT(CONCAT('%',?),'%')";
}
```

Specify the finder method we need for the Webbank sample as follows:

1. In AAT, select the **Method Extensions** entry under the BranchAccount entity bean.
2. Right-click the **findByNameLike** home method in the method list, and select **Properties** from the pop-up menu.
3. Check the **Finder descriptor** option.
4. As shown in Figure 18-6, select the **Where clause** option and enter the following where clause:

   ```
   T1.BRANCHNAME LIKE CONCAT(CONCAT('%',?),'%')
   ```

5. Click **OK**.

6. Save the EJB module!

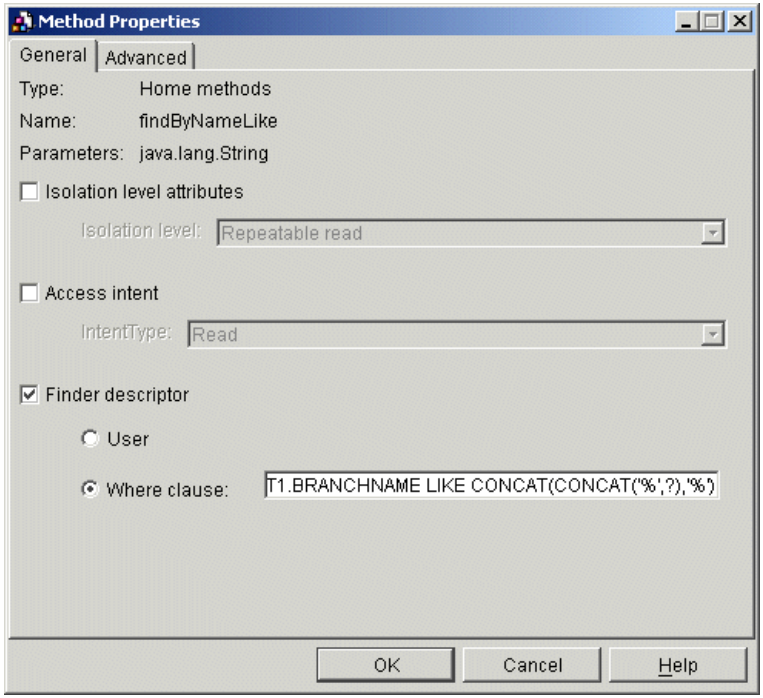


Figure 18-6 Adding an entity bean finder method

When you create a finder, it inserts a definition in the ibm-ejb-jar-ext.xmi file, as shown in Example 18-2.

Example 18-2 Finder descriptor in ibm-ejb-jar-ext.xmi

```
<finderDescriptors xmi:type="ejbext:WhereClauseFinderDescriptor"
   xmi:id="WhereClauseFinderDescriptor_1"
   whereClause="T1.BRANCHNAME LIKE CONCAT(CONCAT('%',?),'%')">
   <finderMethodElements xmi:id="MethodElement_3" name="findByNameLike"
         parms="java.lang.String " type="Home">
         <enterpriseBean xmi:type="ejb:ContainerManagedEntity"
            href="META-INF/ejb-jar.xml#ContainerManagedEntity_1"/>
   </finderMethodElements>
</finderDescriptors>
```

Note that the finder in Example 18-2 is linked to the BranchAccount entity bean using the entity bean name `ContainerManagedEntity_1`. If you change this name in the ejb-jar.xml file (you can only change this by editing the file manually), you must update the extensions and the bindings file accordingly.

Our EJB module is now complete. Next, we package the Webbank Web module.

18.4 Packaging a Web module

Web modules are used to package servlets, JavaServer Pages, and eventually the static part of a Web-based client, such as HTML files or images. All the components of a Web module form a Web application. For this chapter, we have chosen to package the dynamic and static parts together. In this configuration, WebSphere will serve both the static and dynamic content. In a production environment, it is advisable to separate the static contents from the dynamic contents of an application. Please refer to 19.9, “Separating static content from dynamic content” on page 718 for step-by-step deployment instructions for this topology.

The Webbank Web application consists of the following components: a servlet (TransferServlet), a JSP page (message.jsp), and a welcome page (webbank.html). This is a very simple example, but enough to demonstrate the principles of Web module packaging. A Web module is represented by a WAR file (Web archive). A WAR deployment descriptor is stored in a web.xml file.

18.4.1 Creating a Web module

To create a Web module, you can either select the **Web Module** option in the AAT welcome window when it starts up, or click **File** -> **New** -> **Web Module**. A Web module with a default name (New_Web_moduleX.jar) appears on the left-hand side of the AAT window. The following properties can be set for a Web module, as shown in Figure 18-7 on page 654:

- Display Name

  The name of the Web module, used in the AAT or the administrative console.

- Description

  An optional description of what this WEB module contains.

- Distributable

  Checking the Distributable option means that this Web module can be deployed on multiple JVMs, which basically means it can be deployed in a cluster. Servlets which are part of a distributable Web container must be coded in a proper way. For example, the application developer cannot assume than only one instance of this servlet will run (multiple instances will run in multiple JVMs). As a general rule of thumb, all servlets should be developed to be distributable.

- Classpath

  The list of JAR/ZIP files and classes that the EJBs contained in EJB module depends on. The TransferServlet uses classes from webbankCommon.jar, so you must declare this dependency here. If you need to specify multiple JAR files, they must separated by a *space*. Setting this property adds a Class-Path entry to the WAR manifest file, like this:

```
Manifest-Version: 1.0
Class-Path: webbankCommon.jar
```

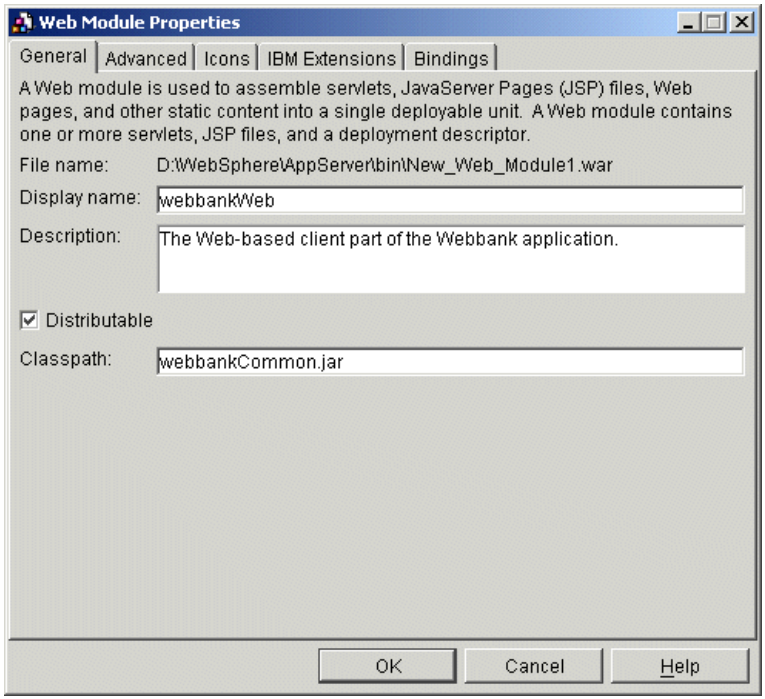


Figure 18-7 Setting the Web module properties

Once you have defined all the basic properties of your Web module, you should save it:

1. Click **File** -> **Save** from the menu bar.
2. Save the Web module as D:\webbank\webbankWeb.war.

You are now ready to add a servlet to your Web module.

18.4.2 Adding files to a Web module

A WAR file contains all the files needed to run the Web-based client side of an enterprise application. Those files are grouped in three categories:

- Class files

  This category contains all servlet classes, as well as the classes servlets may depend on, if they are not packaged in a JAR file. The TransferServlet class file, as well as the BasicHTTPServlet class file fall into this category. All class files are located under the WEB-INF/classes folder. Note that properties files or resource bundles also fall into this category, since they must be available on the CLASSPATH.

- JAR files

  This category contains all JAR/ZIP files that the Web module may depend on. The TransferServlet does depend on the webbankCommon.jar file. However, since it is also needed by the EJB module, we will store it at the enterprise application level (in the EAR file). If a JAR file is used only by a Web module, you should add it to this category. All JAR files are located under WEB-INF/lib.

- Resource files

  This category contains all other files, such as JSP files, HTML files, or images. Resource files go at the root of the WAR file.

For our sample, we need to add the servlet class files, as well as all the resource files. Proceed as follows to add this different files to the Web module:

1. Select **Files -> Class Files -> Add Files**.
2. Click **Browse...**, and select the D:\webbank\webModule directory.
3. Expand the tree under D:\webbank\webModule until you reach the servlets folder, and select both the TransferServlet and BasicHTTPServlet class files.
4. Click **Add**.
5. Click **OK**.
6. Select **Files-> Resource Files -> Add Files**.
7. Click **Browse...**, and select the D:\webbank\webModule directory.
8. Hold the Ctrl key and select all gif, jsp, and html files, and click **Add**.
9. Click **OK**.

18.4.3 Adding a servlet to the Web module

One servlet, TransferServlet, needs to be added to the Web module. To add this servlet:

1. Select the **Web Components** entry and click **New** from the pop-up menu. A window similar to Figure 18-8 starts.
2. Enter a Component name, such as TransferServlet.
3. Optionally, provide a Display name and the description of this component.
4. Select the **Servlet** component type, and click the **Browse...** button.
5. In the window that opens, navigate in the WEB-INF/classes folder, select the TransferServlet class file, and click **OK**.
6. The class name should now be filled up with the correct class name, that is itso.webbank.servlets.TransferServlet.
7. You must check the **Load on startup** option if you want the servlet to be loaded in memory as soon as the server starts (as opposed to when the servlet is first invoked). This is not necessary for the TransferServlet.
8. Click **OK**.

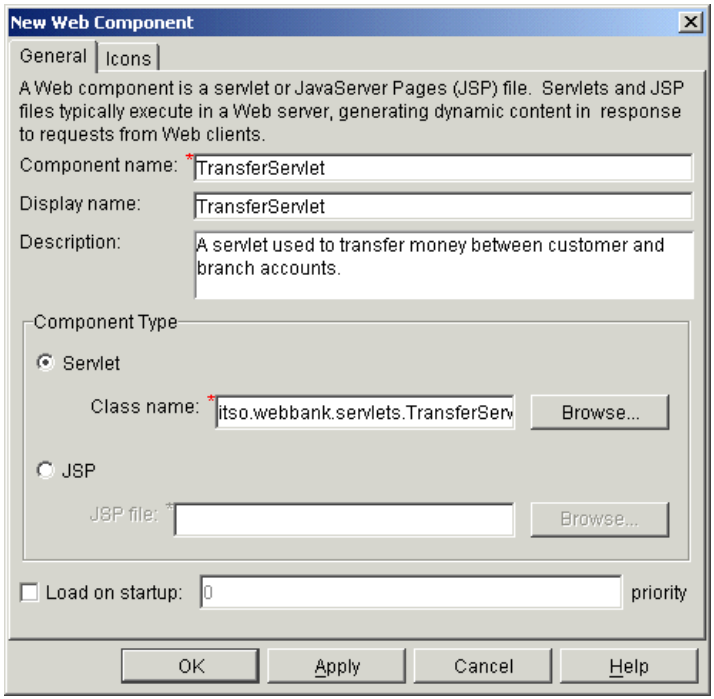


Figure 18-8 Creating the TransferServlet component

18.4.4 Customizing a Web module

Listed here are the standard operations you can perform on a Web module.

Adding initialization parameters

You can supply initialization parameters to the servlet. These parameters can be set from the Initialization Parameters window. To add an initialization parameter to the TransferServlet, you must:

1. Select the **Initialization Parameters** entry under the TransferServlet, and click **New** from the pop-up menu.
2. Supply a Parameter name, such as TraceIsActive, and a Parameter value such as False. These parameters are available from the HTTPConfig object once the servlet is started.
3. Click **OK**.

Adding servlet context parameters

Servlet parameters can also be specified at the Web application level. You will therefore no longer need to duplicate the specification of commonly used parameters. Servlets access these parameters by using the ServletContext methods getInitParameter() and getInitParameterNames(). You define a servlet context parameter similarly to a servlet parameter. Example 18-3 shows how a servlet context parameter is specified in the web.xml file.

Example 18-3 Servlet context parameter deployment descriptor entry

```
<context-param>
   <param-name>TraceIsActive</param-name>
   <param-value>False</param-value>
   <description>Whether trace should be active or not</description>
</context-param>
```

Adding JSP tag libraries

If your application uses custom JSP tag libraries, you must define them at this level. For example, if your application uses the Struts framework, you must define here the different Struts tag libraries used in the JSPs. To add a tag library definition:

1. Select the **Tag Library** entry and choose **New** from the pop-up menu.
2. Specify the tag library file name, which is the URI you use to refer to a tag library file in a JSP. For example, if you specify `<%@ taglib uri="/tags/struts-bean.tld" prefix="bean" %>` then /tags/struts-bean.tld is the URI for the struts-bean.tld file.
3. Specify the tag library location, which is the actual physical location of the struts-bean.tld file, relative to the root of the Web module. Typically, you would place tag library files under WEB-INF\lib. Therefore, you must specify /WEB-INF/lib/struts-bean.tld as the file location.
4. Click **OK**.

Example 18-4 shows the entry added to the web.xml file when defining the tag library.

Example 18-4 Tag library deployment descriptor entry

```
<taglib>
   <taglib-uri>/tags/struts-bean.tld</taglib-uri>
   <taglib-location>/WEB-INF/lib/struts-bean.tld</taglib-location>
</taglib>
```

Adding error pages

Specify one error page per exception type, or per error type (403, 404, and so on). In the error pages properties list, you may specify special Web pages when a special status code or exception occurs. For example, you may specify different Web pages to display when status 404 or 500 occurs, or you may specify different Web pages to display when exceptions such as javax.servlet.ServletException and java.io.IOException are thrown.

If a request causes both a status code to be generated and an exception to be thrown, and both these errors have specified error pages, then WebSphere uses the error page configured for the status code. If an error occurs that is not included in the list of error pages, the default error page (see 18.12.4, “Default error page” on page 682) is used.

Adding welcome files

The Servlet API V2.2 defines facilities for specifying a list of welcome files for Web applications. Welcome files define the response to requests without a specific file name, for example `http://<hostname>/webbank`. Welcome files are only effective when the Web application enables the file serving servlet as described in 18.12.1, “File serving servlet” on page 681. If static files are served by an HTTP server, then the rules of the HTTP server apply. For the IBM HTTP Server, the DirectoryIndex directive is used.

WebSphere searches the list of welcome files and returns the first file in the list that is actually present on disk. If none of the specified welcome files can be found, WebSphere looks for the default, index.html.

Adding MIME mappings

For the MIME table property, specify mappings between extensions and MIME types. The MIME table consists of:

- MIME type

  The defined MIME type associated with the extension, such as text/plain.

- Extension

Text string describing an extension, such as .txt.

You can also specify MIME table properties at the virtual host level, but the MIME table properties you specify for a Web application take precedence (local scope). In other words, the MIME table of the Web application is searched first. If a match is not found, then the MIME table configured for the virtual host is searched.

18.4.5 Declaring servlet mappings

You can declare one or several servlet mappings for each servlet. Servlet mappings are used to link a “logical” name, such as TransferServlet, to the actual TransferServlet code. If you do not create a servlet mapping, you must invoke a servlet using its full class name (itso.webbank.servlets.TransferServlet). This is only possible if you have checked the "Serve servlets by class name" option for the Web module. This option is described in 18.12.3, “Serve servlets by class name” on page 682.

It is highly recommended that you always create servlet mappings for your servlet. Lots of features (such as security) are not available if your servlet has no mapping.

Follow those steps to create a servlet mapping for the TransferServlet:

1. Select the **Servlet Mapping** entry, and choose **New** from the pop-up menu.
2. Enter `/TransferServlet` as the URL pattern and select the TransferServlet in the Servlet drop-down. The leading slash is critical; make sure not to omit it.

Note: If you want to invoke a servlet each time a file with a certain extension is handled by the server, you should use a `*.extension` mapping instead. A servlet mapping of `*.xml` implies that any request of type `http://host/URI/*.xml` is handled for this servlet.

3. Click **OK**.
4. Save the Web module.

18.4.6 Declaring JSPs as Web components

Any JSP file located relative to the root of a Web module can be served by the WebSphere Application Server. It is then handled by the JSP 1.1 Processor servlet. However, you can also declare a JSP as a Web component. If you do so, you will be able to:

- Secure the JSP file, as any other servlet.
- Pass initialization parameters to the JSP.

- Compile and load the JSP at server startup. If you select the **Load on startup** option, the JSP file is compiled and loaded in memory as soon as the server starts. This prevents the first user of the application from paying for the JSP compile time.

Note: You can also use the JSP batch compiler to precompile all JSPs before starting a server.

Example 18-5 shows how a JSP component is declared in the web.xml file.

Example 18-5 JSP component deployment descriptor entry

```
<servlet>
   <servlet-name>Message JSP</servlet-name>
   <jsp-file>message.jsp</jsp-file>
   <load-on-startup>1</load-on-startup>
</servlet>
```

Your Web module is now complete. We now cover how to package a client application.

18.5 Packaging a client application

Two types of client applications can use components deployed in a WebSphere application server:

- J2EE clients

  A J2EE client runs in a J2EE client container, which you have to install on each client machine. This client gives you full "J2EE power". Like any J2EE component, a J2EE client has an XML deployment descriptor. J2EE naming (including EJB and resources references) as well as security are available to J2EE clients.

- Java thin clients

  A thin Java client does not require a specific container to run in. You only need to deploy the WebSphere runtime classes on the client machine. Using this client is best suited to environments migrating from WebSphere V3.x, or when modifying existing applications to use enterprise beans. Additionally, a thin Java client requires a thinner, more lightweight environment than a J2EE client to run. However, you do not benefit from the J2EE packaging features and their potential future enhancements.

In this section, we cover how to package the stand-alone client for the Webbank application as a J2EE client. This client lets you to retrieve the branch and customer accounts balance. J2EE client applications are packaged in a JAR file. To package a client module, you must:

1. Click **File** -> **New** -> **Application Client** to create a new application client module.
2. The next step is to add all the client files (and eventually their dependent classes) to the client application. Click **Files -> Add Files**.
3. Click **Browse** and select **D:\webbank\appClientModule**.
4. Expand the tree view in the left pane until you reach the testclients folder, select it, select all classes in the right pane, and click **Add**.
5. The next step is to provide the general client application settings in the window shown in Figure 18-9 on page 662:
   - Display name

     The name used by GUI tools when displaying this client application.
   - Description

     An optional description of the contents of this client application.
   - Classpath

     The list of classes, JAR/ZIP files that this client application depends on. Our client depends on the webbankCommon.jar and the JAR file that contains all EJB classes. You only need to list the webbankCommon.jar file in this field. The dependency on EJB JAR file is added automatically when you package the client application in an enterprise application, as described in the next section.
   - Main class

     The name of the class that will be invoked when using the LaunchClient program (the class that has the main() entry point). Click the **Browse...** button next to the field, and select the itso.webbank.testclients.AccountViewer class.
6. Save the client application module as D:\webbank\webbankClientApp.jar by selecting **File** -> **Save**.

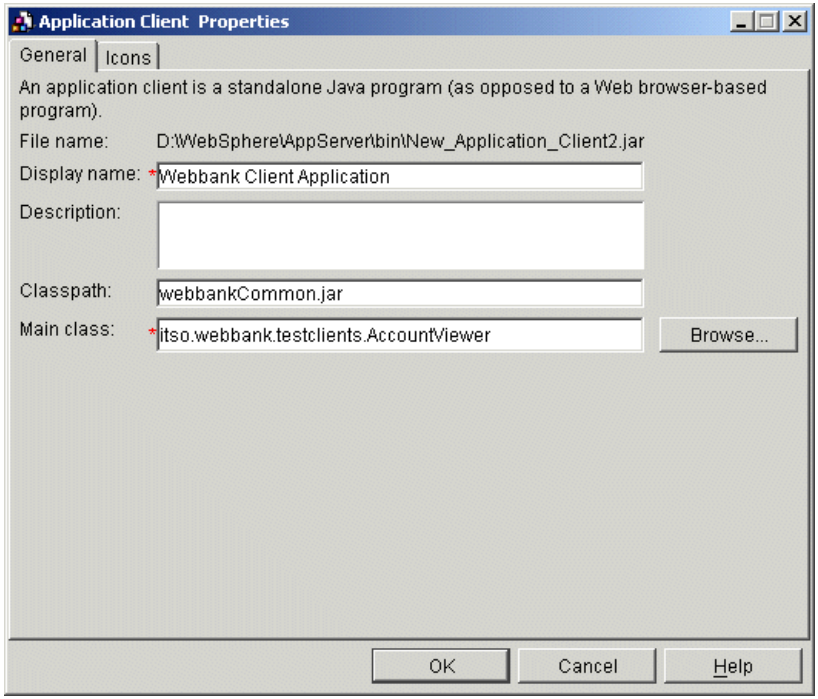


Figure 18-9 Setting the application client general properties

You have now completed the packaging of the application client. Next, you will associate all the modules you have created so far in an enterprise application.

18.6 Packaging the Webbank enterprise application

You should now have four archive files in the D:\webbank directory:

- webbankClientApp.jar, which contains the stand-alone client application.
- webbankWeb.war, which contains the Web-based client application.
- webbankEJBs.jar, which contains the Webbank enterprise beans.
- webbankCommon.jar, which contains the Webbank common classes.

You will now use the Enterprise Application wizard to create the archive for the Webbank enterprise application, which consists of the modules listed above. The wizard consists of multiple windows, which we detail now. To package the full Webbank application:

1. Start the Enterprise Application wizard by clicking **File** -> **Wizards** -> **Create Application Wizard**. The wizard opens with the Specifying Application Properties window.
2. You can specify the following properties for the application:
   - Display name

     The name used to identify this application, such as webbankApplication.
   - File name

     The name of the EAR file corresponding to this application, such as D:\webbank\webbank.ear.

     Tip: If you just supply the file name (such as webbank.ear), the file will be saved in the AAT working directory. If you have started the AAT from the command line, for example from D:\webbank, then D:\webbank is your working directory. Otherwise, <WAS_HOME>\bin is your working directory. Supply a full path name, such as D:\webbank\webbank.ear, to prevent this.
   - Description

     An optional description of what this application does. We recommend that you put a meaningful description of the application for the application deployer.
3. Click **Next**. You are now on the Adding Supplementary Files window. So far, we have created multiple references to the webbankCommon.jar file in various modules. However, this file is not stored in any of the modules we have created. You will now place it at the root of the EAR file from this window. All modules that have declared webbankCommon.jar in their classpath will load it from there.
4. Click **Add**, navigate to the D:\webbank folder, and click **Select**.
5. Select the webbankCommon.jar in the right pane, click **Add**, then **OK**.
6. Click **Next**. You are now on the Choosing Application Icons window. You could select .gif or .jpg icons. (Sorry, but we didn't have time to create icons for the Webbank application.)
7. Click **Next**. You are now on the Adding EJB Modules window. Click **Add**, and select the D:\webbank\webbankEJBs.jar file. You are then prompted to confirm values, where you can specify an Alternate DD (deployment descriptor) value. Click **OK**.
8. Click **Next**. You are now on the Adding Web Modules window. Click **Add**, and select the D:\webbank\webbankWeb.war file. You are now prompted for the Web application context root, that is the unique URI of the Web application.

You should enter /webbank in this field, so that all HTTP requests starting with `http://<hostname>/webbank` are handled by this application. Click **OK**.

9. Click **Next**. You are now on the Adding Application Client Modules window. Click **Add**, and select the D:\webbank\webbankClientApp.jar file. Click **OK** to Confirm values.
10. Click **Next**. You are now on the Adding Security Roles window. Please refer to Chapter 21, “Configuring security” on page 739 for more information.
11. Click **Finish** to create the enterprise application.

Note: If you select the Webbank application client, you can see that the webbankEJBs.jar file has been automatically added to the Classpath field. A J2EE client application is very likely to use enterprise beans packaged in the same application file. Therefore, this dependency is automatically managed for you.

18.7 Declaring environment variables

In WebSphere V3.5, environment variables used by enterprise beans were stored in the EJB runtime context. With WebSphere V4.0, you must use the J2EE naming service. The CustomerAccount EJB uses an environment variable to store the overdraft limit allowed on an account. Environment entries declared at the EJB module level belong to the EJB module local JNDI namespace; they cannot be reached from another module (please refer to Chapter 3, “The Java 2 platform” on page 29 if local and global JNDI namespaces do not sound familiar).

Follow those steps to declare the OverdraftValue variable:

1. Locate the CustomerAccount entity bean in the navigation pane.
2. Right-click **Environment Entries** and select **New** from the pop-up menu. A window similar to Figure 18-10 on page 665 opens.
3. Supply the following information to create the entry:
   - Name

     The environment entry name, such as "OverdraftValue". You can also create a subcontext using "webbank/OverdraftValue" for example.
   - Value

     The environment entry value.
   - Type

     The Java type of this entry. Only basic Java types, such as Integer, Float, or String, are allowed.

– Description

  Optionally, you can provide a short description of the value.

Example 18-6 shows how the OverdraftValue environment entry is declared in the ejb-jar.xml file.

Example 18-6 Environment entry in deployment descriptor

```
<env-entry id="EnvEntry_1">
   <description>The limit for customer accounts overdrafts.</description>
   <env-entry-name>OverdraftValue</env-entry-name>
   <env-entry-type>java.lang.Integer</env-entry-type>
   <env-entry-value>5000</env-entry-value>
</env-entry>
```

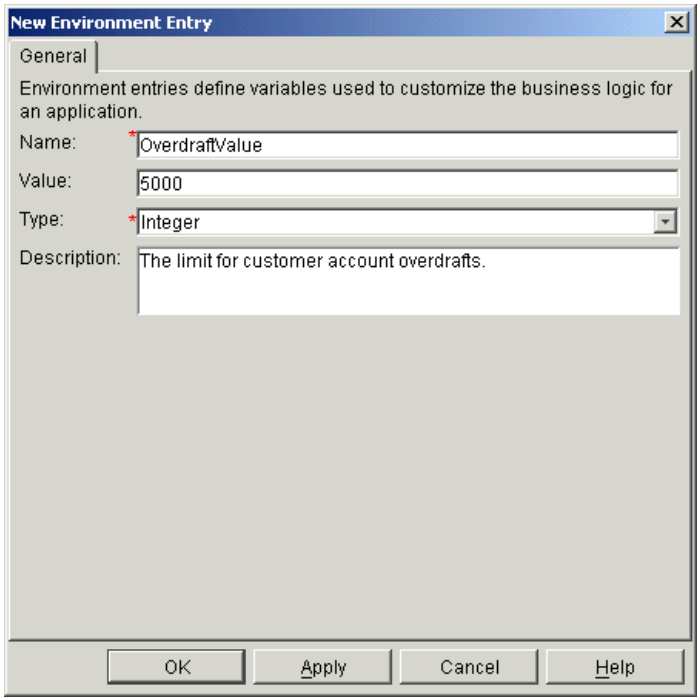


Figure 18-10 Adding a new environment entry

Note: Proceed the same way to add an environment variable at the Web module or enterprise application levels.

The OverdraftValue environment entry value can then be retrieved using the snippet of code given in Example 18-7.

Example 18-7 Retrieving an environment entry value

```
javax.naming.Context ic = new javax.naming.InitialContext();
Context environment = (Context) ic.lookup("java:comp/env");
Integer overdraftValue = (Integer) environment.lookup ("OverdraftValue");
```

18.8 Creating EJB references

J2EE naming enforces the notion of local namespaces for all J2EE components, such as servlets or EJBs. The goal is to shield the developer from all deployment details. All EJB clients (such as servlets, J2EE clients, or other EJBs) should use J2EE naming to lookup EJBs.

As an example, the Consultation session bean uses the BranchAccount entity bean. Therefore, at packaging time, we need to define an EJB reference to create the "ejb/BranchAccount" environment entry in the Consultation EJB local JNDI namespace. At deployment time (see 19.2.3, “Binding EJB references to EJB JNDI names” on page 700), we link this EJB reference to the actual JNDI name of the BranchAccount EJB. To look up the BranchAccount EJB, we could then use the code snippet given in Example 18-8.

Example 18-8 Retrieving an EJB environment entry value

```
javax.naming.Context ic = new javax.naming.InitialContext();
Object o = ic.lookup("java:comp/env/ejb/BranchAccount");
BranchAccountRef = (BranchAccountHome)
   PortableRemoteObject.narrow (o,BranchAccountHome.class);
```

Important: Each EJB has its own local JNDI name space. For example, EJB references created for the Transfer session bean are *not* visible from the Consultation session bean. For simplicity, we chose in this sample to use the same ejb/BranchAccount JNDI name in both. These two names are completely independent and unrelated. You could very well create an ejb/BranchAccountRW EJB reference for the Transfer session bean, and an ejb/BranchAccountRO EJB reference for the Consultation session bean.

Follow these steps to create an EJB reference between the Consultation bean and the BranchAccount bean:

1. Select the **EJB References** entry for the Consultation bean, and click **New** from the pop-up menu. A window similar to Figure 18-11 on page 667 starts.

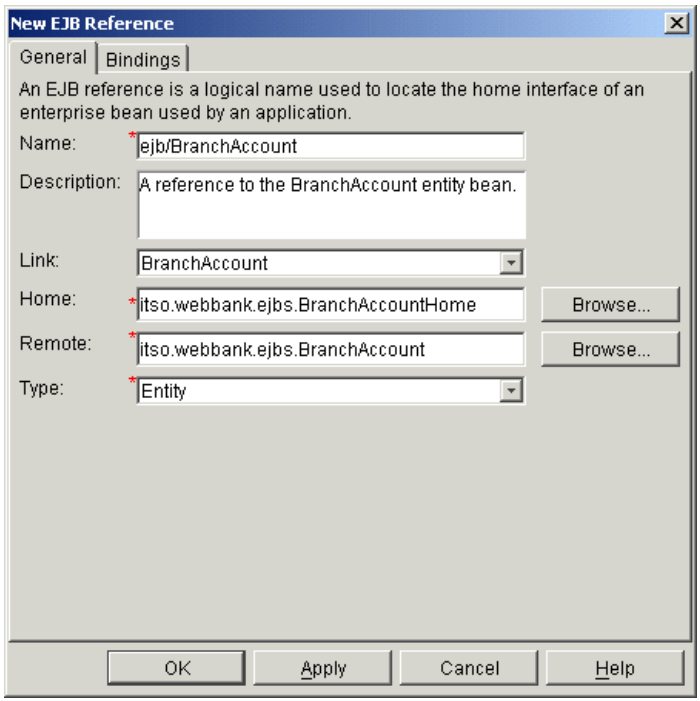


Figure 18-11 Creating an EJB reference

2. Provide the following information to create the EJB reference:
   - Name

     The EJB reference name, used to look up the EJB in the local JNDI name space, that is, "ejb/BranchAccount".
   - Description

     Optionally, provide a description of this EJB reference.
   - Link

     If the target EJB is located in the same or in a different EJB module *in the same EAR file*, you can optionally link to it. Use the drop-down menu to select the BranchAccount EJB and the remaining fields are automatically filled in. You must use the ejb-name of the target bean in this field. Do not use an EJB link if the EJB is deployed in a different EAR file.
   - Home

     The type of the target enterprise bean home interface, such as itso.webbank.ejbs.BranchAccountHome.

– Remote

The type of the target enterprise bean remote interface, such as itso.webbank.ejbs.BranchAccount.

– Type

The EJB type. This field must be set to either "Session" or "Entity".

3. Do not worry about the bindings page for now. We will cover bindings in the deployment phase.
4. Click **OK**.
5. Using the information supplied in Table 18-3, repeat steps 1 to 4 to create all necessary EJB references in the Webbank application.

Example 18-9 shows the BranchAccount EJB reference entry in the EJB deployment descriptor.

Example 18-9 EJB reference deployment descriptor entry

```
<ejb-ref id="EjbRef_1">
   <description>A reference to the BranchAccount entity bean</description>
   <ejb-ref-name>ejb/BranchAccount</ejb-ref-name>
   <ejb-ref-type>Entity</ejb-ref-type>
   <home>itso.webbank.ejbs.BranchAccountHome</home>
   <remote>itso.webbank.ejbs.BranchAccount</remote>
   <ejb-link>BranchAccount</ejb-link>
</ejb-ref>
```

You should use the information supplied below to create all references for the Webbank application.

Table 18-3 EJB references list

From	To	EJB reference name
Consultation Session bean	BranchAccount entity bean	ejb/BranchAccount
Consultation Session bean	CustomerAccount entity bean	ejb/CustomerAccount
Transfer Session bean	BranchAccount entity bean	ejb/BranchAccount
Transfer Session bean	CustomerAccount entity bean	ejb/CustomerAccount
Transfer Servlet	Transfer Session Bean	ejb/Transfer
Client Application	Consultation Session Bean	ejb/Consultation

18.9 Creating resource references

Servlets and EJBs can access external resources, such as databases, or messaging systems. You should be using the local JNDI namespace to store local references to those resources at packaging time. At deployment time, you will map this resource reference to the actual resource. The following types of resources are supported:

- JMS connection factories, for access to messaging systems. It is recommended that you use the *jms* subcontext to store these resources.
- JavaMail connection factories, for mailing support. It is recommended that you use the *mail* subcontext to store these resources.
- JDBC data sources, for access to relational databases. It is recommended that you use the *jdbc* subcontext to store these resources.
- URL connection factories, for access to URLs, such as FTP or HTTP URLs. It is recommended that you use the *url* subcontext to store these resources.
- J2C connection factories. It is recommended that you use the *eis* subcontext to store these resources.

The Webbank sample EJBs do not use resource references, but let's suppose you want an EJB to use a messaging system to post a message on a logging queue whenever a transfer is completed. The snippet of code in Example 18-10 can be used to look up the message queue connection factory.

Example 18-10 Retrieving a resource reference

```
javax.naming.Context ic = new javax.naming.InitialContext();
webbankQCF = (javax.jms.QueueConnectionFactory)
   ic.lookup ("java:comp/env/jms/msgQCF");
```

You can create an EJB resource reference for this JMS queue connection factory as follows:

1. Select the **Resources References** entry under the required bean, and click **New**. A window similar to Figure 18-12 on page 670 appears.

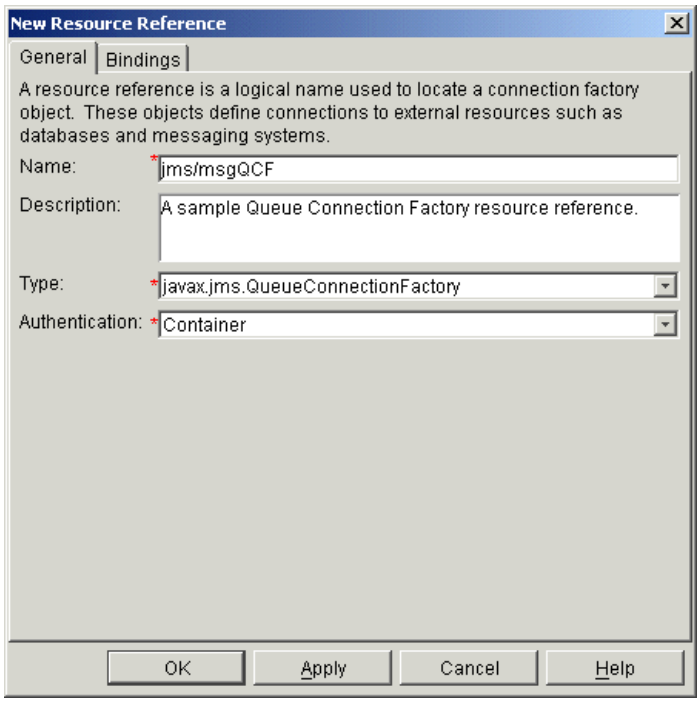


Figure 18-12 Creating a resource reference

2. Provide the following information to create the JMS resource reference:
   - Name

     The resource reference name, for example "jms/msgQCF".
   - Type

     The type of resource this reference points to.
   - Description

     An optional description of this reference.
   - Authentication

     This value sets whether the authentication to the resource (such as the database user ID/password) is done by the container or in the application. If the value is set to Container, it indicates that the authentication to the resource will be done using the authentication information supplied in the resource definition. Otherwise, the application developer must provide authentication information, for example when obtaining a connection to a

database. It is recommended that you use the Container, because the application developer should not worry about getting the right user ID/password at development time. Example 18-11 shows the resource reference entry in the EJB deployment descriptor.

Example 18-11 Resource reference deployment descriptor entry

```
<resource-ref>
   <description>A sample Queue Connection Factory resource reference.
      </description>
   <res-ref-name>jms/msgQCF</res-ref-name>
   <res-type>javax.jms.QueueConnectionFactory</res-type>
   <res-auth>Container</res-auth>
</resource-ref>
```

See also Chapter 16, “Configuring WebSphere resources” on page 563.

18.10 Setting EJB transactional attributes

We assume in this section that you are already familiar with transaction concepts. For an introduction to enterprise beans and transactions, please refer to Chapter 3, “The Java 2 platform” on page 29.

The transactional attribute sets how the EJB container manages transactions when a home or remote method is invoked. This attribute determines if and how transactions are needed.

18.10.1 Transaction attributes overview

The possible transaction attribute values are listed in Table 18-4.

Table 18-4 Transaction attributes values

Transaction attribute	What the container does when this value is set
Mandatory	The container must call methods from the transactional context established by the client. If the client has a transaction context, it is propagated to the bean. If the client does not have a transaction context, the container throws a TransactionRequiredException. Consider using Mandatory with entities to assure a transaction is active before the entity is called (to enforce that they are called from a session bean, and not directly from a client application).

Transaction attribute	What the container does when this value is set
Required	The container must invoke methods within a transactional context. If the client has a transaction context, it is propagated to the bean. If not, the container starts a transaction. Consider using Required with session beans.
Requires New	The container must start a new transaction for the method. If the client has a transaction context, it is suspended for the duration, and a new one is started. If not, the container starts a transaction.
Supports	If the client has a transaction context, it is propagated to the bean. If not, the method is run under what the EJB specification calls the unspecified transaction context.
Not Supported (Default)	The container must not invoke methods transactionally. If the client has a transaction context, it is suspended for the duration of the method call, and resumes when the method invocation returns.
Never	The container must not invoke methods transactionally. If the client has a transaction context, a java.rmi.RemoteException exception is thrown.

Explaining in detail the different transaction attributes values and how they should be used is beyond the scope of this book. Yet, we recommend that you use them with care, and provide a few cautions:

- Use transactions when you need them. That may seem obvious, but using transactions is not free (there is an execution time cost).
- Let the container handle transactions. Although you can either decide from a client or a session bean transaction when a transaction begins and ends (this is called transaction demarcation), the EJB container can do this better 95% of the time!
- Make sure you understand the scope of the transactions in your application to avoid "long-lived" transactions. Most of the time, transactions involve locks on back-end systems, such as databases. While you are locking a resource, the chances are other clients can't do much with it. This can greatly influence the performance and scalability of your application.
- Use Required or Requires New in your session beans to assure a transaction is started.
- Use Mandatory in your entity beans to assure the changes are part of a bigger transaction being coordinated by a session bean.

18.10.2 How to set the transaction attribute

A transaction attribute is set from the Container Attributes entry in an EJB module. You can set the transaction attribute for:

- All the enterprise bean methods
- All the home interface methods
- All the remote interface methods
- An individual method

You can combine these options. For example, you can set all methods of your session bean to have the Required transaction attribute but one, which would have the Requires New attribute.

18.11 IBM EJB extensions

Multiple IBM extensions are available to influence the behavior of the EJB container at runtime.

18.11.1 EJB container caching option for entity beans

The Enterprise JavaBeans specification defines three EJB caching options: options A, B, or C. Those options define how the EJB container handles entity bean instances between transactions. EJB caching options are set at the bean level, and are part of the IBM extensions deployment descriptor.

Caching option A

With caching option A, you assume that the entity bean has *exclusive* access to the underlying persistent store. In other words, between transactions, no one can modify the data. This includes a batch program updating the data, a Java application updating the data, or even the same entity bean running in a *different* container. This implies option A cannot be used in a clustered environment (WLM). Note that it is your responsibility to ensure no other application will modify the data, as the EJB container has no way to control write access to the underlying database from other servers.

When caching option A is used, the entity bean instance is kept in a memory cache across transactions. At transaction commit, the entity bean attributes are synchronized with the underlying persistent store, and the bean instance remains cached in memory.

If you were tracing the calls made by the container, you would see something similar to Example 18-12. The first time the entity bean is used, its runtime context is set (step 1), a bean is taken from the entity beans instance pool (step 2), the bean instance attributes are synchronized with the underlying data store (step 3), the method setBalance is invoked on the bean (step 4), and finally the bean attributes are saved back to the database (step 5). The bean is not returned to the pool. On subsequent calls, the setBalance method is invoked directly on the cached bean instance, and the bean attributes are synchronized with the underlying persistent datastore.

Example 18-12 Entity beans call trace with option A caching

```
Transaction 1 (Begin)
Step 1: 1c9585f1 BranchAccount E called setEntityContext() method
Step 2: 1c9585f1 BranchAccount E called ejbActivate() method
Step 3: 1c9585f1 BranchAccount E called ejbLoad() method
Step 4: 1c9585f1 BranchAccount E called setBalance() method
Step 5: 1c9585f1 BranchAccount E called ejbStore() method
Transaction 1 (Commit)

Transaction 2 (Begin)
Step 1: 284485f1 BranchAccount E called setBalance() method
Step 2: 284485f1 BranchAccount E called ejbStore() method
Transaction 2 (Commit)
```

Using caching option A can provide some performance enhancements at the expense of higher memory usage. You should only use it if you do not intend to use WebSphere clustering capabilities and you mostly access data in read mode.

Caching option B

With caching option B, you assume that you have *shared* access to the underlying database, which means the data could be changed by another application between transactions. When option B is used, the bean instance attributes are always synchronized with the underlying back-end datastore at the beginning of every transaction. Similarly to Option A, the bean is kept in the cache between transactions. Therefore, if you were tracing the different calls made in Option B, you would obtain the trace shown in Example 18-13.

Example 18-13 Entity beans call trace with option B caching

```
Transaction 1 (Begin)
Step 1: 1c9585f1 BranchAccount E called setEntityContext() method
Step 2: 1c9585f1 BranchAccount E called ejbActivate() method
Step 3: 1c9585f1 BranchAccount E called ejbLoad() method
Step 4: 1c9585f1 BranchAccount E called setBalance() method
Step 5: 1c9585f1 BranchAccount E called ejbStore() method
Transaction 1 (Commit)
```

```
Transaction 2 (Begin)
Step 1: 284485f1 BranchAccount E called ejbLoad() method
Step 2: 284485f1 BranchAccount E called setBalance() method
Step 3: 284485f1 BranchAccount E called ejbStore() method
Transaction 2(Commit)
```

Caching option B can be safely used in a clustered environment, or when you are not sure if you have exclusive access to data. You are assured that you always work with the last committed data. Option B memory usage is the same as for option A. The performance of both options may slightly differ depending on the nature of your application. Option B support is new in WebSphere Application Server V4.0.

Caching option C

Similarly to option B, caching option C assumes *shared* access to the database. Unlike option B or A, the bean instance is returned to the entity beans pool at the end of the transaction. A new bean instance is used at the beginning of every transaction. Each transaction results in the sequence of calls shown in Example 18-14.

Example 18-14 Entity beans call trace with option C caching

```
Transaction (Begin)
Step 1: 1c9585f1 BranchAccount E called setEntityContext() method
Step 2: 1c9585f1 BranchAccount E called ejbActivate() method
Step 3: 1c9585f1 BranchAccount E called ejbLoad() method
Step 4: 1c9585f1 BranchAccount E called setBalance() method
Step 5: 1c9585f1 BranchAccount E called ejbStore() method
Step 6: 1c9585f1 BranchAccount E called ejbPassivate() method
Step 7: 1c9585f1 BranchAccount E called unsetEntityContext() method
Transaction (Commit)
```

Caching option C has the best memory usage at the expense of a larger number of methods calls. This is the default behavior.

How to set the EJB caching option

In WebSphere V3.5, you could set this option from the administrative console, by setting the access to the database to Shared or Exclusive. With WebSphere Application Server V4.0, you have to use the AAT. The setting is found on the IBM Extensions window of an entity EJB, in the Bean Cache category. You must combine the "Activate at" and "Load at" options to set the EJB caching option to A, B, or C. Use Table 18-5 to choose the right combination.

Table 18-5 Setting entity EJB caching properties

Option	Activate at must be set to	Load at must be set to
Option A	Once	Activation
Option B	Once	Transaction
Option C (default)	Transaction	Transaction

This setting is saved in the IBM extensions deployment descriptor, ibm-ejb-jar-ext.xmi file. It corresponds to the following entry (one line per entity bean you have set this option for):

```
<beanCache xmi:id="BeanCache_1" activateAt="ONCE" loadAt="ACTIVATION"/>
```

See also 17.5.2, “How EJBs participate in WLM” on page 616.

18.11.2 EJB container caching option for stateful session beans

Similarly to entity beans, you can specify which caching strategy should be used for stateful session beans. This caching option specifies the point at which an enterprise bean is activated and placed in the cache. Removal from the cache and passivation are also governed by this setting. Valid values are:

- Once (default)

  Indicates that the bean is activated when it is first accessed in the server process. It is passivated (and removed from the cache) at the discretion of the container, for example, when the cache becomes full.

- Transaction

  Indicates that the bean is activated at the start of a transaction and passivated (and removed from the cache) at the end of the transaction.

You can set this caching option using the "Activate at" setting, found on the IBM Extensions window of a stateful session EJB, in the Bean Cache category.

18.11.3 Stateful EJB timeout option

Additionally, you can specify a timeout value for stateful session beans. A bean can time out in the METHOD_READY or in the PASSIVATED state. If you try to access a bean that has timed out, you will see an exception similar to:

```
com.ibm.ejs.container.SessionBeanTimeoutException: Stateful bean
StatefulBean0(BeanId(Webbank#webbankEJBs.jar#Transfer, ebf64d846a),
state = METHOD_READY) timed out.
```

Session beans that have timed out can be removed by the container, for example if it needs to free memory. However, a well-written application should not rely on beans to time out to free memory. Instead, it is important that the developer explicitly calls remove() when a stateful bean is not needed anymore.

The default timeout is 600 seconds. You can set the timeout value from the AAT as a parameter of a stateful session bean. Setting this timeout will insert the following property in the ejbExtensions tag of the IBM bindings file:

```
<ejbExtensions xmi:type="ejbext:SessionExtension"
   xmi:id="Session_1_Ext" timeout="120">
```

Note: If a bean times out in the METHOD_READY state and is consequently removed by the container, the ejbRemove() method will be called on the bean instance. If a bean times out in the passivated state, ejbRemove() is not called (as per the EJB specification).

18.11.4 Local transactions settings

A local transaction context is created when a method executes in what the EJB specification refers to as an unspecified transaction context. This includes methods such as ejbCreate, ejbRemove or ejbActivate when their transaction attribute has been set to Required or Requires New (you can refer to the EJB specification for a full list). The EJB specification does not describe any standard way to recover from a failure during the execution of such a method. The settings given in Table 18-6 provide some options for handling such transactions.

Table 18-6 Local transactions settings

Local transactions setting	Details
Boundary	Specifies when a local transaction begins. The default behavior is that the local transaction begins when the bean method begins, and ends when the bean method ends. This property is not applicable for session beans.

Local transactions setting	Details
Unresolved action	Specifies the action the container must take if resources are uncommitted by an application in a local transaction. Valid values are Rollback and Commit. The default is Rollback.

You can set this option in AAT using the Local Transactions category, found on the IBM Extensions window of an EJB.

18.11.5 Isolation level attributes

The transaction isolation attribute tells the container how to limit concurrent reads in a database. The EJB 1.1 specification removed the guidelines for managing transaction isolation levels for beans with container-managed transaction demarcation. But since bean deployers still require mechanisms to govern EJB concurrency, WebSphere V4.0 continues to support it.

Transaction isolation levels overview

Transaction isolation levels provides a trade-off between accuracy of reads versus concurrent readers. The levels can best be described by the types of read anomalies they permit and forbid. Consider the read anomalies that can occur with two concurrent transactions, T1 and T2:

- Dirty read: T1 reads data that has been modified by T2, before T2 commits.
- Non-repeatable read: This is caused by fine-grained locks.
  - T1 reads a record and drops its lock.
  - T2 updates.
  - T1 re-reads different data.
- Phantom read: A non-repeatable read involving a range of data and inserts or deletes on the range.
  - T1 reads a set of records that match some criterion.
  - T2 inserts a record that matches the criterion.
  - T1 continues processing the set, which now includes records that were not part of the original matching set.

There are four possible settings for the transaction isolation level:

- Repeatable read (`TRANSACTION_REPEATABLE_READ`)

  Permits phantom reads and forbids both dirty and unrepeatable reads.

- Read committed (`TRANSACTION_READ_COMMITTED`)

  Permits non-repeatable and phantom reads and forbids dirty reads.
- Read uncommitted (`TRANSACTION_READ_UNCOMMITTED`)

  Permits all the read anomalies including dirty reads, non-repeatable reads, and phantom reads.
- Serializable (`TRANSACTION_SERIALIZABLE`)

  Forbids all the read anomalies.

The container applies the isolation level as follows:

- For entity beans with CMP, the container generates code that assures the desired level of isolation for each database access.
- For session beans and BMP entity beans, the container sets the isolation level at the start of each transaction, for each database connection.

The transaction isolation level is tied to a database connection. The connection uses the isolation level specified in the first bean that uses the connection. The container throws an IsolationLevelChangeException whenever the connection is used by another bean method that has a different isolation level.

Not all databases support all JDBC isolation levels. Moreover, JDBC definitions for isolation levels may not match the database definition of isolation levels. As an example, DB2 definitions for isolation levels follow the naming conventions used in Jim Gray's classic book on transaction processing, *Transaction Processing: Concepts and Techniques*. Table 18-7 shows a mapping between EJB and DB2 isolation levels.

Table 18-7 Mapping JDBC isolation levels to DB2 isolation levels

JDBC isolation level	**DB2 isolation level**
TRANSACTION_SERIALIZABLE	Repeatable Read
TRANSACTION_REPEATABLE_READ	Read Stability
TRANSACTION_READ_COMMITTED	Cursor Stability
TRANSACTION_READ_UNCOMMITTED	Uncommitted Read

To learn more, you should refer to the documentation provided by the database product.

Setting isolation levels

Isolation levels can be configured in AAT from an EJB's Method Extensions entry. They can be set at the bean level (* - All methods option), at the home methods level (* - Home Methods option), at the remote methods level (* - Remote methods option) or at the individual level option. It is often easier to set this value at the bean level, since managing this at the method level may add complexity to deployment and debugging.

18.11.6 Read-only methods

When you invoke an entity bean business method, the EJB container assumes that changes have been made to the bean attributes and systematically synchronizes the bean attributes with the persistent datastore. When a method is marked as read-only, the container skips the ejbStore() operation at the end of a transaction (regardless of the EJB caching option you are using, A, B, or C).

You should mark all the getXXX() methods of your entity beans as being read-only. This avoids possible read to write lock promotions and can significantly improve performance and concurrency. Note that it is your responsibility to ensure that methods you mark as read-only *are* read-only methods. The EJB container has no way of knowing this. Moreover, you should use this flag only for your business methods.

In WebSphere V3.5, you could mark methods as being read-only from the administrative console. With WebSphere Application Server V4.0, you must use the AAT. This setting is accessible from the Method Extensions entry (for entity beans only). To set a method as read-only:

1. Select one method in the methods list.
2. Check the **Access intent** option.
3. Set the IntentType property to **Read**.
4. Click **Apply**.

Example 18-15 shows lines inserted in the ibm-ejb-jar-ext.xmi file when you use this flag.

Example 18-15 Access intents deployment descriptor entry

```
<accessIntents xmi:id="AccessIntent_2" intentType="READ">
   <methodElements xmi:id="MethodElement_3"
      name="getBalance" parms="" type="Remote">
      <enterpriseBean xmi:type="ejb:ContainerManagedEntity"
         href="META-INF/ejb-jar.xml#ContainerManagedEntity_1"/>
   </methodElements>
</accessIntents>
```

18.11.7 EJB inheritance/relationships

This functionality, which is not part of the EJB 1.1 specification, is only available via the WebSphere application development tools, such as WebSphere Studio Application Developer or VisualAge for Java. Please refer to the documentation of those tools for more details.

18.12 IBM Web modules extensions

WebSphere Application Server V4.0 provides multiple extensions for Web modules. These extensions are available on the IBM Extensions tab on a Web module's properties pane.

18.12.1 File serving servlet

When dealing with static HTML pages, you can choose to have static pages served by WebSphere or have them served by the Web server itself.

If you want WebSphere to serve the static content of your application, you must enable the file servlet (or file serving servlet or file serving enabler). This servlet serves up any resource file packaged in the WAR file. This option is set to true by default, and should be set to true for the Webbank application.

For the case where HTML pages are served by the Web server, as opposed to being served by the WebSphere, there may be an increase in performance, since the Web server is serving the pages directly. Moreover, a Web server has much more customization options than the file servlet can offer. However, using the WebSphere file serving servlet has the advantage of keeping the static content organized in a single deployable unit with the rest of the application. Additionally, this allows you to protect the static pages using WebSphere security.

Attributes used by the file serving servlet can be specified in AAT under a Web module's Assembly Property Extensions entry.

Note: WebSphere V3.5 allowed you to protect static resources even if they were not served by the application server. This is not possible anymore.

Please refer to 19.9, "Separating static content from dynamic content" on page 718 to learn more about deploying the static content of an application to a Web server.

18.12.2 Web application auto reload

If you check the **Reloading enabled** option, the classpath of the Web application is monitored and all components (JAR or class files) are reloaded whenever a component update is detected. (The Web module's classloader is brought down and restarted.) The reload interval is the interval between reloads of the Web application. It is set in seconds.

The auto reload feature plays a critical role in hot deployment/dynamic reload of your application. Please refer to 19.8, "Dynamic and hot deployment" on page 718 for more details.

This option is set to true by default, with the reload interval set to 3 seconds. In production mode, you should make the reload interval much higher.

18.12.3 Serve servlets by class name

The invoker servlet can be used to invoke servlets by class name. Note there is a potential security risk with leaving this option set in production; it should be seen as more of a development-time feature, for quickly testing your servlets.

This option is off by default.

18.12.4 Default error page

This page will be invoked to handle errors if no error page has been defined, or if none of the defined error pages matches the current error.

18.12.5 Directory browsing

This boolean defines whether it is possible to browse the directory if no default page has been found. By default, this option is set to true.

18.12.6 JSP attributes

The following options can be set for the JSP compiler:

- keepgenerated

  If this boolean is set to true, the source code of the servlet created by compilation of a JSP page is kept on the file system. Otherwise, it is deleted as soon as the servlet code has been compiled (only the .class file is available).

- scratchdir

  This string represents the directory in which servlets code will be generated. If this string is not set, code is created under <WAS_HOME>\temp\<hostname>\<application_server_name>\<applicationname>\<webmodulename>.

JSP attributes can be set in AAT under a Web module's Assembly Property Extensions entry.

18.13 Verifying the contents of an archive

Once you are done with packaging an enterprise application, we recommend that you validate its contents by clicking **File** -> **Verify...**. This will check the integrity of the different deployment descriptors.

Tip: Close and then re-open the application archive if AAT reports problems reflecting Webbank classes in its parent command window.

18.14 Packaging recommendations

Here are some basic rules to consider when packaging an application:

- The EJB JAR modules and Web WAR modules comprising an application should be packaged together in the same EAR module.
- When a Web module accesses an EJB module, you should not package the EJB interfaces and stubs in the WAR. Rather, you should express the dependency on the EJB JAR using a MANIFEST Class-Path entry.

18.14.1 Where should common classes go?

Exploring the application server and the J2EE specification reveals five places where classes can be stored so that they can be found by the modules within a J2EE application. Each of these locations is discussed below, as well as whether each location is applicable to utility classes.

1. As loose classes within a Web module's WEB-INF/classes folder.

   These classes are visible only within the same Web module, and this is usually not a valid location because the classes may not have anything to do with the Web module that contains them.

2. As a JAR file within a Web module's WEB-INF/lib folder.

   This is a good place to put utility classes used by the Web module. However, since these JAR files are only visible from within the Web module, this is not a good place for JAR files that are used by multiple modules.

3. As loose classes within an EJB module.

   Although these classes are visible from within other modules that use a manifest file, this is also a weak solution, because the utility classes may not have anything to do with the other code contained in the EJB module.

4. As JAR files or loose classes on the application server's global classpath.

   This appears to be an easy solution, since it makes the utility classes visible to any modules running on the server. However, you should avoid this technique for four reasons:

   – Portability

     By placing the code outside of the application server, these classes or JAR files must be copied along with the application. This technique also requires changing the global classpath of each server it is deployed to. Because each application server has different classloaders and this falls outside the J2EE specification, each server type may require these classes to be handled differently.

   – Visibility

     This technique makes the classes visible to all applications running on the server. Even if the classes are only required by a single application, other applications can inadvertently use them.

   – Maintainability

     This technique forces all applications running on the server to use the same version of the classes. If one application is upgraded to a newer version of the classes, all of the other applications must be upgraded and/or retested.

   – Compatibility

     This technique does not let an application use a different version of any classes that are used by the application server or other applications.

5. As a JAR file at the root of the enterprise application file (EAR file).

   These classes are visible to any module within the application that has a valid manifest file. This is usually the best solution, since it keeps the classes packaged in their own JAR file, which is usable by any modules within the application. This solution does not suffer from any of the problems of using the global classpath.

For the reasons above, you should usually package utility classes as JAR files and place them in the WEB-INF/lib folder, or more likely, directly in the enterprise application, and reference them using a MANIFEST Class-Path entry. This is what we have done with the webbankCommon.jar file.

See also 19.7, “Understanding WebSphere classloaders” on page 711.

19

Deploying an application

The previous chapter has taken you through packaging the Webbank application. In this chapter, you get to deploy it! First, we take you through setting up the right environment for the application, such as creating virtual hosts, data sources, and binding these resources to the Webbank application. Then, we take you through deploying the Webbank application. Next, we explain deploying the client part of the application, by installing the J2EE client. We also describe how to separate the static content from the dynamic content of your application.

Finally, we describe how WebSphere classloaders work, and give you some packaging and maintenance advice.

We assume in this chapter you are familiar with basic WebSphere administrative tasks, such as starting or stopping the administrative server or the administrative console.

In this chapter, you exclusively use the Application Assembly Tool (AAT) and the administrative console. All deployment tasks can be automated using command-line tools as explained in Chapter 23, “Command-line administration and scripting” on page 881.

19.1 Preparing the environment

To successfully deploy the Webbank application, you must define the necessary resources, such as data sources and virtual hosts, and create and populate the Webbank database. Steps in this section are typically performed by the application deployer.

19.1.1 Defining the Webbank virtual host

Web modules need to be bound to a specific virtual host. For our sample, we chose to bind the Webbank Web module to a specific virtual host called webbank_vhost. This virtual host has the following host aliases:

- www.webbank.itso.ibm.com:90
- www.webbank.itso.ibm.com:9090
- www.webbank.itso.ibm.com:443 (for SSL access).

Any request starting with <webbank_vhost_host_alias>/webbank, such as `http://www.webbank.itso.ibm.com:90/webbank/TransferServlet` is served by the Webbank Web application. Details about virtual hosts and how they are used by the Web server plug-in can be found in 14.2, “Virtual hosts” on page 491.

To create the webbank_vhost virtual host:

1. Start the WebSphere administration server and the administrative console.
2. Select the **Virtual Hosts** entry and choose **New...** from the pop-up menu.
3. Supply the following information to create the virtual host.
   - Name

     The virtual host name, that is "webbank_vhost".
   - Aliases

     The list of host aliases which are associated to this virtual host, as listed above. Click **Add** to add an entry to the list, and type in the host alias. After you have entered the three host aliases, you should see a window similar to Figure 19-1.

     Note: A host alias can be associated to one virtual host and one only. This is enforced by WebSphere.
   - MIME Types

     This setting can be found on the Advanced tab. The default MIME types are fine for this application.

4. Click **OK**.

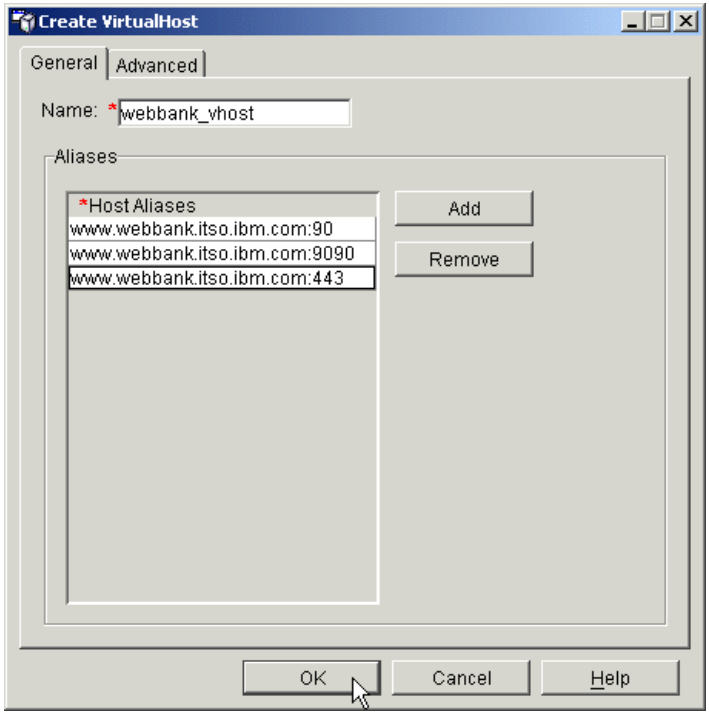


Figure 19-1 Creating the Webbank virtual host

19.1.2 Creating the virtual host for IBM HTTP Server (or Apache)

Now that we have defined a webbank_vhost virtual host, we need to configure the Web server to serve the host aliases defined in the virtual host. We also need to configure the Web server to listen on port 90 as well as configure it for SSL. IBM HTTP Server V1.3.19 is provided with WebSphere, but it is actually built on top of Apache V1.3.20. The steps below are valid for both servers.

Configuring listening ports

For this sample, we have chosen to use the port 90. This is mainly for the sake of demonstrating how it can be done. You can just as well use the default port (80). You can use the Listen directive to tell the Web server to listen to requests on a specific port:

1. Make a backup of the <web_server_install_path>\conf\httpd.conf file.
1. Start your favorite editor, and edit this file.

2. Search for the line containing the `#Listen` string, and add the following line after it:

   ```
   Listen 90
   ```

3. Save the httpd.conf file.

Configuring virtual hosting

Creating virtual hosts is done using the VirtualHost directive, like this:

```
<VirtualHost www.webbank.itso.ibm.com:90>
   ServerAdmin webmaster@itsowebbank.com
   ServerName www.webbank.itso.ibm.com
   ErrorLog logs/webbankHost-error.log
   TransferLog logs/webbankHost-access.log
</VirtualHost>
```

If you want to have multiple virtual hosts for a same IP address, you must use the NameVirtualHost directive, as shown in Example 19-1.

Example 19-1 Using the NameVirtualHost and VirtualHost directives

```
NameVirtualHost 127.0.0.1:90

<VirtualHost itsohost:90>
   ServerAdmin webmaster@itsohost.com
   ServerName itsohost
   DocumentRoot "D:/IBM HTTP Server/htdocs/itsohost"
   ErrorLog logs/itsohostHost-error.log
   TransferLog logs/itsohostHost-access.log
</VirtualHost>

<VirtualHost www.webbank.itso.ibm.com:90>
   ServerAdmin webmaster@webbank.itso.ibm.com
   ServerName www.webbank.itso.ibm.com
   DocumentRoot "D:/IBM HTTP Server/htdocs/webbank"
   ErrorLog logs/webbankHost-error.log
   TransferLog logs/webbankHost-access.log
</VirtualHost>
```

The `www.webbank.itso.ibm.com` and the `itsohost` hosts have the same IP address, that is 127.0.0.1 (the standard loopback interface IP address for our test). We have set this by inserting the following line in the machine `hosts` file (located in %windir%\system32\drivers\etc or in /etc on UNIX systems):

```
127.0.0.1 www.webbank.itso.ibm.com itsohost
```

In a real-life environment, this would probably be done by creating aliases at the DNS level. In any event, you must be able to ping the host you have defined, such as **`ping www.webbank.itso.ibm.com`**.

You must restart the IBM HTTP Server to apply these changes. If you are running a Windows system, we recommend that you start the server by running **apache.exe** from the command line. This allows you to spot error messages thrown at server startup. As you can see in the example below, each virtual host has a different document root. We recommend that you place an index.html file at the document root that tells which virtual host is being called. This lets you easily test which virtual host is being used. If your virtual hosts are correctly configured, invoking `http://itsohost:90` or `http://www.webbank.itso.ibm.com:90` will return different HTML pages.

19.1.3 Creating a JDBC provider and data source

The Webbank application uses a relational database, via entity beans, to store account information. To access this database, you have to define a data source, which you then associate with the entity beans. First, you have to define a JDBC provider, then associate a data source with this driver.

Creating the Webbank DB2 driver

You could very well reuse an existing driver (for example, the one used for the WebSphere samples). We get you to use a different one mainly for education purposes. The following steps take you through the creation of a JDBC provider targeting a DB2 database. To create a JDBC provider from the administrative console:

1. Expand the **Resources** entry, and select the **JDBC Providers** entry.
2. Select **New** from the pop-up menu.
3. You need to specify the following information on the General tab, as shown in Figure 19-2:
   - Name

     The JDBC provider name, such as "WebbankDB2Provider". For clarity, it is recommended that you include the name of the target database in this name.
   - Description

     An optional description of the JDBC provider.
   - Implementation class

     The Java class that provides the JDBC service, such as COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource. A list of supported implementation classes is available if you click the [**...**] button on the right-hand side of this field.

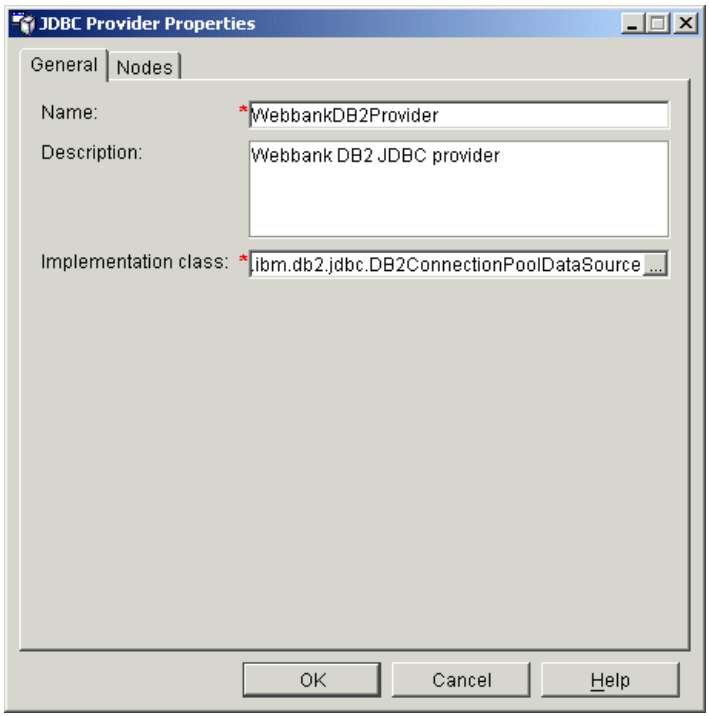


Figure 19-2 Webbank JDBC provider properties

Note: If you need two-phase commit support for this data source, you should select a driver that contains the "XA" keyword. We recommend that you indicate this is a JTA-enabled provider by including the JTA keyword in the provider name, such as JTAWebbankDB2Provider.

Next, you must specify where the implementation class can be found, by installing this JDBC provider on a node.

4. Switch to the **Nodes** tab.
5. Click **Install New...**
6. Select the node on which you want to install this JDBC provider.
7. Click **Specify Driver...**, then **Add Driver...** in the window that pops up.
8. Navigate the file system until you reach the JAR/ZIP file which contains the implementation of the class you have specified above, for example D:\SQLLIB\java\db2java.zip and click **Open**.
9. Click **Set -> Install -> OK**. The JDBC provider is now installed on the node.

Tip: A JDBC provider needs to be installed on each node that will use the provider. If you are using the workload management features of WebSphere and clone the Webbank application on a different node, you will have to install the JDBC provider on this node for the application to work.

Creating the Webbank data source

The next step is to create a data source that uses this JDBC provider. To create a data source, you need to:

1. Select the **Data Sources** folder under WebbankDB2Provider and choose **New...** from its pop-up menu.
2. As shown in Figure 19-3, the following properties can be specified for a data source:
   - Name

     The data source name, which must be unique in the domain. It is recommended that you use a value that indicates the name of the database this data source is used for, such as "WebbankDS".
   - JNDI name

     The name by which applications access this data source. If not specified, this value defaults to the data source name prefixed with "jdbc/". If you specify a JNDI name, we recommend you prefix it with "jdbc/", to respect the naming conventions of the J2EE specification. For the Webbank application, this field must be set to "jdbc/webbank". This value can be changed at any time after the data source has been created.

     Note: To check that a data source has been correctly bound in the server name space, you can use the dumpNameSpace utility. Data sources are bound in the name space as soon as they are created.
   - Description

     An optional (but recommended) description of this data source.
   - Database name

     If applicable, the name of the database you are connecting to. This field is mandatory for DB2 databases, and therefore you must provide WEBBANK (all caps) as the database name.
   - User ID

     The default user ID that is used when a user ID is not specified by the application when obtaining a connection, that is when datasource.getConnection() is called.

– Password

  This value must be provided if a default user ID has been provided.

– Custom Properties

  Depending on the database you are connecting to, you must define one or multiple properties for the data source. For a description of those properties, please see Section 6.6.14.0.1 "Properties of data sources" in the InfoCenter.

3. Click **OK** to create the data source.

The Connection Pooling tab allows you to specify connection pooling properties. Those properties are covered in “Configuring connection pooling” on page 569.

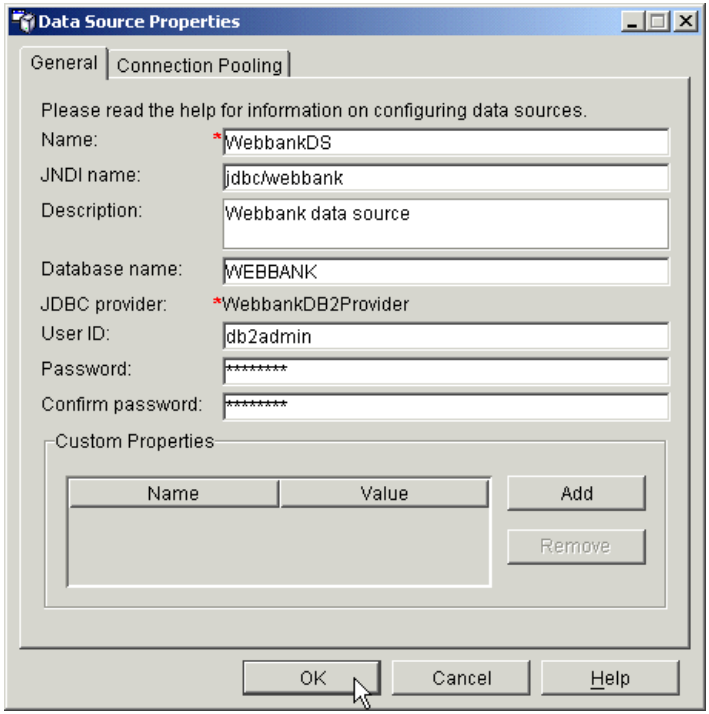


Figure 19-3 Webbank data source properties

19.1.4 Creating the Webbank application server

Now that all resources have been created, you are ready to create the application server where the Webbank application will run. You can set lots of properties for an application server. We cover here only the ones that are not linked to a specific functionality, such as HTTP session management or Web server plug-in configuration. Those settings are covered in their own chapters.

To create an application server, you can select **Console -> New -> Application Server...** and provide information as per Table 19-1. Properties followed by a (*) are mandatory.

We recommend that you start by creating a specific working directory for the Webbank application server, for example:

```
mkdir D:\webbank\WebbankServer01\logs
```

We then created the Webbank application server specifying the following properties and accepting the defaults for others:

- On the General tab, set the Application Server name as WebbankServer01.
- On the General tab, set the Working directory as D:/webbank/WebbankServer01.
- On the General tab, set the Module visibility as Application.
- On the File tab, set theStandard output as logs/stdout.txt.
- On the File tab, set the Standard error as logs/stderr.log.
- On the Services tab, click **Web Container Service -> Transport tab -> HTTP transports -> Edit...** and change the transport port to 9090.

See also 13.2.3, “Creating an application server” on page 461.

Table 19-1 Application server properties

Property Name	Tab Name	Details
Application Server name*	General	The name of this application server, that is WebbankServer01.
Node*	General	The node on which this application server resides.
Environment	General	A list of name/value pairs, which are passed to the JVM (-Dname=value)

Property Name	Tab Name	Details
Working directory	General	This directory is used for multiple purposes. First, it is the relative root for searching files. For example, if you do a File.open("foo.gif"), foo.gif must be present in the working directory to be found. It is also the root for creating the logs files. This directory must exist (it will not be created by WebSphere). We recommend that you create a specific working directory for each application server, for example /projects/webbank.The default working directory is <WAS_HOME>/bin.
Node startup state	General	The state in which this application server should be when the node (that is, the administrative server) is restarted. This behavior can be overridden by using the com.ibm.ejs.sm.adminServer.disableAutoServerStart property in the admin.config file.
Maximum startup attempts	General	The number of times the administration server will try to start this application server before giving up.
Module visibility	General	This setting defines how WebSphere classloaders interact with each other and how classes are shared among classloaders. See 19.7.3, “Setting the module visibility” on page 716 for more details.
Ping interval (seconds)	Advanced	The administration server regularly pings application servers to check whether they are still alive. This setting indicates how often.
Ping timeout (seconds)	Advanced	After ping timeout, the administration server will decide that this application server has problems and will try to restart it.
Ping initial timeout (seconds)	Advanced	Same as above, but for the application server startup. Make sure that Ping initial timeout is greater than your application server initial startup time!
User ID	Advanced	The user ID under which this application server will run. This must be an existing user ID, with sufficient administration rights, such as creating files for logging.
Group ID	Advanced	The group ID under which this application server will run.

Property Name	Tab Name	Details
Standard output	File	The standard output file name. You can specify an absolute path name, or a path relative to the working directory (recommended). For example, specify logs/stdout.txt. Use the File Permissions table to define the rights on these files. If you precede the file name with a ! (such as !logs/stdout.txt), the file is recycled at server startup. Otherwise, the output is appended to the existing file.
Standard error	File	The standard error file name. You can specify an absolute path name, or a path relative to the working directory (recommended). For example, specify logs/stderr.txt. If you precede the file name with a ! (such as !logs/stderr.txt), the file is recycled at server startup. Otherwise, the output is appended to the existing file.
JVM Settings	JVM Settings	The JVM settings page can be used to configure the JVM runtime mode, for example the initial/maximal heap size, JVM system properties (-Dname=value), or the classpath. The Advanced page lets you put the JVM in debug mode for example, or change the bootstrap classpath. You can set most of the JVM -X options from this page.

19.2 Creating application bindings

At packaging time, you have created references to resources. Prior to deploying the application you need to bind these references to the actual resources, such as JDBC data sources, created from the administrative console. This needs to be done for EJB references and resource references. You also need to define the enterprise beans JNDI names, and security roles. Security roles are already covered in 21.2, “Configuring enterprise application security roles” on page 755, and therefore are not covered in this chapter.

Bindings can be defined in the AAT or at deployment time. Both ways are valid. However, if you define the bindings in the AAT, you obtain an EAR file that contains all necessary information for the deployer to install the application in the application server. If necessary, bindings can be overridden using the administrative console or command-line tools at deployment time. In the following sections, we use the AAT to create the bindings definitions. These steps can be performed by the application developer or the application deployer.

All bindings definitions are stored in the ibm-xxx-bnd.xmi files, where xxx can be ejb-jar, web, application, or application-client.

In the next steps, you define the following bindings:

- EJB JNDI names
- Data sources for entity beans.
- EJB References
- Virtual_host bindings for Web modules

All sections below assume you have started the Application Assembly Tool (AAT) and opened the Webbank application EAR file, that is webbank.ear as created in Chapter 18, “Packaging an application” on page 639. If you haven’t already created webbank.ear, use webbank\finished\webbank.ear, provided in webbank.zip. Instructions for finding webbank.zip are in Appendix D, “Additional material” on page 1083.

19.2.1 Defining EJB JNDI names

For each enterprise bean, you must specify a JNDI name. This name is used to bind an entry in the global JNDI namespace for the EJB home object. The bind happens automatically when the application server starts.

All the Webbank enterprise beans are declared to be bound in the "webbank/" subcontext. For clarity, we recommend that you place all enterprise bean JNDI names for an application in a separate subcontext. You can find the JNDI names for each Webbank EJB in Table 19-2. Use this table and the instructions below to define a JNDI name for each Webbank enterprise bean:

1. Select the Transfer session bean, and go to the Bindings page.
2. In the JNDI name field, enter `webbank/Transfer` .
3. Click **Apply**.
4. Repeat for each of the enterprise beans listed in Table 19-2.

Table 19-2 Webbank enterprise bean JNDI names

EJB Name	JNDI Name
Transfer session bean	webbank/Transfer
Consultation session bean	webbank/Consultation
BranchAccount entity bean	webbank/BranchAccount
CustomerAccount entity bean	webbank/CustomerAccount

Tip: To check whether your EJBs were successfully bound in the global JNDI namespace once the application is deployed, you can use the **dumpNameSpace.bat** command. By executing **dumpNameSpace -startAt webbank**, you can see all EJB JNDI entries related to the Webbank application. Since JNDI bindings occur when the application server is started, this list will be empty if the server where you deployed the application is not running.

19.2.2 Defining data sources for entity beans

Both entity beans in our application are container-managed EJBs. The EJB container handles the persistence of the EJB attributes in the underlying persistent store. You must specify which datastore will be used. This is done by binding an EJB module or an individual EJB to a data source. If you bind the EJB module to a data source, all EJBs in that module use the same data source for persistence. If you specify the data source at the EJB level, then this data source is used instead.

For our sample, we chose to specify the data source at the EJB level. This data source (or JDBC resource) has already been created and its JNDI name has been set to "jdbc/webbank" (all data sources are stored by convention in the "jdbc" subcontext).

To bind the CustomerAccount EJB to this data source, you must:

1. Select the CustomerAccount entity bean, and switch to the **Bindings** page.
2. In the Datasource JNDI name field, enter `jdbc/webbank`.
3. You can optionally supply a user ID and password for the data source. These values are used to obtain a connection to the database from the data source. These values have precedence over the user ID and password values supplied when defining the data source.

   In our case, we left these values blank so the data source values are used.
4. Click **Apply**.
5. Repeat for the BranchAccount entity bean.

Note: When dealing with CMP entity beans, you must supply a user ID and password at some point, either at the data source level, or when binding the EJB to a data source to indicate to the container how it should connect to the database (this should not be handled in the code).

19.2.3 Binding EJB references to EJB JNDI names

At packaging time, you have defined EJB references, that is logical names (or "nicknames") for your EJBs. You now have to link those EJB references to the actual EJB JNDI names. Follow these steps to bind an EJB reference to a JNDI name:

1. Select the **EJB References** entry for the Transfer session bean.
2. Select the **ejb/BranchAccount EJB** reference in the top pane.
3. Select the **Bindings** tab on the bottom pane.
4. Enter `webbank/BranchAccount` as the JNDI name.
5. Click **Apply**.
6. Use Table 19-3 to set JNDI names for all EJB references in the Webbank application.

Note: EJB references bindings can be defined at deployment time in the administrative console for all modules except for application clients, for which you must use the AAT.

Table 19-3 EJB references - JNDI Names List

Component	EJB Reference	Corresponding JNDI Name
Transfer session bean	ejb/BranchAccount	webbank/BranchAccount
Transfer session bean	ejb/CustomerAccount	webbank/CustomerAccount
Consultation session bean	ejb/BranchAccount	webbank/BranchAccount
Consultation session bean	ejb/CustomerAccount	webbank/CustomerAccount
webbankWeb Web module	ejb/Transfer	webbank/Transfer
webbankClientApp application client	ejb/Consultation	webbank/Consultation

19.2.4 Binding Web modules to virtual hosts

Web modules need to be bound to a specific virtual host. For our sample, we want to attach the Webbank application to a specific virtual host called webbank_vhost. This virtual host has the following host aliases:

- www.webbank.itso.ibm.com:90
- www.webbank.itso.ibm.com:9090
- www.webbank.itso.ibm.com:443 (for SSL access).

By associating a Web module to a specific virtual host, you tell the Web server plug-in that all requests that match this virtual host, such as `http://www.webbank.itso.ibm.com:90/webbank`, must be handled by the Webbank Web application.

You must bind all Web modules to a virtual host. To do this:

1. Select the webbankWeb Web module, and switch to the **Bindings** page.
2. Supply "webbank_vhost" as the virtual host name.
3. Click **Apply**.

All bindings are now complete.

19.3 Generating deployment code

At some point, you need to generate the deployment code for the Enterprise JavaBeans. You can do this in the application development tool (such as WebSphere Studio Application Developer), in the AAT, from the command line, or at deployment time.

19.3.1 Using the AAT

To generate the EJB deployment code from the AAT:

1. Click **File -> Generate code for deployment...**. A window similar to Figure 19-4 appears.
2. Supply the following information for deployment:
   - Deployed Module Location: the name of the file that will be created.
   - Working Directory: the temporary directory where the EJB JAR will be extracted.
   - Dependent classpath: the list of JAR/ZIP files that the EJBs “depend on”. If an EJB uses a class as a parameter in a method or as a return type, the JAR/ZIP that contains this class must be on the dependent classpath.
   - Verify Archive: check this option if you want to verify the EJB deployment descriptors and archive integrity before deploying the code.
   - Code generation only: if this option is checked, the tool only generates the stubs as skeletons. The RMIC compiler is not run. You should use this option if you want to see the source files for the stubs and skeletons (which are kept in the resulting JAR file).
   - RMIC options: a list of options to be passed to the RMI Compiler (you can run **`<WAS_HOME>/java/bin/rmic.exe -help`** for a list of available options).

Some options (-keep, -iiop for example) are already passed to the compiler and cannot be overridden.

The following options apply to CMP entity beans only:

- Database Type: the type of the database you use for the EJB persistence. This setting affects the generated persistence code.
- Database name: the name of the database you will use for persistence. This setting is not mandatory, and only affects the contents of the DDL file generated at deployment time. This DDL file lets you create the database/tables used by the EJBs, as described in 19.5, “Creating and populating the Webbank database” on page 705.
- Schema Name: the name of the database schema/user. This setting is used both in the generated code and in the DDL file.

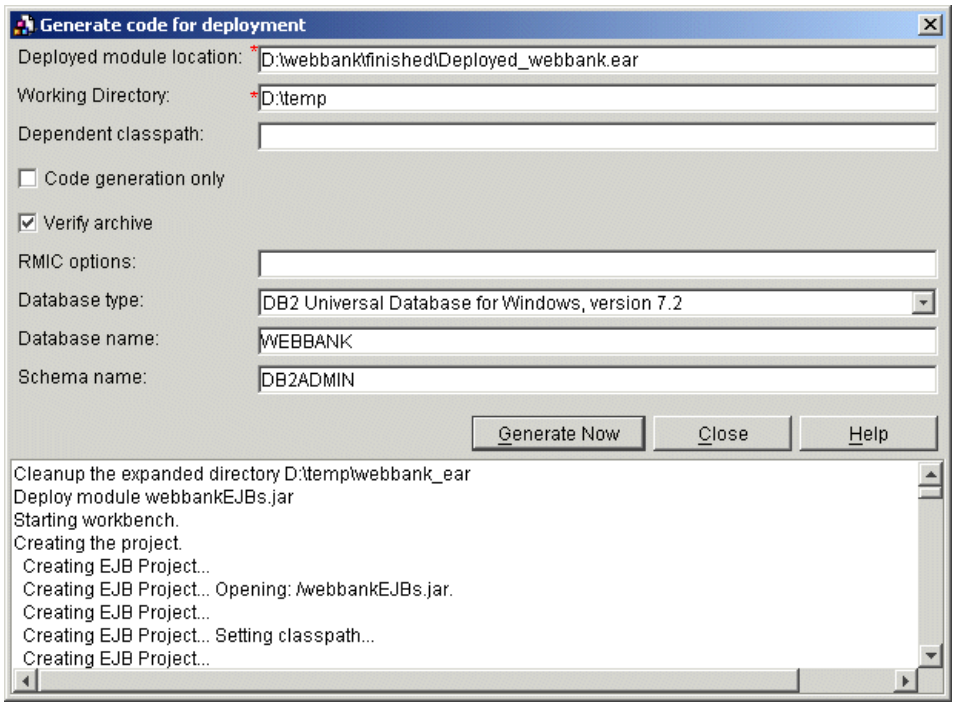


Figure 19-4 Generating the EJB deployment code from the AAT

19.3.2 Using EJBDeploy

You can also generate the EJB deployed code using the EJBDeploy command-line tool. This tool has basically the same options as the ones described for the AAT. The EJBDeploy tool syntax is shown in Example 19-2.

Example 19-2 EJB deploy syntax

```
EJBDeploy
Syntax: EJBDeploy inputJar workingDirectory outputJar [options]
Options:
  -cp "<zip/jar files separated by ;>"  classpath
  -codegen    Only generate the deployment code, do not run RMIC or Javac
  -dbname "Name"      The name of the database to create
  -dbschema "Schema"  The name of the schema to create
  -dbvendor DBTYPE  Set the database vendor type, to one of:
            SQL92 SQL99 DB2UDBWIN_V71 DB2UDBOS390_V6 DB2UDBAS400_V4R5
            ORACLE_V8 INFORMIX_V92 SYBASE_V1192 MSSQLSERVER_V7 MYSQL_V323
  -keep          Do not delete the contents of the working directory
  -ignoreErrors  Do not halt for compilation or validation errors
  -quiet         Only output errors, suppress informational messages
  -novalidate    Disable the validation steps
  -nowarn        Disable warning and informational messages
  -noinform      Disable informational messages
  -rmic "options" Set additional options to use for RMIC
  -35            Use the WebSphere 3.5 compatible mapping rules
  -trace         Enable internal tracing
```

Example 19-3 shows a sample EJBDeploy run using the Webbank EJB module.

Example 19-3 EJBDeploy sample run

```
D:\webbank\finished>ejbdeploy webbankEJBs.jar d:\temp Deployed_webbankEJBs.jar
-cp ..\webbankCommon.jar -dbname webbank -dbschema db2admin -dbvendor
DB2UDBWIN_V71

Starting workbench.
Creating the project.
Importing file..
Starting Validation.
Creating Top-Down Map...
Generating deployment code
Building: /Deployed_webbankEJBs.jar.
Invoking RMIC.
Building: /Deployed_webbankEJBs.jar.
Generating DDL
Shutting down workbench.
EJBDeploy complete.
0 Errors, 0 Warnings, 0 Informational Messages
```

19.4 Deploying the application

We now have an enterprise application file that is ready to be deployed. By default, WebSphere looks for installable applications in the <WAS_HOME>/installableApps directory, so you may want to copy the webbank.ear file there for convenience (the following steps assume you have done so).

We have also have created all the resources that this application uses. Deploying the application is basically a matter of clicking **Next** on every page of the wizard. Each time you click Next, the data on the page is checked (for example, to make sure you use a data source that does exist). Note that this is a non-modal window, which means you can go back to the administration console main window, fix the problem, and continue.

Follow those steps to deploy the application:

1. Select **Console -> Wizards -> Install Enterprise Application** from the administrative console main menu.
2. In the Specifying the Application or Module window, select the **Install Application** option, and click **Browse**. Select the **webbank.ear** file, and click **Next**.
3. The next two windows are related to security. See Chapter 21, “Configuring security” on page 739 for details. You can skip these windows for this example.

   Click **Next** until you reach the Binding Enterprise Beans to JNDI Names window.
4. In the Binding Enterprise Beans to JNDI Names window, you should see the EJB JNDI names you defined earlier. Click **Next**.
5. In the Mapping EJB References to Enterprise Beans window, you should see the EJB references bindings defined earlier. Click **Next**.
6. In the Mapping Resource Reference to Resources window, click **Next**. If the Webbank application had any resource references, you would see the bindings there.
7. In the Specifying the Default Datasource for EJB Modules window, you can define data sources at the EJB module level, but we have chosen to put them at the EJB level. Click **Next**.
8. In the Specifying Data Sources for Individual CMP Beans window, you can define the data sources for the BranchAccount and CustomerAccount EJBs. Both should already be set to jdbc/webbank. Click **Next**.

9. In the Selecting Virtual Hosts for Web Modules window, you can associate Web modules to virtual hosts. The webbank_vhost should already be selected. Click **Next**.
10. In the Selecting Application Servers window, you must define on which application server the EJB and Web modules will run. Select the Webbank EJB and Web modules, and click **Select Server...**. Select the **WebbankServer01** server, and click **OK**. Click **Next**.

Note: For performance reasons, it is recommended that you run all modules from an application on the same application server.

11. In the Completing the Application Installation Wizard window, you are at the final stage before the application is deployed. This window tells you where the application will be installed, namely under <WAS_HOME>/installedApps/<ear_name>. Click **Finish**.
12. You should see a message prompt stating that the deployment code has already been generated and asking if it should be regenerated now. If you have already gone through deployment code generation, as we did, you can click **No**.

Deployment is now complete. Before trying the application, you need to create the Webbank database and populate it with some values.

19.5 Creating and populating the Webbank database

The Webbank CMP entity beans are using DB2 as the underlying persistent store. You must create this database as well as the tables for each of the entity beans. The tables can be created using a DDL file provided in the webbankEJBs.jar file. We assume in the following steps that a DB2 instance is up and running and that you are using the db2admin user to administer your database.

1. Start a command prompt, and start a DB2 command window:

   ```
   db2cmd
   ```

   Perform all actions below from this DB2 command window.
2. Create the Webbank database:

   ```
   db2 create db webbank
   ```
3. Instructions for locating the provided webbank/Table.ddl file are in Appendix D, “Additional material” on page 1083.

Alternatively, if you generated the DDL file when generating deployment code, you can extract the generated Table.ddl file as follows:

```
PATH=%PATH%;%WAS_HOME%\java\bin
jar -xf %WAS_HOME%/installedApps/webbankApplication.ear/webbankEJBs.jar
META-INF/Table.ddl
```

4. Use Table.ddl to create the BranchAccount and CustomerAccount tables:

   ```
   db2 connect to webbank user db2admin using db2admin
   db2 -tvf META-INF\Table.ddl
   ```

   The SQL definition for the BranchAccount and CustomerAccount tables is listed in Example 19-4.

Example 19-4 Table.ddl

```
CREATE TABLE BRANCHACCOUNT
  (BRANCHID VARCHAR(8) NOT NULL,
   BRANCHADDRESS VARCHAR(40),
   BRANCHNAME VARCHAR(30),
   BALANCECENT INTEGER);

ALTER TABLE BRANCHACCOUNT
  ADD CONSTRAINT C4913881 PRIMARY KEY (BRANCHID);

CREATE TABLE CUSTOMERACCOUNT
  (BALANCECENT INTEGER,
   CUSTOMERNAME VARCHAR(40),
   ACCOUNTTYPE VARCHAR(1),
   CUSTOMERID VARCHAR(8) NOT NULL,
   ACCOUNTNUMBER VARCHAR(2) NOT NULL);

ALTER TABLE CUSTOMERACCOUNT
  ADD CONSTRAINT C4801637 PRIMARY KEY (CUSTOMERID, ACCOUNTNUMBER);
```

Next, create some accounts so you can transfer between accounts. The Webbank application does not have a function to create customer or branch accounts. Therefore, you must do it by inserting some rows in the different tables.

5. From the DB2 command window, execute the following lines of SQL to create BranchAccount and CustomerAccount records:

   ```
   db2 insert into branchaccount values ('Sophia','Unknown','Branch
   #Sophia',100000.0)
   db2 insert into customeraccount values (1000.0,'Customer
   #Isabelle','C','Isabelle','A1')
   ```

   Alternatively, use the provided webbank/webbank.cli SQL script (instruction are found in Appendix D, “Additional material” on page 1083) as follows:

```
db2 -tf webbank.cli
```

6. Reset the DB2 connection:

```
db2 connect reset
```

7. Next, start the Webbank application from the administrative console. Locate and right-click the **webbankApplication** in the Enterprise Applications folder and select **Start** from the pop-up menu.
8. Regenerate the Web server plug-in configuration by right-clicking the node and selecting **Regen Webserver Plugin** from the pop-up menu.

You should now be able to make a customer to branch transfer from `http://www.webbank.itso.ibm.com:90/webbank/webbank.html`, as shown in Figure 19-5.

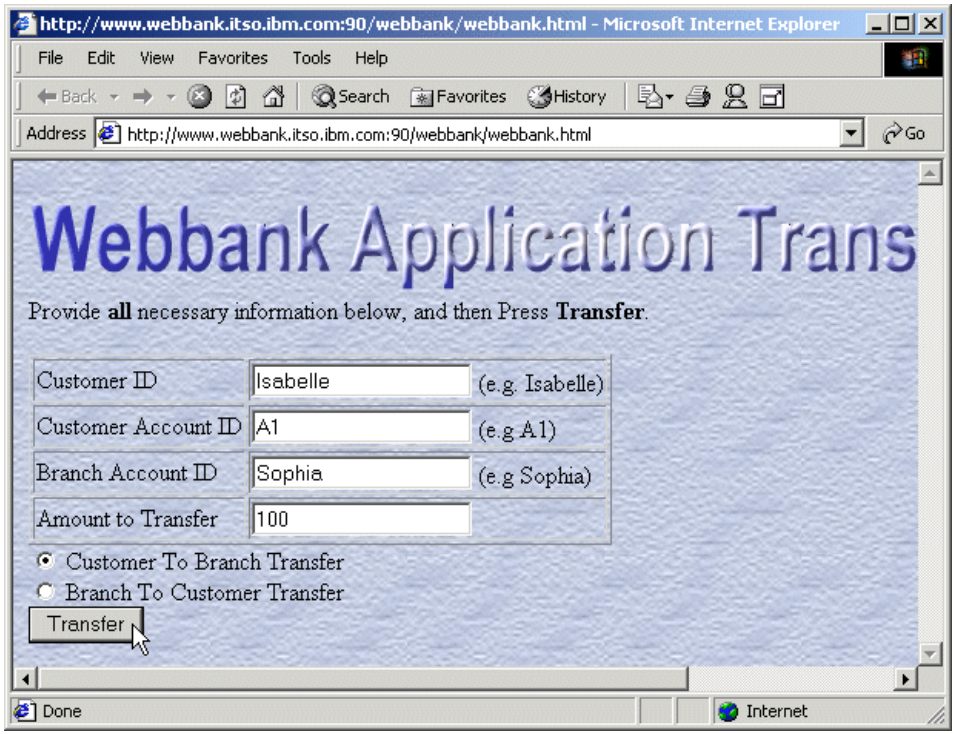


Figure 19-5 WebBank customer to branch transfer

You should get a message indicating that the transfer was completed successfully, as shown in Figure 19-6. Click your browser's **Back** button to try another transfer.

If you get an error indicating that there was a problem processing your request, check that you set the application server Module visibility property to "Application", as described in 19.1.4, “Creating the Webbank application server” on page 695.

Figure 19-6 WebBank successful transfer

19.6 Deploying application clients

There are two types of application clients you can deploy:

- J2EE application client: This client is a Java application program that accesses EJBs, JDBC databases, and Java Message Service message queues. The J2EE application client program runs on client machines. This program allows the same Java programming model as other Java programs; however, the J2EE application client depends on the application client runtime to configure its execution environment, and it uses the JNDI name space to access resources.

  The J2EE application client brings the J2EE programming model to the client, and gives:

  - XML deployment descriptors
  - J2EE naming (java:comp/env) including EJB references and resource references

- Java thin application client: This client provides a lightweight Java client programming model. This client is best suited for use in situations where a Java client application exists but the application must be enhanced to make use of EJBs, or where the client application requires a thinner, more lightweight environment than the one offered by the J2EE application client.

You can use the WebSphere J2EE client and thin client installation programs to install the necessary WebSphere runtime on a client machine, as well as the correct JRE. The J2EE client also comes with the J2EE client container. This J2EE client container is also installed as part of a full WebSphere install. In other words, if you already have installed WebSphere, you do not need to install the WebSphere J2EE client on top. The client install programs can be found on a separate CD in the WebSphere box.

The Webbank application client is a J2EE client. The following steps will take you through the full configuration of this client, assuming you have a J2EE client environment available.

19.6.1 Defining application client bindings

The first thing you need to do is to configure the bindings for the J2EE client. This can only be done in the AAT. Normally, you should already have done this in 19.2.3, “Binding EJB references to EJB JNDI names” on page 700. If not, you can follow the steps in that section to link the ejb/Consultation reference to the webbank/Consultation JNDI name.

19.6.2 Exporting the EAR to the client system

You then have to copy the EAR file to the client machine. Although you do not need the complete contents of the EAR file to run the application client (for example the Web modules), it is better to keep a single EAR file, mainly for maintenance purposes.

19.6.3 Launching the J2EE client

A J2EE client needs a container to run in. This container can be started using the **launchClient** program, which can be found under <WAS_HOME>/bin. The launchClient program has the following syntax:

```
Usage: launchClient [<userapp.ear> | -help | -?] [-CC<name>=<value>] [app args]
```

Where:

<userapp.ear>	The path/name of the .ear file containing the client application.
-help, -?	Print this help message.

Where the -CC properties are for use by the Application Client Runtime:

`-CCverbose`	<true|false> Use this option to display additional informational messages. The default is false.
`-CCclasspath`	A classpath value. When an application is launched, the system classpath is not used. If you need to access classes that are not in the EAR file or part of the resource classpaths, specify the appropriate classpath here. Multiple paths may be concatenated.
`-CCjar`	The name of the client JAR file within the EAR file that contains the application you wish to launch. This argument is only necessary when you have multiple client JAR files in the EAR file.
`-CCBootstrapHost`	The name of the host server you wish to connect to initially. Format: your.server.ofchoice.com
`-CCBootstrapPort`	The server port number. If not specified, the WebSphere default value is used.
`-CCinitonly`	<true|false> This option is intended for ActiveX applications to initialize the Application Client runtime without launching the client application. The default is false.
`-CCtrace`	<true|false> Use this option to have WebSphere write debug trace information to a file. You may need this information when reporting a problem to IBM Service. The default is false.
`-CCtracefile`	The name of the file to write trace information. The default is to output to the console.
`-CCpropfile`	Name of a properties file containing launchClient properties. In the file, specify the properties without the `-CC` prefix. For example: `verbose=true`.

Where `app args` are for use by the client application and are ignored by WebSphere.

If the WebSphere server is local to the client, you can start the Webbank application client, shown in Figure 19-7 on page 711, as follows:

```
LaunchClient /webbank/webbank.ear
```

Otherwise, use the `-CCBootstrapHost` and `-CCBootstrapPort` options to access a remote WebSphere server, for example:

```
LaunchClient /webbank/webbank.ear -CCBootstrapHost=myserver
-CCBootstrapPort=900
```

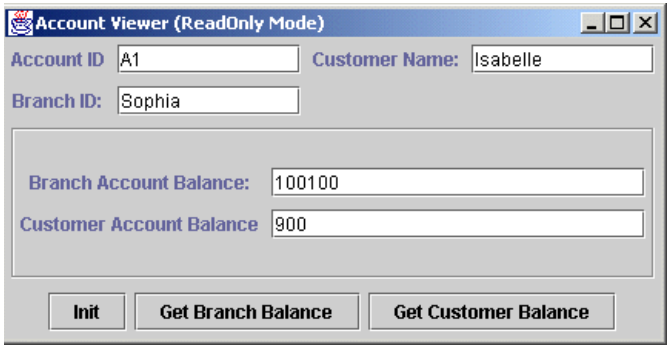


Figure 19-7 WebBank Account Viewer J2EE client

This client can only be used to see the balance of the branch and customer accounts. You need first to supply the account information, then click **Init**. This triggers a findByPrimaryKey() call on the entity beans. Then, you can get the branch and customer account balances.

19.7 Understanding WebSphere classloaders

The classloaders used by WebSphere have been quite deeply changed in WebSphere V4.0. This section starts by giving a little background on Java 2 classloaders and how they work. Then, we describe the different WebSphere classloaders. Finally, we explain how you can customize WebSphere classloaders behavior.

19.7.1 A brief introduction to Java 2 classloaders

Classloaders enable the Java Virtual Machine (JVM) to load classes. Given the name of a class, the classloader should locate the definition of this class. Each Java class must be loaded by a classloader. Classloaders, which are written in Java, are themselves loaded by a specific classloader, the bootstrap classloader. The bootstrap classloader is also responsible for loading the core Java libraries, that is <JAVA_HOME>/lib/rt.jar and <JAVA_HOME>/lib/i18n.jar. This classloader, which is part of the JVM, is written in native code. Classloaders can be extended and customized to include support for new repositories and partitioning of user code on a server.

When you start a JVM, you use three classloaders: the bootstrap classloader mentioned previously, the extensions classloader, and the system classloader.

- Extensions classloader: The extensions classloader is responsible for loading the code in the extensions directories (<JAVA_HOME>/lib/ext). This classloader is implemented by the sun.misc.Launcher$ExtClassLoader class.
- System classloader: The system classloader is responsible for loading the code that is found on java.class.path, which ultimately maps to the system CLASSPATH variable. This classloader is implemented by the sun.misc.Launcher$AppClassLoader class.

Since Java 2, a delegation design has been introduced for classloaders. This is a key concept to understand. It states that a custom classloader delegates class loading to its parent, which can either be another custom classloader or the bootstrap classloader.

The extensions classloader is the parent for the system classloader. The bootstrap classloader is the parent for the extensions classloader. If the system classloader needs to load a class, it first delegates to the extensions classloader, which in turn delegates to the bootstrap classloader. If the parent classloader cannot load the class, the child classloader tries to find the class in its own repository. In this manner, a classloader is only responsible for loading classes that its ancestors cannot load.

This behavior can lead to some interesting problems if a class is loaded from a classloader that is not on a leaf node in the classloader tree. Consider Example 19-5: A class called WhichClassLoader2 loads a class called WhichClassLoader3, which in turn invokes a method called WhichClassLoader4.

Example 19-5 WhichClassLoader 2 and WhichClassLoader3 java source code

```
public class WhichClassLoader2 {

   // Constructor
   WhichClassLoader2() {
   }
   public static void main(String[] args)
                     throws javax.naming.NamingException {

      StringBuffer bootstrapClassPath=
            new StringBuffer(System.getProperty ("sun.boot.class.path"));
      StringBuffer extClassPath =
            new StringBuffer (System.getProperty("java.ext.dirs"));
      StringBuffer systemClassPath =
            new StringBuffer (System.getProperty("java.class.path"));

      System.out.println("\n Bootstrap classpath=\n"
                         + bootstrapClassPath + "\n");
      System.out.println("\n Extension classpath=\n"
                         + extClassPath + "\n");
      System.out.println("\n System classpath=\n"
```

```
                        + systemClassPath + "\n");

      // Loading Classes
      Object obj = new Object();
      WhichClassLoader2 wcl2 = new WhichClassLoader2();
      WhichClassLoader3 wcl3 = new WhichClassLoader3();
      // Who loaded what?
      System.out.println ("Object was loaded by " +
                          obj.getClass().getClassLoader());
      System.out.println ("WCL2 class was loaded by " +
                          wcl2.getClass().getClassLoader());
      System.out.println ("WCL3 class was loaded by " +
                          wcl3.getClass().getClassLoader());

   wcl3.getTheClass();
}
package com.ibm.wss.lge.classloaders;
public class WhichClassLoader3 {

   public WhichClassLoader3() {
   }
   // This method is invoked from WhichClassLoader3.
   public void getTheClass() {
      WhichClassLoader4 wcl4 = new WhichClassLoader4();
      System.out.println ("\n WCL 4 was loaded by: " +
                          wcl4.getClass().getClassLoader());
   }
}
```

If all WhichClassLoaderX classes are put in the system classpath, the three classes are loaded by the system classloader, and this sample runs just fine. Now suppose you package the WhichClassLoader3.class file in a JAR file that you store under <JAVA_HOME>/lib/ext directory; you would then see the output listed in Example 19-6.

Example 19-6 NoClassDefFoundError exception trace

```
Bootstrap classpath=
   D:\WSAD\jre\lib\rt.jar;D:\WSAD\jre\lib\i18n.jar;D:\WSAD\jre\classes

 Extension classpath= D:\WSAD\jre\lib\ext

 System classpath=
      D:\WSAD\jre\lib\indicim.jar;D:\WSAD\workspace\ClassLoaderTests

Object was loaded by null //(Comment: null = bootstrap classpath)
WCL class was loaded by sun.misc.Launcher$AppClassLoader@3dfa4204
WCL3 class was loaded by sun.misc.Launcher$ExtClassLoader@3df84204
java.lang.NoClassDefFoundError: com/ibm/wss/lge/classloaders/WhichClassLoader4
```

```
   at
com.ibm.wss.lge.classloaders.WhichClassLoader3.getTheClass(WhichClassLoader3.ja
va:11)
   at
com.ibm.wss.lge.classloaders.WhichClassLoader2.main(WhichClassLoader2.java:34)
Exception in thread "main"
```

As you can see, the program fails with a NoClassDefFoundError exception, which may sound weird since WhichClassLoader4 is on the system classpath. The problem is that it is on the wrong classpath.

As you can see from the trace, the WhichClassLoader3 class was loaded by the extensions classloader. In fact, the system classloader delegated the load of the WhichClassLoader3 class to the extensions classloader, which in turn delegated the load to the bootstrap classloader. Since the bootstrap classloader could not find the class, it invoked its child classloader, the extensions classloader. The extensions classloader found the class, and therefore loaded it. Now, the extensions classloader needs to load the WhichClassLoader4 class. It delegates to the bootstrap classpath, which can't find the class, then tries to load it itself and does not find it either. A NoClassDefFoundError exception is thrown. Once a class is loaded by a classloader, any new classes that it tries to load will reuse the same classloader, or go up the hierarchy to find a class. A classloader can only find classes up in the hierarchy, not down.

19.7.2 WebSphere classloaders

WebSphere provides several main custom delegated classloaders: the bootstrap classloader, the application extensions classloader, and the numerous application classloaders.

Bootstrap classloader

The bootstrap classloader, whose parent is the system classloader, is created during the activation of the administration server and is responsible for the loading of the class libraries in the following directories:

- <WAS_HOME>\classes (Runtime Class Patches directory, or RCP)
- <WAS_HOME>\lib (Runtime classpath directory, or RP)
- <WAS_HOME>\lib\ext (Runtime Extensions directory, or RE).

The bootstrap classloader is also responsible for loading of the resources, such as database resources, used by the application server. The bootstrap classloader classpath can be modified by setting the ws.ext.dirs system property. The default value of ws.ext.dirs is the following:

```
ws.ext.dirs =
<WAS_HOME>/java/lib;<WAS_HOME>/classes;<WAS_HOME>/lib;<WAS_HOME>/lib/ext;C:/SQL
LIB/java/db2java.zip
```

The RCP directory is intended to be used for E-fixes and other APARs that are applied to the application server runtime. These patches override any copies of the same files lower in the RP and RE directories. The RP directory contains the core application server runtime files. The bootstrap classloader first finds classes in the RCP directory then in the RP directory. The RE directory is used for extensions to the core application server runtime.

Application extensions classloader

The application extension classloader, whose parent is the bootstrap classloader, is also created during the activation of the administration server and is responsible for the loading of the classes in the <WAS_HOME>\lib\app (Application Extensions, or AEX) directory. This AEX directory is intended for class libraries that need to be shared among all J2EE applications installed on the server. These class libraries are not visible to the application server runtime classloaders (RCP, RP, and RE), which are higher in the hierarchy. Therefore, this directory could contain updated versions of common libraries that are present in the core runtime, for example the xerces.jar or xalan.jar, without any risk to compromise the WebSphere runtime integrity.

Note: Unlike most classloaders, the application extensions classloader works in a self-delegating mode. It first looks in its repository and, if it fails to find a class, delegates that task to its parent, the bootstrap classloader.

Application classloaders

An application classloader, whose parent is the application extensions classloader, is created during the startup of an application server. There is one application classloader per module installed per application server. The delegation of class loading depends on the module type: EJB or WAR.

- Web modules:

  For Web modules, the class search order is the module itself -> AEX -> RCP -> RP -> RE. Within the module itself, WebSphere searches for classes in this order:

  - The root of the WAR for class libraries
  - The WEB-INF/classes directory of the WAR
  - The class libraries in the WEB-INF/lib directory
  - The class libraries that are set using the MANIFEST Class-Path directive.

- EJB modules:

  For EJB modules, the class search order is AEX -> RCP -> RP -> RE -> the module itself. Within the module itself, WebSphere first searches the root of EJB jar and then the class libraries that are set using the MANIFEST Class-Path directive.

Changing application classloaders delegation

The application classloader delegation behavior can be changed by defining the following system properties for each application server:

- com.ibm.ws.classloader.warDelegationMode (false by default).
- com.ibm.ws.classloader.ejbDelegationMode (true by default).

These properties can be set to either true or false resulting in the following search order changes from the above:

- **true**: AEX -> RCP -> RP -> RE -> the module itself.
- **false**: the module itself -> AEX -> RCP -> RP -> RE.

19.7.3 Setting the module visibility

The relationship of the application classloaders can be specified using the module visibility property, for each application server. By setting the module visibility, you decide how isolated the modules are, and how your application is partitioned. The supported settings are Module, Application, Server, and Compatibility.

- **Server** visibility allows all application classloaders on the system to have visibility of all other application classloaders in the system. Search order is the same order as when the modules were initialized into the system.
- **Application** visibility allows all classloaders in a J2EE application to have visibility of other classloaders in the same application. Search order is the order the modules are defined in the application.xml for the EAR.
- In **Module** visibility, each module (EAR, JAR, or WAR) has its own unique classloader. Those modules cannot share classes. Visibility to other modules in the application is only achieved when MANIFEST Class-Path entries are added to a module.
- **Compatibility** visibility should only be used to help migrating applications from WebSphere Application Server V3.5.x and V3.0.2.x. In this mode, all EJB module classloaders have visibility of all other EJB module classloaders and all Web application modules have visibility of the EJB classloaders. The search order for the EJB classloaders is determined by the order in which the EJB modules were initialized.

The recommended visibilities are Module and Application.

19.7.4 How classloaders influence packaging

The big question that probably comes to your mind now is: How do I package applications, and where should common classes/libraries go? Well, as usual, it depends.

Here are a few packaging rules that should help you:

1. JAR files (not ZIP files!) used only by one Web module should be added to the WEB-INF/lib directory of that Web module.
2. Common classes that need to be shared among Web modules and EJB modules should be packaged into a separate JAR file. This JAR file should be put in the EAR file, and referenced in the MANIFEST Class-Path of the EJB and Web modules, like this: `Class-Path: webbankCommon.jar.`
3. Classes common to multiple applications should be placed in the AEX directory (loaded by the application extensions classloader). Note that this classloader is not seen by the WebSphere bootstrap classloader, and therefore can be used to share class libraries that are also used by the WebSphere runtime. For example, if you wish to use an XML parser that is different from the one WebSphere uses, you can place the xerces.jar or xalan.jar files in the AEX directory. Only your applications will see them.
4. The system classloader and the WebSphere bootstrap classloader (which are up in the classloader chain) cannot see classes contained in your EJB or Web modules. If you add a JAR file to the system classpath, or the ws.ext.dirs property, it cannot depend on classes within your module.
5. The WebSphere Application Server runtime is loaded by the bootstrap classloader. Because the system classloader is the parent of the WebSphere bootstrap classloader, the classes put in the system classpath are not able to load classes from the WebSphere runtime (including J2EE APIs). If you need a common utility JAR file or database driver that relies on the J2EE runtime, it must be loaded by the WebSphere bootstrap classloader. You should therefore add it to the ws.ext.dirs property, not the system classpath.

19.8 Dynamic and hot deployment

WebSphere V4.0 provides new support for dynamic deployment, that is the ability to deploy new code without stopping the application. The "Hot deployment and dynamic reloading" article in the InfoCenter exhaustively lists which dynamic changes can be made. However, the InfoCenter seems to imply that you have to use **`DrAdmin -restart`** under certain circumstances. **`DrAdmin -restart`** can only be invoked in the Advanced Edition, Single Server. On the full Advanced Edition, you must use **`DrAdmin stop`**, then **`DrAdmin start`** on the module.

Note: The relationship between all the classloaders at runtime makes dynamic reloading sometimes difficult. Some classes will be reloaded but dependent classes might not be, which may lead to ClassCastExceptions. Your safer bet is to reload the complete application. You then have the guarantee that all related classes are reloaded in unison.

19.9 Separating static content from dynamic content

By default, WebSphere serves all the artifacts of an application, should they be static or dynamic. If your application uses a lot of static content, you may want to configure WebSphere so that the static content is served by the Web server instead. This comes at the expense of packaging the static contents of the application separately, and deploying them manually (there is no standard process to do this).

To do this, you must disable the File Serving Servlet. This can be done from the AAT, from the IBM Extensions tab of the Web Module Properties window, by un-checking the **File serving enabled** option.

You must then reinstall the application, and regenerate the plug-in configuration. To check that this has worked, you can look at the plugin-cfg.xml file, where you should see something similar to this, as opposed to a single rule that redirects all `/webbank/*` requests to WebSphere:

```
<UriGroup Name="webbankApplication/webbankWeb_URIs">
   <Uri Name="/webbank/TransferServlet"/>
   <Uri Name="/webbank/*.jsp"/>
   <Uri Name="/webbank/*.jsv"/>
   <Uri Name="/webbank/*.jsw"/>
   <Uri Name="/webbank/j_security_check"/>
</UriGroup>
```

You should also configure the Web Server to recognize the `/webbank` URI. This can easily be done in Apache/IHS using an Alias directive, like this:

```
Alias /webbank/* "D:/webbank/web/"
```

This assumes all the static content of the Webbank application has been deployed in the D:/webbank/web directory.

19.10 Maintenance best practices

The main piece of advice is to keep your EAR files in sync. It is likely that you use a tool such as Rational ClearCase or Intersolv PVCS to manage your code. Basically, you should always be able to retrieve from your configuration management tool the exact configuration that runs in production or test.

You can get out of sync by updating code directly under the <WAS_HOME>/installedApps directory, mainly for hot deployment purposes, as opposed to redeploying an EAR file that contains the latest changes. If you do this, make sure to rebuild an EAR that contains the latest changes, and put this EAR file back in the configuration management system. Note that you can do this by exporting the enterprise application from the administration console (click **Export Application...** from the enterprise application's pop-up menu). This exports the actual contents of the <WAS_HOME>/installedApps/<applicationName> directory. Anything you changed directly on the file system is saved in the exported EAR file.

However, the best way to update an application is definitely to use your configuration management system as the unique place you deploy from, and the EAR file as the deployment unit. This is especially true if you need to install an application on multiple systems, where maintenance can become a nightmare.

You can use the samples that come with this book to create a WSCP script that takes the new EAR file, uninstalls the current application, installs the new version, and restarts the application. See Chapter 23, "Command-line administration and scripting" on page 881 for details.

20

Packaging and deploying Web services

The process used to package and deploy a Web service consists of the following steps:

1. Package your artifact into a JAR file
2. Package your JAR file into an enterprise application archive (EAR)
3. Create a SOAP Deployment Descriptor for your Web service
4. SOAP-enable your Web service enterprise application archive
5. Deploy your Web services EAR to WebSphere
6. Test your Web services

Figure 20-1 illustrates this process.

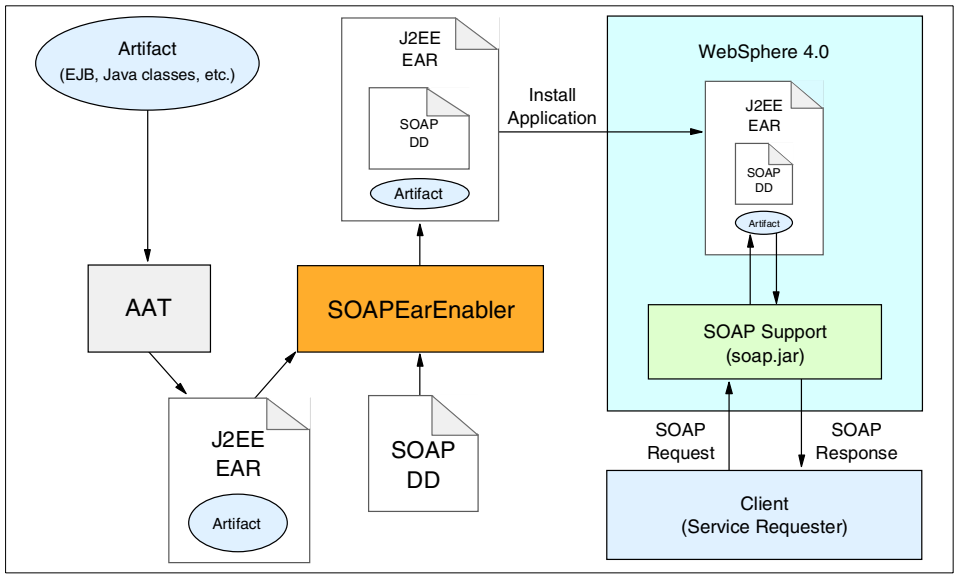


Figure 20-1 Packaging and deploying a Web service

In the following sections, we demonstrate each of these steps using one of the Web service samples supplied with WebSphere Application Server V4.0.

20.1 Package your Web service artifact

The Web service artifact would normally be packaged by the application development team responsible for developing the Web service.

For our example, we will use the stock quote sample that is packaged in the samples.jar file, in the WebSphere SOAP samples enterprise application archive (soapsamples.ear). To access samples.jar, expand soapsamples.ear into a temporary directory.

1. We created a soapsamples directory in the <WAS_HOME>\temp directory, as follows:

```
cd \WebSphere\AppServer\temp
md soapsamples
```

2. We then expanded soapsamples.ear, as follows:

```
..\bin\EARExpander -ear ..\installableApps\soapsamples.ear -expandDir
soapsamples -operation expand -expansionFlags war
```

Note: We invoke EARExpander with the *expansionFlags* option set to *war* so samples.jar is not expanded.

20.2 Package your Web service artifact into an EAR

Create the enterprise application archive (EAR) using the WebSphere Application Server Application Assembly Tool (AAT).

20.2.1 Start the Application Assembly Tool

1. Start the Application Assembly Tool by selecting **Tools** -> **Application Assembly Tool** from the WebSphere Administrative Console main menu, as shown in Figure 20-2 on page 724.

Note: You can start the Application Assembly Tool from the command prompt as follows:

```
<WAS_HOME>\bin\assembly
```

On Windows systems you can also select **Start -> Programs -> IBM WebSphere -> Application Server V4.0 AE -> Application Assembly Tool.**

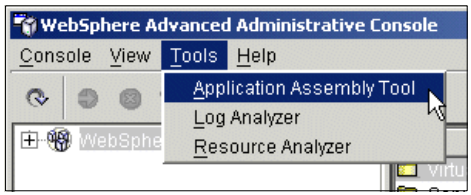


Figure 20-2 Starting the Application Assembly Tool

20.2.2 Package the Web service artifact into an EAR

We package the Web service artifact, in this case a JAR file, in an enterprise application archive (EAR) using the following steps:

1. At the AAT Welcome window, you can create a new application by selecting **Application** from the New tab, as shown in Figure 20-3, and clicking **OK**. Otherwise, you can use **File** -> **New** -> **Application** from the AAT main menu.

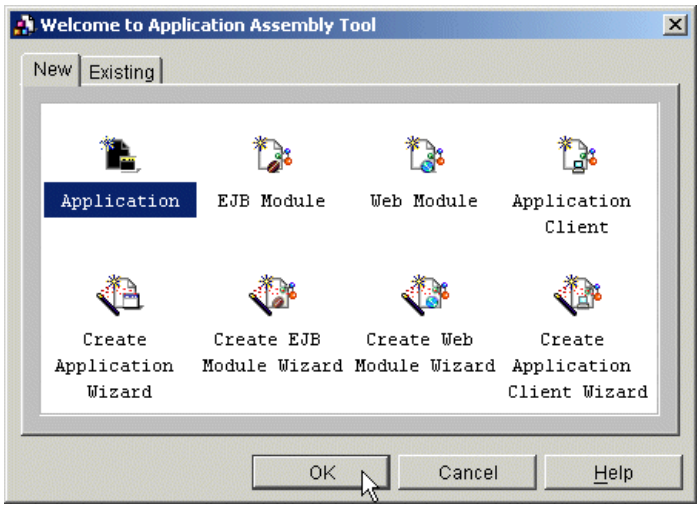


Figure 20-3 Creating a new application

2. Enter the display name and, optionally, a description in the application properties General tab, as shown in Figure 20-4 on page 725, then click **Apply**.

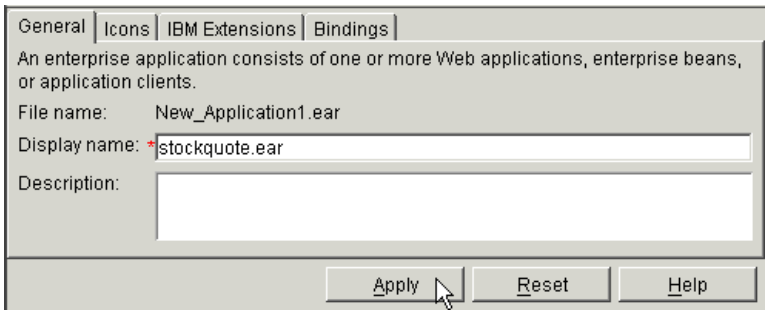


Figure 20-4 Setting the application display name and description

In the next few steps we will add our samples.jar file to the new application.

3. Right-click **Files** and select **Add Files** from the pop-up menu, as shown in Figure 20-5 on page 725.

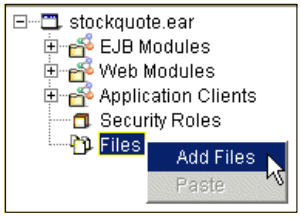


Figure 20-5 Adding a JAR file to the application

4. Click **Browse** in the Add Files window and navigate to <WAS_HOME>\temp, select the **soapsamples** folder, and click **Select**. On returning to the Add Files window, select **samples.jar** in the file pane (on the upper right) and click **Add** to put it in the Selected Files list, as shown in Figure 20-6 on page 726. Click **OK** to add samples.jar to the enterprise application.

Note: Don't try to add a JAR file with the JAR file itself selected as the Root Directory or Archive. The Root Directory or Archive must be either:

- The folder that contains the JAR file, or
- The archive that contains the JAR file.

You can then select the JAR file from the file pane.

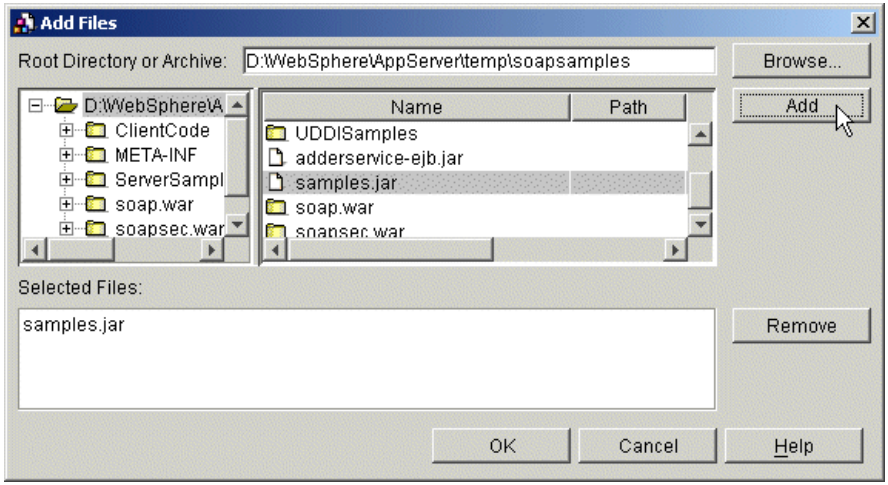


Figure 20-6 Adding samples.jar to the application

20.2.3 Package a Web module archive in the EAR

It is currently a requirement that an enterprise application archive (EAR) file contain at least one J2EE archive. For this reason, we also need to package an empty Web module archive (WAR), even though our application does not include servlets.

1. Select **File -> New -> Web Module** from the AAT main menu.
2. Enter the Display name and, optionally, a description in the Web module properties General tab, as shown in Figure 20-7, then click **Apply**.

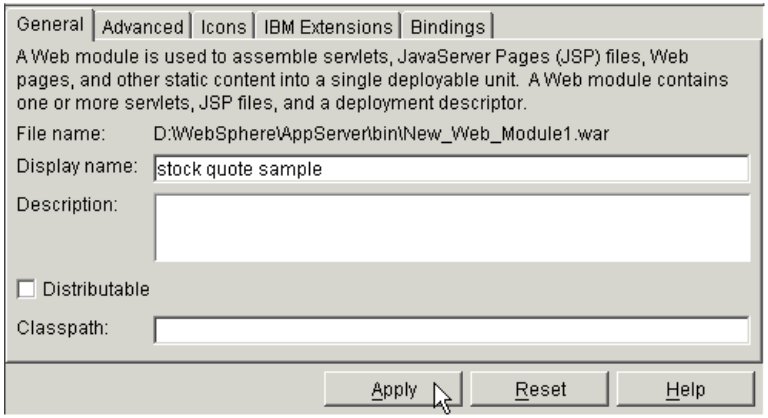


Figure 20-7 Setting the Web module display name and description

3. Now save the Web module by selecting **File -> Save As...** from the AAT main menu. Next, navigate to the <WAS_HOME>\temp\soapsamples directory and save the Web module as stockquote.war.
4. To finish, close the Web module by selecting **File -> Close** from the AAT main menu.

In the next few steps we will add our empty stockquote.war file to the new application.

5. Right-click **Web Modules** and select **Import** from the pop-up menu, as shown in Figure 20-8.

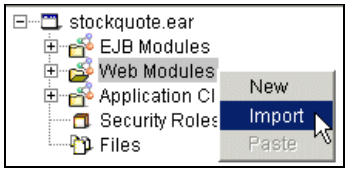


Figure 20-8 Adding a Web module to the application

6. In the Open window, navigate to <WAS_HOME>\temp\soapsamples, select the **stockquote.war** file, and click **Open** to display the Confirm values window. Enter /stockquote for the Context root, as shown in Figure 20-9. Click **OK** to add stockquote.war to the application.

> **Note:** The context root is used to compose the URL user's type for servlet access. In our case we have no servlets, so the context root is completely arbitrary.

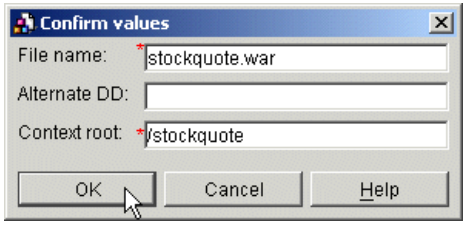


Figure 20-9 Adding stockquote.war to the application

7. Now save the application by selecting **File -> Save As...** from the AAT main menu. Next, navigate to the <WAS_HOME>\temp\soapsamples directory and save the application as stockquote.ear.

8. To finish, close the application by selecting **File** -> **Close** from the AAT main menu. If you leave the application open in AAT, the SOAP EAR Enabler tool will not be able to update.

20.3 Create a SOAP Deployment Descriptor

The contents of the deployment descriptor depend on the type of artifact that is being exposed using SOAP. The deployment descriptors that can be used in WebSphere Application Server V4.0 are:

- Standard Java class deployment descriptor
- EJB deployment descriptor
- Bean Scripting Framework deployment descriptor
- DB2 stored procedure deployment descriptor

In our case, we want to expose a standard Java class, so we use the standard Java class deployment descriptor. The contents of the standard Java class deployment descriptor include:

service urn	The URN that you want to give to the service
provider methods	The list of methods which are being exposed
java class	The fully qualified class name providing the methods that are being exposed

Please refer to the WebSphere InfoCenter if you need the description of one of the other deployment descriptor types.

Figure 20-10 on page 729 shows the deployment descriptor that was supplied with the stockquote sample. For this sample you can see that the service URN is "urn:xmltoday-delayed-quotes", methods is "getQuote", and the class is "samples.stockquote.StockQuoteService".

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
          id="urn:xmltoday-delayed-quotes">
   <isd:provider type="java"
          scope="Application"
          methods="getQuote">
      <isd:java class="samples.stockquote.StockQuoteService"/>
   </isd:provider>
   <isd:faultListener>org.apache.soap.server.DOMFaultListener
   </isd:faultListener>
</isd:service>
```

Figure 20-10 SOAP Deployment Descriptor for stock quote Web service

When we expanded soapsamples.ear in 20.1, “Package your Web service artifact” on page 723, this deployment descriptor was expanded to <WAS_HOME>\temp\soapsamples\ServerSamplesCode\src\stockquote\DeploymentDescriptor\stockquote.xml.

20.4 SOAP enable your Web service EAR

Use the SOAP EAR Enabler tool, SoapEarEnabler, to enable the SOAP services with your enterprise application archive. We used the following steps to SOAP-enable our stockquote.ear enterprise application archive:

1. Make sure you are in the <WAS_HOME>\temp directory:

   ```
   cd \WebSphere\AppServer\temp
   ```

2. Start the SoapEarEnabler tool:

   ```
   ..\bin\SoapEarEnabler
   ```

3. For the stock quote sample, we responded to the SoapEarEnabler tool interactive prompts, as shown in Figure 20-11 on page 730.

```
IBM WebSphere Application Server Release 4
SOAP Enterprise Archive enabler tool.
Copyright IBM Corp., 1997-2001

Please enter the name of your ear file: soapsamples\stockquote.ear

*** Backing up EAR file to: soapsamples\stockquote.ear~

How many services would you like your application to contain (1...n)?1

Now prompting for info for service 1:
Please enter the file name of the SOAP deployment descriptor xml file:
soapsamples\ServerSamplesCode\src\stockquote\DeploymentDescriptor\stockquote
.xml
Is this service an EJB; (y = yes / n = no)?n
How many jarfiles are required for this service (0...n)? 1
Classpath requirement 1: Please choose a file: ([1] samples.jar, [2] sto
ckquote.war): 1
Should this service be secured; (y = yes / n = no)?n

Please enter a context root for your non-secured services (e.g. /soap):
/soapsamples

Do you wish to install the administration client?
Warning, you should not install this client in a production ear unless you
intend to secure the URI to it.

Install the administration client; (y = yes / n = no)?y
```

Figure 20-11 SoapEarEnabler tool interactive prompts

Once you have responded to all the interactive prompts, the SoapEarEnabler tool should SOAP-enable your enterprise application archive.

Note: If you get a SaveFailureException, make sure that you don't still have the stockquote application open in AAT. If stockquote is still open, first close it, then run the SoapEarEnabler tool again.

4. To check that samples.jar has been added to the classpath of the Apache-SOAP Web module created by the SoapEarEnabler tool:
   a. Go to AAT and open stockquote.ear by selecting **File -> Open...** from the AAT main menu.
   b. Check that the Apache-SOAP Web module classpath is set to samples.jar, as shown in Figure 20-12 on page 731.

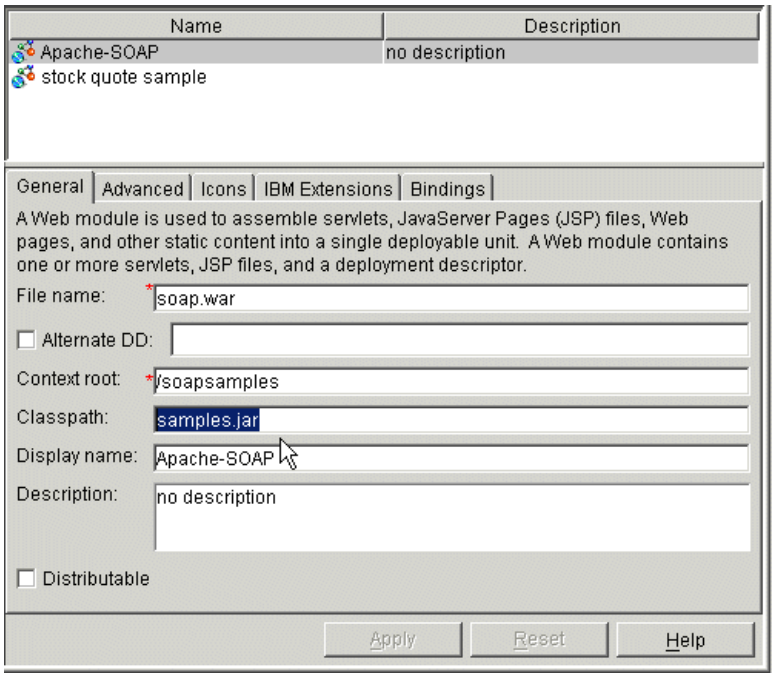


Figure 20-12 Checking the Apache-SOAP classpath

c. Save and close any changes using **File -> Save**, then **File -> Close** from the AAT main menu.

20.5 Deploy your Web services EAR to WebSphere

Use the WebSphere Administrative Console to deploy your Web services enterprise application to your WebSphere Application Server.

1. Start the Install Enterprise Application Wizard by right-clicking **Enterprise Applications** and selecting **Install Enterprise Application** from the pop-up menu, as shown in Figure 20-13 on page 732.

> **Note:** If you already have the WebSphere SOAP samples enterprise application archive (soapsamples.ear) installed on your application server, its context root will clash with our sample's context root. You should remove it from your application server first if you want to deploy our sample. (Right-click the enterprise application and select **Remove** from the pop-up menu.)

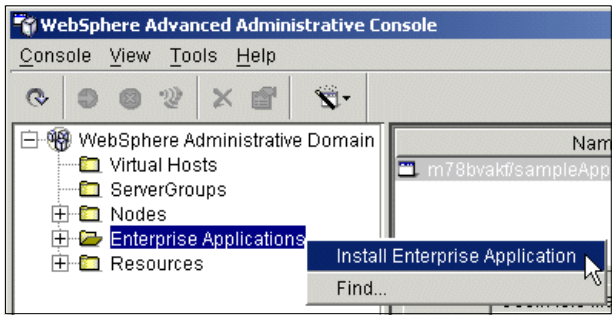


Figure 20-13 Starting the Install Enterprise Application Wizard

2. In the Specifying the Application or Module window, select the **Install Application** option, click **Browse** and navigate to and select <WAS_HOME>\temp\soapsamples\stockquote.ear. Set the Application name to stock quote, then click **Next**.
3. Our application does not use roles, EJBs, or resources so click **Next** to skip the next few windows, until you get to the Selecting the Virtual Hosts for Web Modules windows.
4. Our SOAP enabled enterprise application has two Web modules:
   - Stock quote sample
   - Apache-SOAP

   Specify the virtual host for each Web module using the Selecting the Virtual Hosts for Web Modules window. We used the default_host for both Web modules.
5. Specify the application server(s) where you want to install modules in your application using the Selecting the Application Server window. We used the Default Server for both modules.
6. Confirm your application installation settings by clicking **Finish** in the Completing the Application Installation Wizard window.
7. Start your Web services application by right-clicking it and selecting **Start** from the pop-up menu, as shown in Figure 20-14 on page 733.

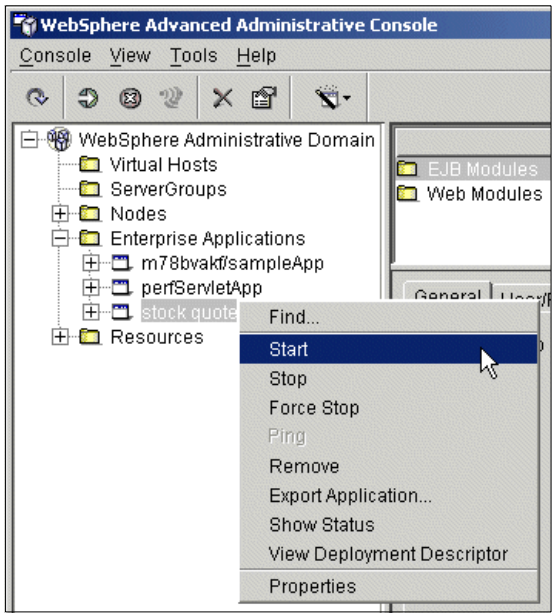


Figure 20-14 Starting your Web services application

8. If your application server does not have Automatic Generation of Plugin Configuration enabled, then you need to manually regenerate the plug-in configuration. You can do this by right-clicking the node and selecting **Regen Webserver Plugin** from the pop-up menu, as shown in Figure 20-15 on page 734.

   You can also manually regenerate the plug-in configuration from the command prompt using:

   ```
   <WAS_HOME>\bin\GenPluginCfg -adminNodeName <Local Node Name>
   ```

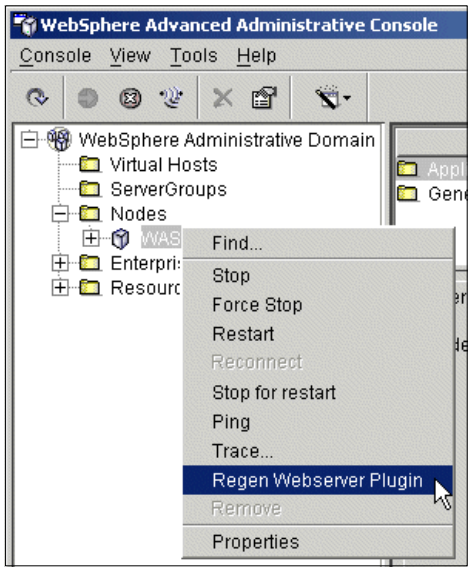


Figure 20-15 Regenerating the Web server plug-in

20.6 Test your Web service

To test the Web service, we first check that the SOAP administration page can be accessed and that the SOAP router servlets can be accessed.

1. Check that the SOAP administration page can be accessed at `http://localhost/soapsamples/admin/index.html`. If you click the **List all services** link, you should see our stock quoting service, as shown in Figure 20-16 on page 735.

Figure 20-16 Checking the SOAP administration page

2. Next check that you can access the SOAP RPC Router and Message Router servlets from your Web browser.

   You should be able to access the SOAP RPC Router at `http://localhost/soapsamples/servlet/rpcrouter`, as shown in Figure 20-17.

Figure 20-17 Checking the SOAP RPC Router

You should be able to access the SOAP Message Router servlet at `http://localhost/soapsamples/servlet/messagerouter`, as shown in Figure 20-18 on page 736.

Figure 20-18 Checking the SOAP Message Router

Next, we check that a client program can access the Web service. We will use the client program supplied with the stock quote sample.

3. In a command window, change to the required sample client folder. We used the Windows sample client which was previously expanded to <WAS_HOME>\temp\soapsamples\ClientCode\nt_bat.

   ```
   cd D:\WebSphere\AppServer\temp\soapsamples\ClientCode\nt_bat
   ```

   Note: Make sure you run this script in the folder described, as the script accesses the parent directory.

4. Run the sample client program from the command prompt. You should receive a current stock quote for the requested ticker symbol. In our examples, shown in Figure 20-19, we received quotes of $108.53 for IBM and $30.19 for Intel.

   If you receive a `connection timed out` error or a `connection refused` error, there may be a problem with your access to the Internet.

   If your receive an `unable to resolve target object` error for samples.stockquote.StockQuoteService, check that samples.jar is in your Apache-SOAP Web module classpath, as described in 20.4, “SOAP enable your Web service EAR” on page 729.

```
StockQuoteSample localhost IBM
108.53
StockQuoteSample localhost INTC
30.19
```

Figure 20-19 Checking the StockQuoteSample client

Note: If you have trouble accessing the Internet through your company's firewall, you could try one of the following options:

- You should be able to set a SOCKS proxy port by adding the following to the JVM command line arguments for the application server:

```
-DsocksProxyHost=<socks server>  -DsocksProxyPort=1080.
```

- You can try changing the stockquote Java code to use a HTTP proxy. Put the following at the beginning of the getQuote method:

```
Properties props = System.getProperties();
props.put("proxySet", "true");
props.put("http.proxyHost", "proxy.sbank.uk.ibm.com");
props.put("http.proxyPort", "8080");
```

20.7 Publishing your Web service to a UDDI registry

Publishing your Web service in a UDDI registry will allow other businesses to find and use it. One of the following methods can be used to publish a Web service to a UDDI registry:

- An application developer can write Java code that uses the UDDI4J API, as provided with WebSphere V4.0, to publish the Web service.

  In this case, the application developer may decide to:

  – Provide a separate publish command-line utility that is run once at deployment time to publish the Web service.

  – Provide code in the Web service itself that checks the UDDI registry and publishes itself if it is not already published.

- A tool, such as the IBM UDDI Explorer available with WebSphere Studio Application Developer, can be used to publish the Web service.

Similar consideration also needs to be given to how the Web service will be unpublished when it comes time to decommission the Web service.

Best practices for the use of UDDI registries are still very much evolving at this time.

For more information, see the *Web Services Wizardry with WebSphere Studio Application Developer*, SG24-6292 redbook (publication due in late 2001), or the WebSphere Studio Application Developer help view on application development documentation.

Note: You can access a SOAP Web service without using a UDDI registry, as long as you already have the Web service description and host name.

21

Configuring security

Setting up security involves the following general steps:

1. Setting global values for use by all applications
2. Refining settings for individual applications

In this chapter, we first discuss how you can secure the WebSphere administrative server to restrict administrator access to your WebSphere environment. We discuss the global security settings you need to consider, such as specifying the user repository that WebSphere will use.

We next explain how you configure security for your enterprise application. We define security roles, and configure the Web module security and user login method. We set up EJB module security, configuring EJB method permissions. We then show you how to grant security roles to users.

Lastly, we explain how to set up IBM SecureWay LDAP directory with WebSphere security, and how to use client certificate-based authentication with the IBM HTTP Server.

21.1 Securing the administrative server

In this section we describe the following tasks:

- Enabling WebSphere security
- Selecting the authentication mechanism
- Selecting local operating system authentication
- Selecting LDAP authentication
- Selecting custom user registry authentication
- Securing only the administrative server
- Unsecuring the administrative server

21.1.1 Enabling WebSphere security

Use the following steps to enable WebSphere security:

1. Start the Security Center by selecting **Console** -> **Security Center...** from the WebSphere Administrative Console main menu, as shown in Figure 21-1.

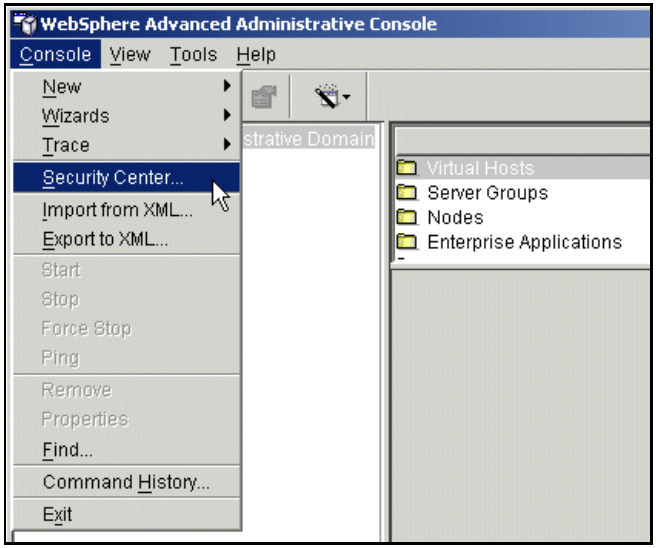


Figure 21-1 Starting the Security Center

2. When the Security Center starts, check the **Enable Security** option, as shown in Figure 21-2.

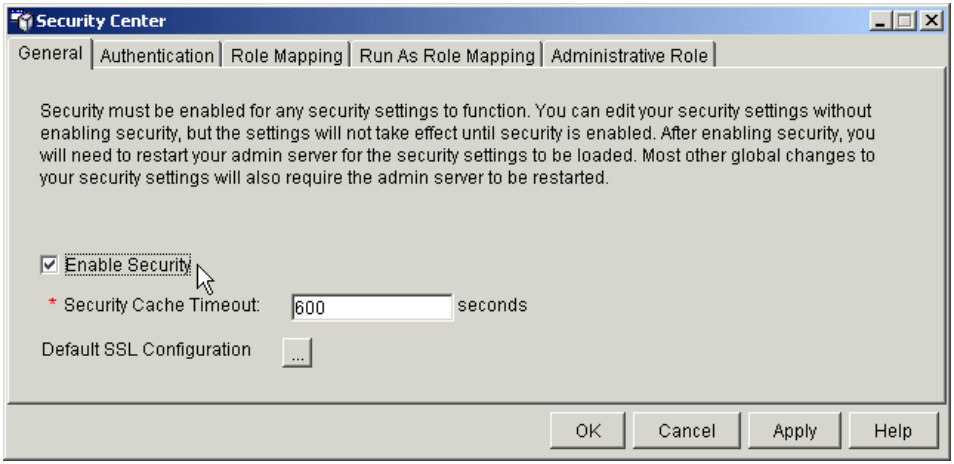


Figure 21-2 Enabling WebSphere security

Note: See 21.9, “Global default SSL configuration” on page 830 for details on setting up the global Default SSL Configuration.

21.1.2 Selecting the authentication mechanism

You now need to select an authentication mechanism. The authentication mechanisms supported by WebSphere are:

- Local operating system authentication
- Lightweight Third Part Authentication (LTPA) using:
  - LDAP directory
  - Custom user registry

In the following sections we describe how each of these mechanisms can be configured.

21.1.3 Selecting local operating system authentication

Use the following steps to select the local operating system as the WebSphere security authentication mechanism:

1. Create the local operating system user to have access to the administrative server. In Windows 2000:
   a. Right-click **My Computer** then select **Manage** from the pop-up menu.

b. From the Computer Management tree, select **System Tools** -> **Local Users and Groups** -> **Users**.

c. Select **Action** -> **New User...** from the main menu, then create your new user.

2. In the Security Center, select the **Authentication** tab and choose **Local Operating System** as the Authentication Mechanism. Set the Security Server ID and password to the local operating system user ID and password you want to use to access the administrative server. (This user ID does not need any special operating system privileges or group memberships.) Our settings are shown in Figure 21-3.

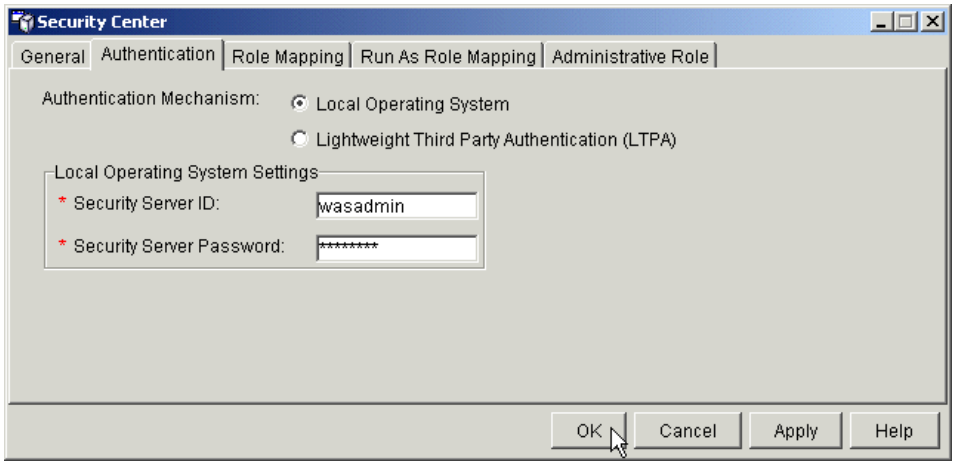


Figure 21-3 Selecting Local Operating System Authentication

3. Click **OK**.

> **Note:** The user ID and password will be verified against the local operating system when you click OK. If you get an error message when clicking OK, make sure all the settings are correct.

4. Restart the administrative server by right-clicking the node in the administrative console, and selecting **Restart** from the pop-up menu, as shown in Figure 21-4.

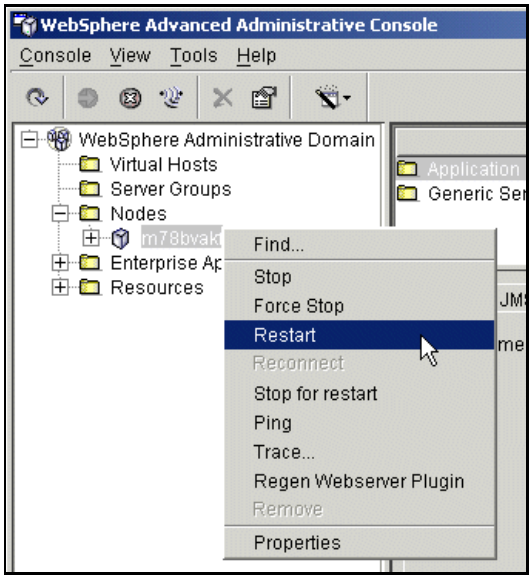


Figure 21-4 Restarting the administrative server

5. When the server is “open for e-business”, start the administrative console. You should be prompted for the administrative user ID and password, as shown in Figure 21-5.

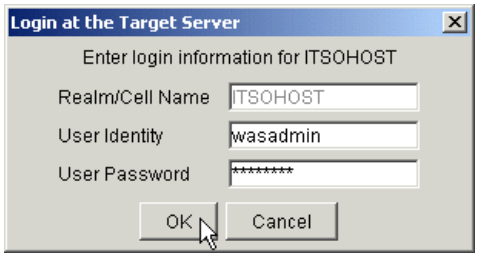


Figure 21-5 Administrative server login with local operating system authentication

Note: The Windows 2000 Server that we used in this example was a member of a Windows NT domain. We created the wasadmin account as a local user. Searching the user registry to map roles to users failed with a `Login failure` exception. We resolved this issue by changing the “IBM WS AdminServer 4.0” service “Log on as” property from wasadmin to “Local System account”.

See “Using Windows NT or Windows 2000 with Local authorization“ in the InfoCenter for information about working with Windows domains.

21.1.4 Selecting LDAP authentication

Use the following steps to select an LDAP (Lightweight Directory Access Protocol) directory as the WebSphere security authentication mechanism:

1. Set up your LDAP directory and create the LDAP user to have access to the administrative server. See 21.7, “Setting up an LDAP directory” on page 783 for details with the IBM SecureWay Directory.
2. In the Security Center, click the **Authentication** tab and select LDAP authentication as follows:
   - Set Authentication Mechanism to Lightweight Third Part Authentication (LTPA), which is required for LDAP
   - Check the **LDAP** option to display LDAP Settings
   - Under LDAP Settings:
     - Set Security Server ID and Password to the LDAP UID and userPassword you want to use to access the administrative server
     - Set Host to your LDAP server host name
     - Set Directory Type to SecureWay
     - Set Port to your LDAP server port (the default is 389)
     - Set Base Distinguished Name to the LDAP suffix that you wish to search
     - Leave Bind Distinguished Name and Bind Password blank and WebSphere will bind anonymously to the LDAP server
     - Click **Advanced...**.

       Changing any of these LDAP advanced properties changes the Directory Type to Custom. You can find more information in 21.8.5, “Using WebSphere security certificate mapping filters” on page 827.
     - Click **Cancel**.

   Our settings are shown in Figure 21-6.

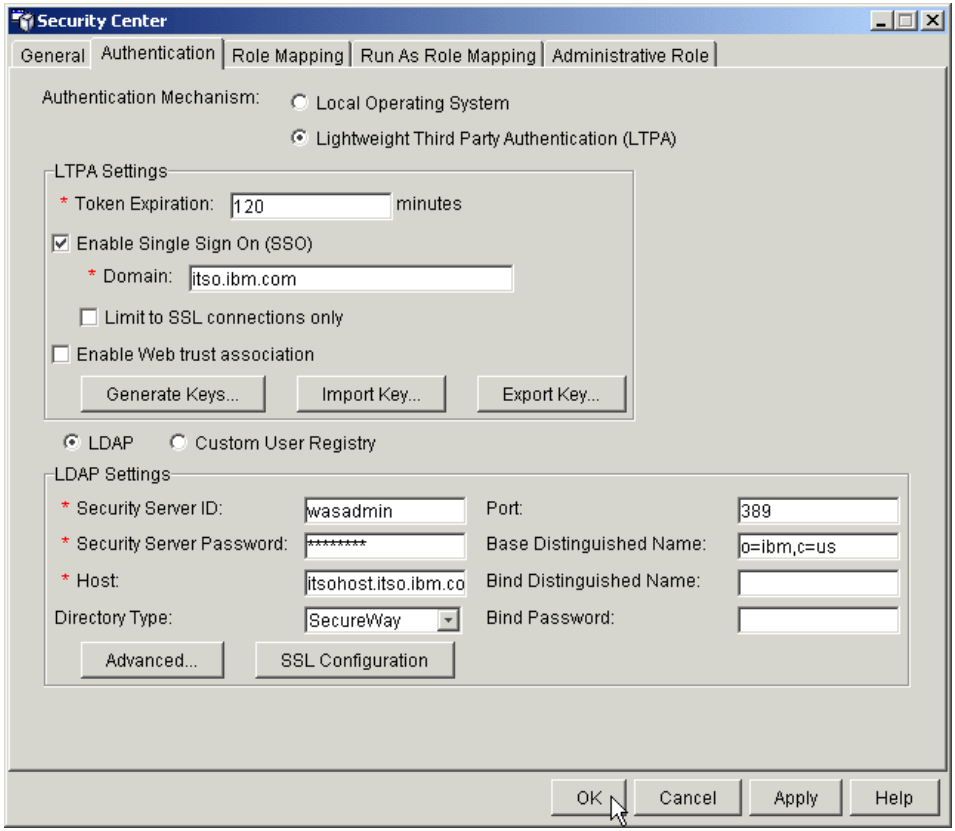


Figure 21-6 Selecting LDAP Authentication

3. Click **OK**.

> **Note:** The user ID and password will be verified against the LDAP server when you click OK. If you get an error message when clicking OK, make sure all the settings are correct and that LDAP is running.

4. If prompted for the LTPA password, as shown in Figure 21-7 on page 746, set the LTPA password to whatever you want.

> **Note:** The LTPA password is the key that is used in the SSO token encryption. Change this password periodically to ensure a secure site, and if you are using SSO, it must be the same for all servers.

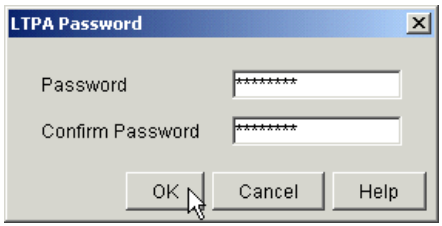


Figure 21-7 Setting the LTPA password

5. Restart the administrative server by right-clicking the node in the administrative console, and selecting **Restart** from the pop-up menu, as shown in Figure 21-4 on page 743.
6. When the server is “open for e-business”, start the administrative console. You should be prompted for the administrative user ID and password, as shown in Figure 21-8.

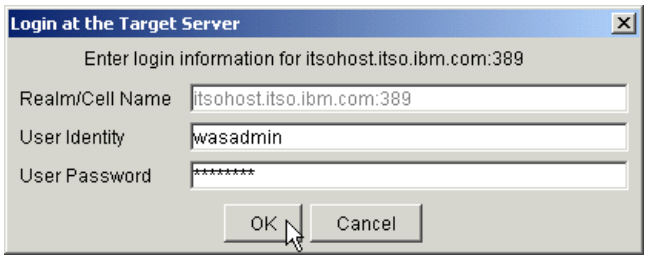


Figure 21-8 Administrative server login with LDAP authentication

21.1.5 Selecting custom user registry authentication

Use the following steps to select a custom user registry as the WebSphere security authentication mechanism:

1. Set up your customer user registry and create the user to have access to the administrative server. For the custom user registry sample provided in the InfoCenter:
   a. Copy the following files to the <WAS_HOME>\temp directory:
      - FileRegistrySample.java (the sample registry implementation)
      - users.props (the user information in the registry)
      - groups.props (the groups information in the registry)

      You can find these files from Appendix D, “Additional material” on page 1083, or in the InfoCenter.
   b. Open a command window and compile the sample:

```
cd D:\WebSphere\AppServer\temp
..\java\bin\javac -classpath ..\lib\websphere.jar
FileRegistrySample.java
```

After compilation, you will have two class files:

- FileRegistrySample.class
- RegExpSample.class

c. Copy the two class files to the <WAS_HOME>\classes directory.

We use this directory for convenience in our sample, because it is already on the administrative server classpath. You would normally add the directory or JAR file containing your custom user registry class files to the classpath by modifying the value of the classpath in the appropriate configuration files, such as admin.config and setupCmdLine.bat/sh.

The sample custom user registry is now ready to use.

Note: This sample custom registry implementation was designed more for simplicity than performance and is intended only to familiarize you with the custom registry feature. An implementation intended for production use requires much better scalability and performance.

2. In the Security Center, click the **Authentication** tab and select custom user registry authentication as follows:
   - Set Authentication Mechanism to Lightweight Third Part Authentication (LTPA), which is required for custom user registry
   - Check **Custom User Registry** option to display Custom User Registry Settings
   - Under Custom User Registry Settings:
     - Set Security Server ID and Password to the custom user registry user ID and password you want to use to access the administrative server
     - Set Custom User Registry Classname to FileRegistrySample

Our settings are shown in Figure 21-9.

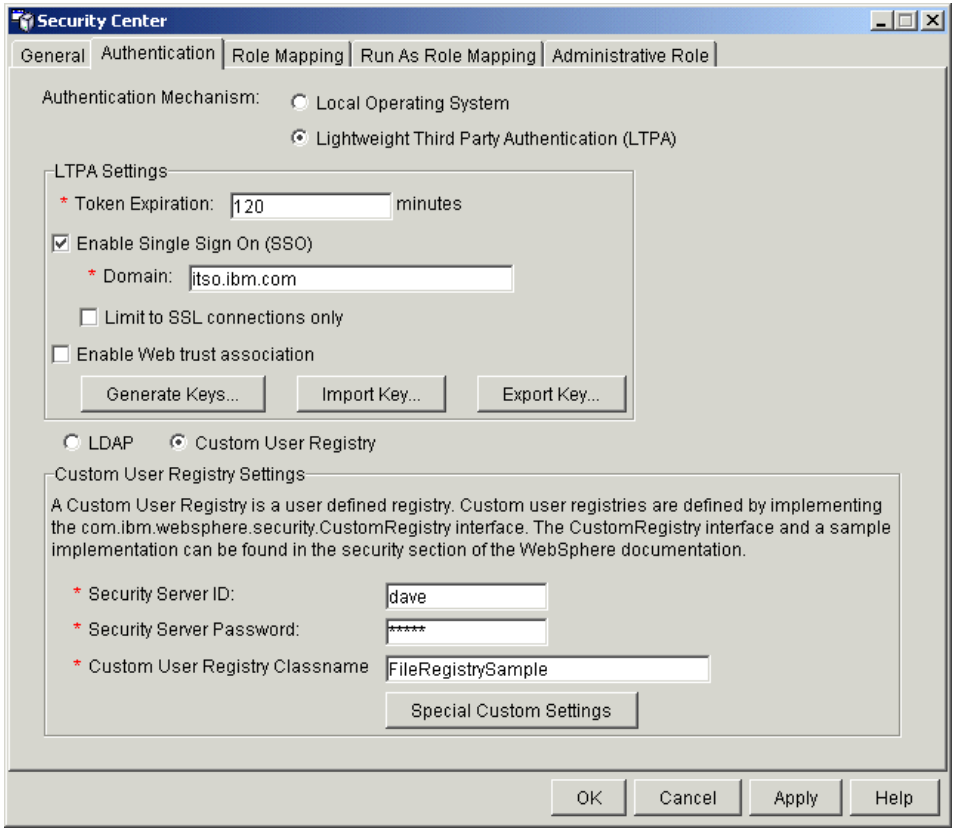


Figure 21-9 Selecting custom user registry authentication

For the FileRegistrySample implementation, two additional properties are needed. They are used for locating the data files that make up the registry.

3. Click **Special Custom Settings** to insert the following custom properties:
   - usersFile <WAS_HOME>\temp\users.props
   - groupsFile <WAS_HOME>\temp\groups.props

   Click **OK** to add the custom properties, as shown in Figure 21-10 on page 749.

   Properties set here are provided to the user registry implementation class during runtime when the initialize method is called.

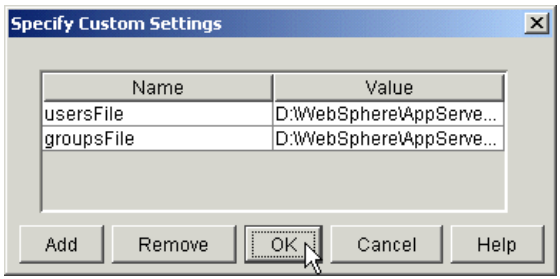


Figure 21-10 Specify Custom Settings

4. Click **OK**.

> **Note:** The user ID and password will be verified against the custom user registry when you click OK. If you get an error message when clicking OK, make sure all the settings are correct.

5. If prompted for the LTPA password, as shown in Figure 21-7 on page 746, set the LTPA password to whatever you want.
6. Restart the administrative server by right-clicking the node in the administrative console and selecting **Restart** from the pop-up menu, as shown in Figure 21-4 on page 743.
7. When the server is “open for e-business”, start the administrative console. You should be prompted for the administrative user ID and password, as shown in Figure 21-11.

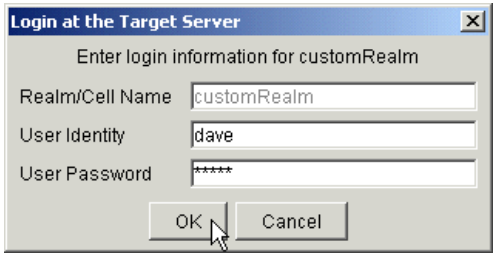


Figure 21-11 Administrative server login with custom user registry authentication

See “Introduction to custom registries“ in the InfoCenter for more information on using custom user registries.

21.1.6 Securing only the administrative server

When you enable WebSphere security, the administrative server and application server(s) on the node are protected by default. With WebSphere V4.0 you can turn off security for selected application servers on a node. You can use this feature if want to protect the administrative server but don't want to impose the overhead of secure encryption on application server communications, for application servers that don't require WebSphere security protection.

Disabling security on selected application servers

In this procedure we show you how to unprotect an application server. There are two parts to this procedure:

- Global settings
- Application server settings

Our example unprotects the Default Server application server, and checks the behavior with the default_app Web module.

Global settings

1. Ensure that you have enabled global security and have restarted the administrative server at least once. This ensures that you have the correct security settings in the sas.server.props file.

   The Snoop Servlet supplied with the default_app has a security constraint allowing only authenticated users to access it. Access to it is unrestricted without security enabled. If you enable security, then try to access the Snoop Servlet at URL `http://localhost/servlet/snoop`, you should get the basic authentication challenge seen in Figure 21-12.

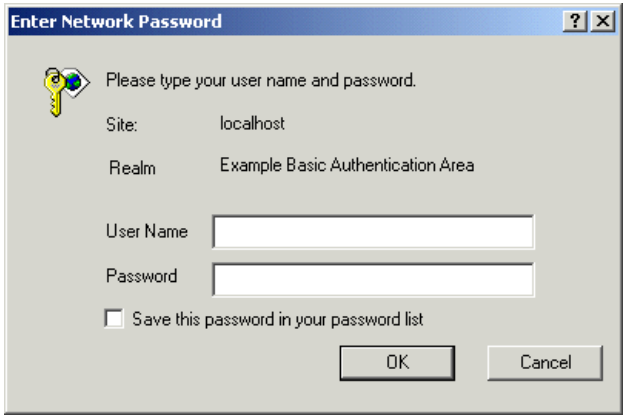


Figure 21-12 Basic authentication challenge with security enabled

2. Stop the administrative server.
3. Delete the <WAS_HOME>\properties\sas.server.props.future file.

   If this file is present when a server restarts, information in the sas.server.props.future file is copied into the sas.server.props file, overwriting your changes to the sas.server.props file.
4. Make a backup copy of the current <WAS_HOME>\properties\sas.server.props file, just in case something goes wrong with the changes.
5. Edit the sas.server.props file and modify the settings as described.

   Important: You must make these changes carefully. Incorrect settings can result in unwanted security behavior, or prevent the administrative server from starting with security enabled. Do not change any values other than the ones listed here unless you are sure of the consequences.

   – If the value of the com.ibm.CORBA.authenticationTarget property is `ltpa`, set the following properties:

     Client-association properties:

     ```
     com.ibm.CORBA.SSLTypeIClientAssociationEnabled=true
     com.ibm.CORBA.LocalOSClientAssociationEnabled=false
     com.ibm.CORBA.LTPAClientAssociationEnabled=true
     ```

     Server-association properties:

     ```
     com.ibm.CORBA.SSLTypeIServerAssociationEnabled=true
     com.ibm.CORBA.LocalOSServerAssociationEnabled=false
     com.ibm.CORBA.LTPAServerAssociationEnabled=true
     ```

   – If the value of the com.ibm.CORBA.authenticationTarget property is `localos`, set the following properties:

     Client-association properties:

     ```
     com.ibm.CORBA.SSLTypeIClientAssociationEnabled=true
     com.ibm.CORBA.LocalOSClientAssociationEnabled=true
     com.ibm.CORBA.LTPAClientAssociationEnabled=false
     ```

     Server-association properties:

     ```
     com.ibm.CORBA.SSLTypeIServerAssociationEnabled=true
     com.ibm.CORBA.LocalOSServerAssociationEnabled=true
     com.ibm.CORBA.LTPAServerAssociationEnabled=false
     ```
6. Restart the administrative server.
7. When the server is “open for e-business”, start the administrative console. You should be prompted for the administrative user ID and password, similar

to that seen in Figure 21-8 on page 746, confirming that administrative server security is still enabled.

Application server settings

1. In the administrative console, right-click the application server you want unprotect, then select **Properties** from the pop-up menu, as shown in Figure 21-13.

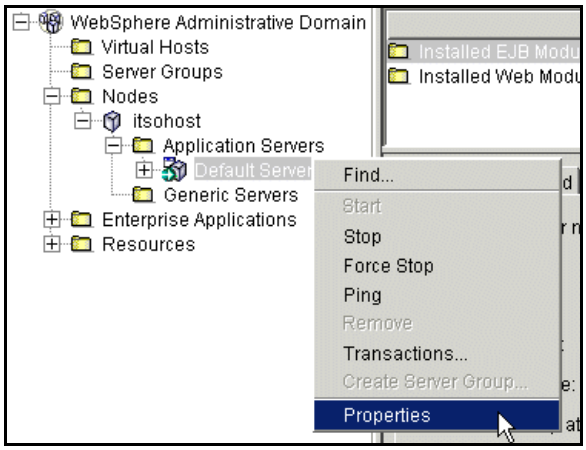


Figure 21-13 Selecting Application Server Properties

2. In the Application Server Properties window, click the **JVM Settings** tab, then set the following System Properties and click **OK**:
   - If the value of the com.ibm.CORBA.authenticationTarget property in sas.server.props is `ltpa`, set the following properties:

     Client-association properties:

     ```
     com.ibm.CORBA.SSLTypeIClientAssociationEnabled=false
     com.ibm.CORBA.LocalOSClientAssociationEnabled=false
     com.ibm.CORBA.LTPAClientAssociationEnabled=false
     com.ibm.CORBA.DCEClientAssociationEnabled=false
     ```

     Server-association properties:

     ```
     com.ibm.CORBA.SSLTypeIServerAssociationEnabled=true
     com.ibm.CORBA.LocalOSServerAssociationEnabled=false
     com.ibm.CORBA.LTPAServerAssociationEnabled=true
     ```

   - If the value of the com.ibm.CORBA.authenticationTarget property in sas.server.props is `localos`, set the following properties:

     Client-association properties:

     ```
     com.ibm.CORBA.SSLTypeIClientAssociationEnabled=false
     ```

```
com.ibm.CORBA.LocalOSClientAssociationEnabled=false
com.ibm.CORBA.LTPAClientAssociationEnabled=false
com.ibm.CORBA.DCEClientAssociationEnabled=false
```

Server-association properties:

```
com.ibm.CORBA.SSLTypeIServerAssociationEnabled=true
com.ibm.CORBA.LocalOSServerAssociationEnabled=true
com.ibm.CORBA.LTPAServerAssociationEnabled=false
```

In our example we are using an LDAP directory so we added the LTPA properties, as shown in Figure 21-14.

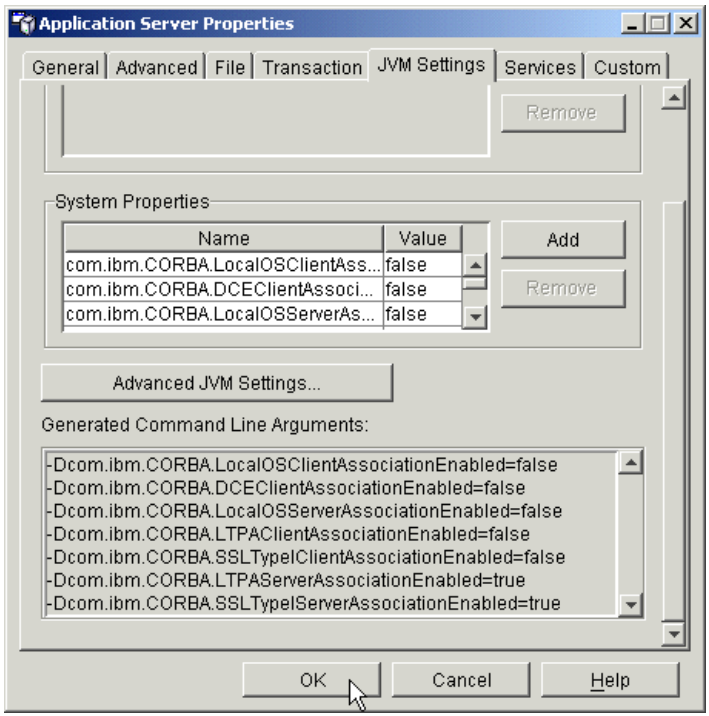


Figure 21-14 Application server properties for unsecuring an application

3. Stop then restart the application server to load the new properties.
4. Check that security is disabled for the application server. You should be able to access the Snoop Servlet at URL `http://localhost/servlet/snoop`, without an authentication challenge when security is disabled.

Repeat these steps for any other application servers that you want to unprotect.

Note: See “Disabling security on specific application servers“ in the InfoCenter for more information. However, we could not disable application server security using a separate sas.appserver.props file, as described in the InfoCenter. Contact WebSphere technical support for details.

21.1.7 Unsecuring the administrative server

What happens if you have enabled WebSphere security but don’t remember the administrative user ID and password? You can look up details on the user repository and user ID in the <WAS_HOME>\properties\sas.server.props file, then reset the password in the user repository. For example, with our WebSphere Application Server configured to use an LDAP directory, we find the following entries in the sas.server.props:

```
com.ibm.CORBA.principalName=itsohost.ral.ibm.com\:389/wasadmin
com.ibm.CORBA.securityEnabled=true
com.ibm.CORBA.loginUserid=wasadmin
com.ibm.CORBA.authenticationTarget=LTPA
```

If your user directory is not accessible, then this approach won’t help. More drastic action may be required.

Important: Take care and use this procedure only as a last resort. Reconfigure WebSphere security immediately!

1. Stop the administrative server.
2. Delete the <WAS_HOME>\properties\sas.server.props.future file. If this file is present when the server restarts, information in the sas.server.props.future file is copied into the sas.server.props file, overwriting your changes to the sas.server.props file.
3. Open the <WAS_HOME>\properties\sas.server.props for editing and set the com.ibm.CORBA.securityEnabled entry to false:

   ```
   com.ibm.CORBA.securityEnabled=false
   ```

   Save your changes.
4. Update EJSADMIN.SECURITYCFG_TABLE in the administrative repository to set securityenabled to 0.

   For a DB2 administrative repository, start a command window and enter the following commands:

   ```
   db2cmd
   db2 connect to was user db2admin
   db2 update ejsadmin.securitycfg_table set securityenabled=0
   ```

```
db2 connect reset
exit
```

5. Restart the administrative server. When the server is back up you should be able to open the administrative console without authentication.
6. Start the Security Center and reconfigure WebSphere security.

21.2 Configuring enterprise application security roles

A security role is a logical grouping of principals. Access to operations (such as EJB methods) is controlled by granting access to a role.

To add a security role:

1. Start the Application Assembly Tool (AAT). You can start AAT by selecting **Tools** -> **Application Assembly Tool** from the administrative console main menu.
2. Select **File** -> **Open...** from the AAT main menu and open your enterprise application archive. We are using webbank.ear in our example. You can find webbank.ear from Appendix D, “Additional material” on page 1083.
3. Right-click **Security Roles** and select **New** from the pop-up menu, as shown in Figure 21-15. Note we are selecting Security Roles at the application level, not at the EJB module or Web module level.

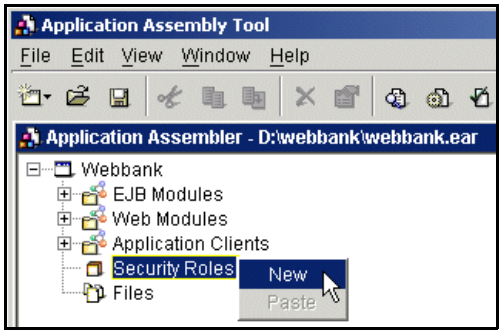


Figure 21-15 Adding a new application security role

4. In the New Security Role window:
   a. Set the Role Name.
   b. Optionally add a description of the security role.

   Our new security role general properties are shown in Figure 21-16.

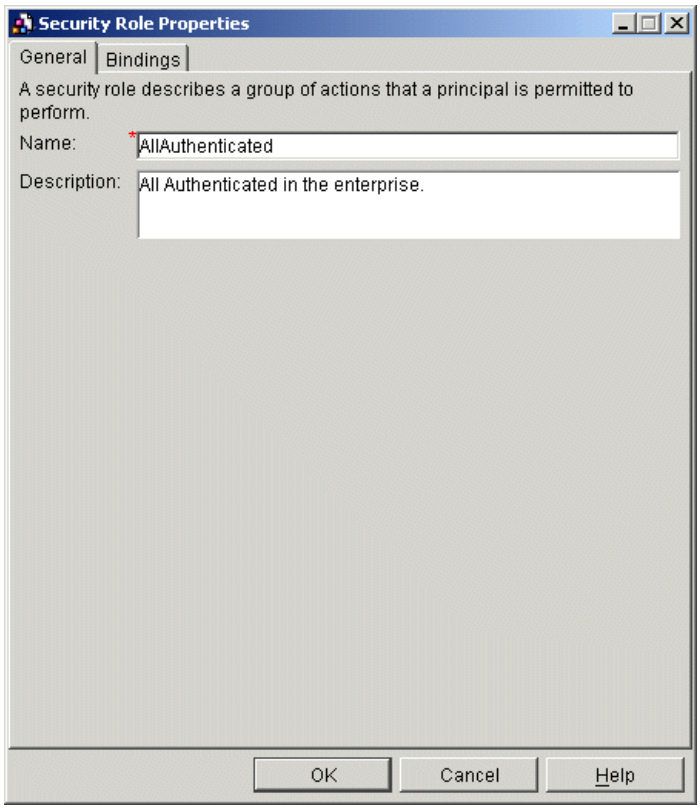


Figure 21-16 Security Roles Properties - General tab

5. Click the **Bindings** tab to set binding information:
   a. Click Groups: **Add...** to grant the security role to user groups.
   b. Click Users: **Add...** to grant the security role to individual users.
   c. Click Special Subjects: **Add...** to grant the security role to one of two special categories of users:
      - Choosing **All authenticated users** grants the role to any user who can authenticate by using a valid user ID and password.
      - Choosing **Everyone** grants the role to all users, including those who did not authenticate. In other words, a method on an enterprise bean or a URI is unprotected if any of the required roles for that method are granted to the special subject Everyone.

      We granted the new security role to All authenticated users, as shown in Figure 21-17.

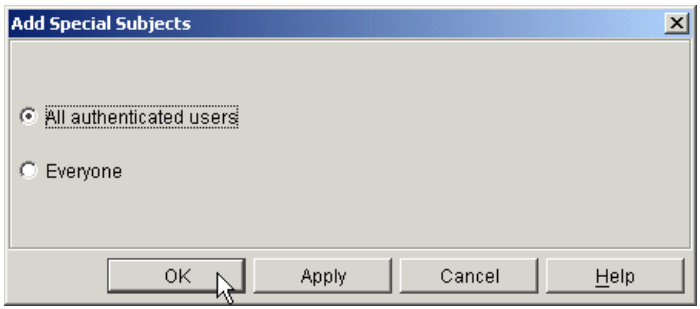


Figure 21-17 Granting a security role to a special subject

Our new security role bindings information is shown in Figure 21-18.

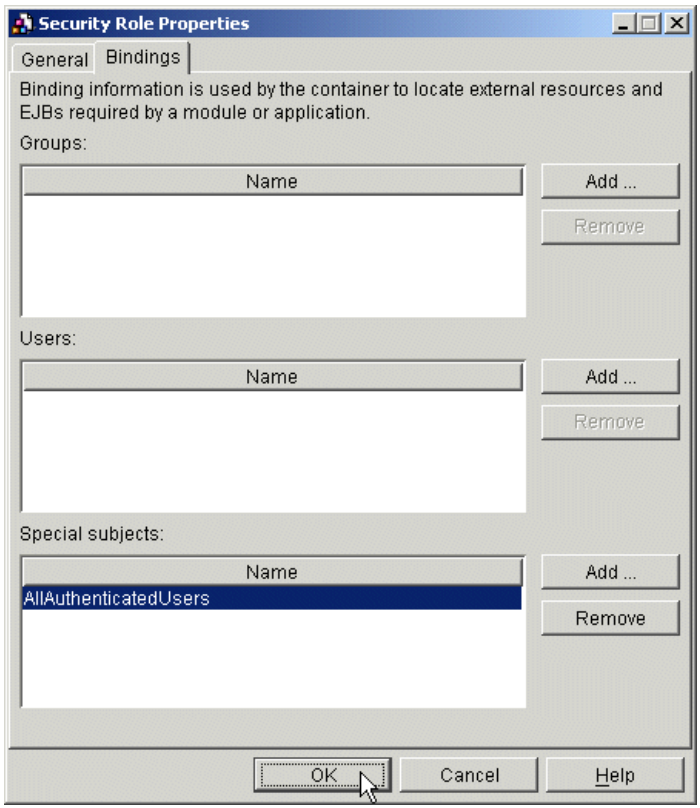


Figure 21-18 Security Roles Properties - Bindings tab

6. Click **OK** to create the new security role.

We also created three other roles for the WebBank application:

- Employee role with no bindings.
- Manager role with no bindings.
- Everyone role with special subject binding of Everyone.

We will specify Employee and Manager role bindings at deployment time, as described in 21.5, “Assigning users/groups to security roles” on page 777.

21.3 Configuring Web module security

In this section we look at the tasks needed to secure Web applications. We use the Application Assembly Tool to:

- Configure security constraints to specify how content should be protected
- Use login configuration to select and configure each of the supported authentication mechanisms:
  - Basic authentication
  - Form authentication
  - Client certificate authentication

21.3.1 Configuring security constraints

Security constraints specify how Web content is to be protected. These properties associate security constraints with one or more Web resource collections. A constraint consists of a Web resource collection, an authorization constraint, and a user data constraint.

- A Web resource collection is a set of resources (URL patterns) and HTTP methods on those resources. All requests that contain a request path that matches the URL pattern described in the Web resource collection are subject to the constraint. If no HTTP methods are specified, then the security constraint applies to all HTTP methods.
- An authorization constraint is a set of roles that users must be granted in order to access the resources described by the Web resource collection. If a user who requests access to a specified URI is not granted at least one of the roles specified in the authorization constraint, the user is denied access to that resource.
- A user data constraint indicates that the transport layer of the client/server communications process must satisfy the requirement of either guaranteeing content integrity (preventing tampering in transit) or guaranteeing confidentiality (preventing reading while in transit).

If multiple security constraints are specified, the container uses the "first match wins" rule when processing a request to determine what authentication method to use, or what authorization to allow.

To add a security constraint:

1. Open your enterprise application archive. We are using webbank.ear in our example. You can find webbank.ear from Appendix D, “Additional material” on page 1083.
2. Double-click to expand **Web Modules**.
3. Double-click to expand the Web module you want to work with. We are using the webbankWeb module.
4. Right-click **Security Constraints** and select **New** from the pop-up menu, as shown in Figure 21-19.

Figure 21-19 Adding a new Web module security constraint

5. In the New Security Constraint window:
   a. Set the Security constraint name.
   b. Click **Add...** and select the user roles that are permitted access to this resource collection, as shown in Figure 21-20.
   c. Select the Transport guarantee.

      Transport guarantee indicates how data communicated between the client and the server is to be protected. The options are:

      - **None** means that the application does not require any transport guarantees.
      - **Integral** means that the application requires that the data sent between the client and the server must be sent in such a way that it cannot be changed in transit.
      - **Confidential** means that the application requires that the data must be transmitted in a way that prevents other entities from observing the contents of the transmission.

In most cases, Integral or Confidential indicates that the use of SSL is required.

d. Optionally add Authorization Constraints and User Data Constraint Descriptions.

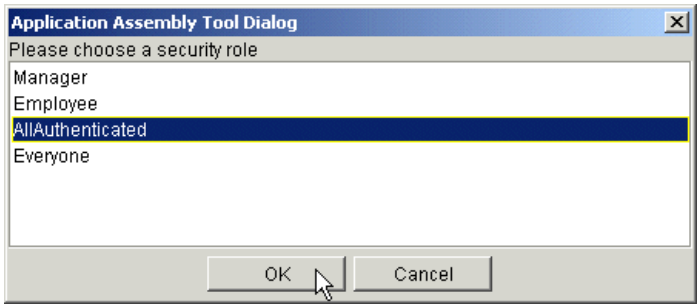


Figure 21-20 Selecting a security role for your security constraint

Our new security constraints properties are shown in Figure 21-21.

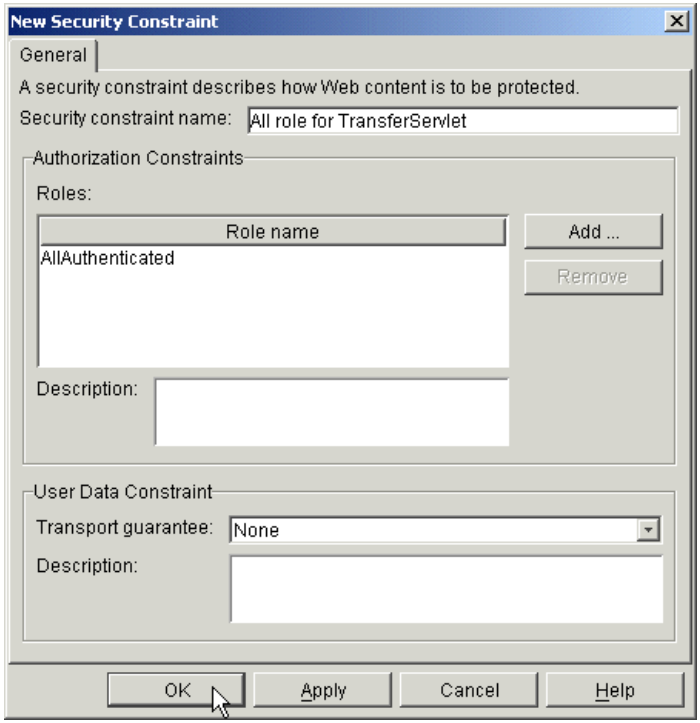


Figure 21-21 New Security Constraint properties

6. Click **OK** to create the new security constraint.

We have just created a new security constraint. We can now associate Web resources with this security constraint.

1. Double-click to expand the new security constraint.
2. Right-click **Web Resource Collections** and select **New** from the pop-up menu, as shown in Figure 21-22.

Figure 21-22 Adding a new Web Resource Collection to your security constraint

3. In the New Web Resource Collection Properties window:
   a. Set the **Web Resource Name**.
   b. Optionally add a description of the Web resource collection.
   c. Click HTTP methods: **Add...** and select the HTTP methods to which the security constraints applies, as shown in Figure 21-23.

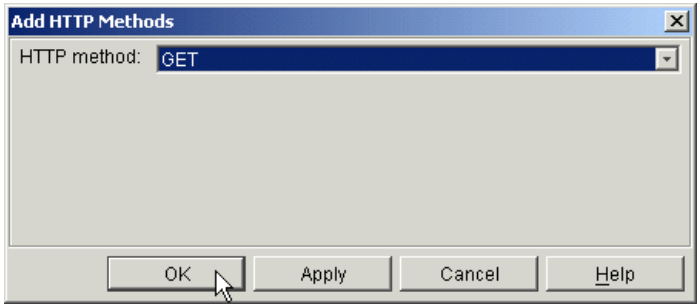


Figure 21-23 Adding an HTTP method for your Web resource collection

   If no HTTP methods are specified, then the security constraint applies to all HTTP methods. The valid values are GET, POST, PUT, DELETE, HEAD, OPTIONS, and TRACE.

   d. Click URLs: **Add...** and set the URL pattern for resources in the Web application, as shown in Figure 21-24.

All requests that contain a request path that matches the URL pattern are subject to the security constraint. Do not to include the context root for the Web module in the URL pattern.

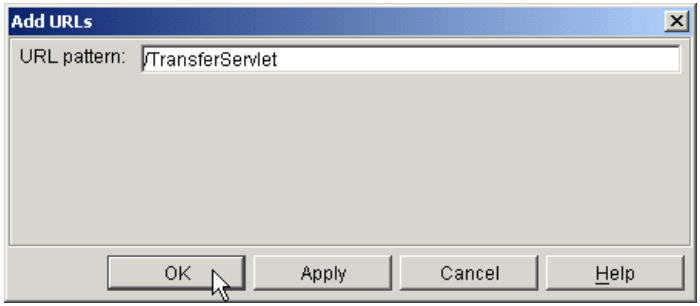


Figure 21-24 Adding a URL pattern for your Web resource collection

Our new Web resource collection properties are shown in Figure 21-25.

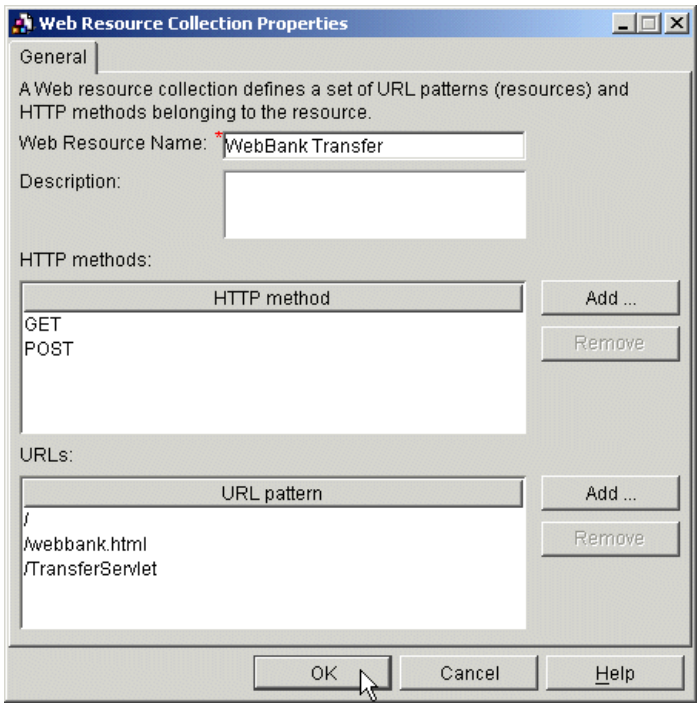


Figure 21-25 New Web Resource Collection Properties

4. Click **OK** to create the new Web resource collection.

Example 21-1 shows the security-constraint element from the web.xml deployment descriptor, for our new *All role for TransferServlet* security constraint.

Example 21-1 "All role for TransferServlet" security-constraint element in web.xml

```
<security-constraint id="SecurityConstraint_1">
   <web-resource-collection id="WebResourceCollection_1">
      <web-resource-name>WebBank Transfer</web-resource-name>
      <url-pattern>/</url-pattern>
      <url-pattern>/webbank.html</url-pattern>
      <url-pattern>/TransferServlet</url-pattern>
      <http-method>GET</http-method>
      <http-method>POST</http-method>
   </web-resource-collection>
   <auth-constraint id="AuthConstraint_1">
      <description>All role for TransferServlet:+:</description>
      <role-name>AllAuthenticated</role-name>
   </auth-constraint>
   <user-data-constraint id="UserDataConstraint_1">
      <transport-guarantee>NONE</transport-guarantee>
   </user-data-constraint>
</security-constraint>
```

21.3.2 Login configuration

To gain access to Web resources protected by an authorization constraint, a user must authenticate using the configured mechanism. Web applications can authenticate a user to a Web server using one of the following authentication methods:

- **Basic** authentication uses the model where the Web server requests the Web client to provide a user name and a password for authentication, in each realm. HTTP basic authentication is not a secure protocol because the user password is transmitted with a simple Base64 encoding and the target server is not authenticated.
- **Digest** authentication transmits the password in encrypted form. Digest authentication is not supported by WebSphere V4.0.
- **Form** authentication allows the Web developer to control the appearance of login windows using custom HTML login forms.
- **Client certificate** authentication uses HTTPS (HTTP over SSL) and requires the user to possess a public key certificate.

Not all login configurations are supported by all of the WebSphere global security authentication mechanisms. The authentication mechanisms are the local operating system and LTPA with an LDAP directory or a custom user registry. Table 21-1 shows which authentication mechanisms are capable of supporting the various authentication methods.

Table 21-1 Supported authentication methods

Authentication method	Local OS	LTPA (LDAP, Custom)
Basic	Yes	Yes
Form	Yes	Yes
Client certificate	Not supported	Yes
Digest	Not supported	Not supported

The Login configuration properties are used to select and configure the authentication method that should be used.

Note: The version of AAT we used (V4.0.1) incorrectly tags an unspecified Authentication method as "All methods". Contact WebSphere technical support for details.

21.3.3 Configuring Basic authentication

To configure basic authentication for a Web module:

1. In AAT, right-click your Web module and select **Properties** from the pop-up menu, as shown in Figure 21-26. We are using the webbankWeb Web module.

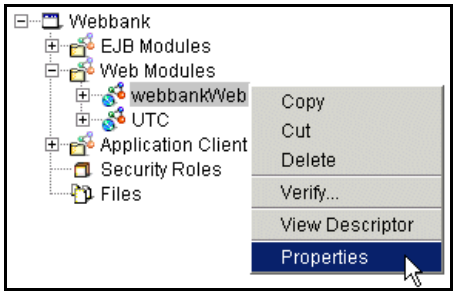


Figure 21-26 Selecting Web module properties

2. In the Web Module Properties window click the **Advanced** tab to set login configuration properties:

a. Check the **Login configuration** box.

b. Set Authentication method to Basic.

c. Specify a realm name.

   An HTTP realm is a string that allows URIs to be grouped together. For example, if a user accesses a secured resource on a Web server within the “finance” realm, subsequent access to the same or different resource within the same realm does not result in a repeat prompt for a user ID and password.

Our login configuration settings are shown in Figure 21-27.

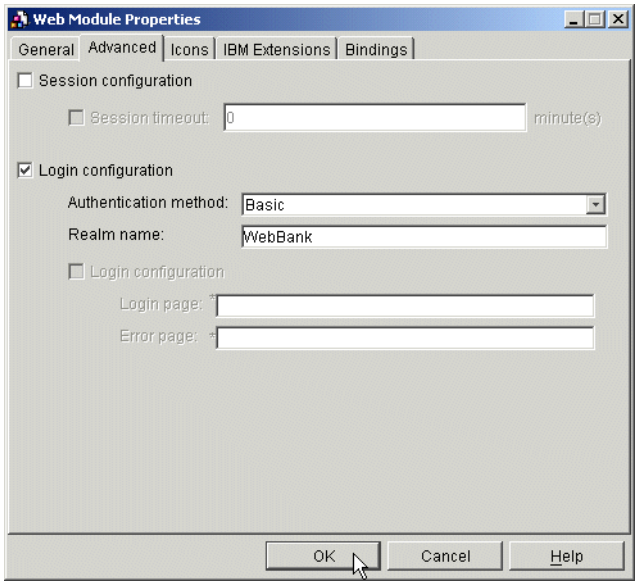


Figure 21-27 Web Module Properties for basic authentication

3. Click **OK** to update the login configuration settings.

Example 21-2 shows the login-config element from the web.xml deployment descriptor, when basic authentication is specified.

Example 21-2 login-config element for basic authentication in web.xml

```
<login-config id="LoginConfig_1">
   <auth-method>BASIC</auth-method>
   <realm-name>WebBank</realm-name>
</login-config>
```

When an unauthenticated user attempts to access a protected resource in the deployed Web module, the basic authentication method uses the basic HTTP authentication challenge, as shown in Figure 21-28.

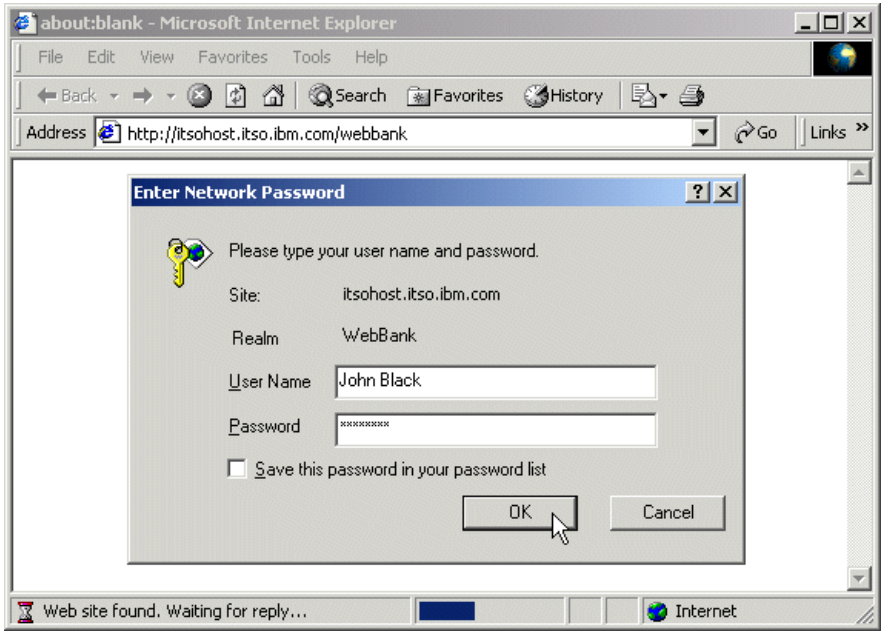


Figure 21-28 Basic authentication challenge

21.3.4 Configuring Form authentication

To configure form authentication for a Web module:

1. In AAT, right-click your Web module and select **Properties** from the pop-up menu, as shown in Figure 21-26 on page 764. We are using the webbankWeb Web module.
2. In the Web Module Properties window click the **Advanced** tab to set login configuration properties:
   a. Check the **Login configuration** box.
   b. Set Authentication method to Form.
   c. Specify a realm name.
   d. Set Login page to the location of the login form.

      The Login page is usually an HTML form with text-entry fields for a user ID and password. As seen in Example 21-4 on page 768, the login page must be implemented as follows:

- The text entry field for the user ID must be named j_username
- The field for the password must be named j_password
- The post action must be j_security_check

The j_security_check post action is recognized by the Web container. It dispatches the action to a special WebSphere servlet that authenticates the user.

e. Set Error page to the location of the error page.

The Error page is displayed if an error occurs during login.

The Login page and the Error page are included in the Web application archive, as shown in Figure 21-30 on page 768.

Our Login configuration settings are shown in Figure 21-29.

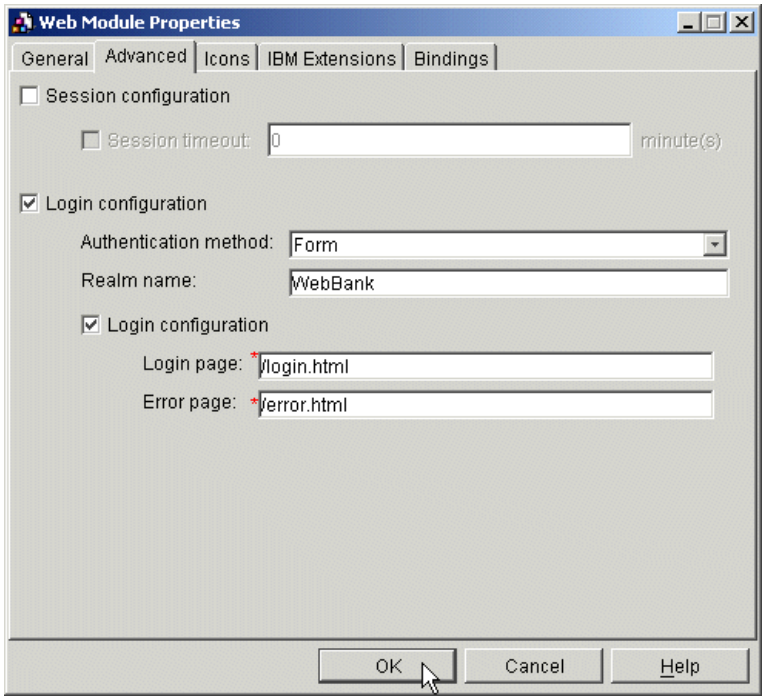


Figure 21-29 Web Module Properties for form authentication

3. Click **OK** to update the login configuration settings.

Example 21-3 shows the login-config element from the web.xml deployment descriptor, when form authentication is specified.

Example 21-3 login-config element for form authentication in web.xml

```
<login-config id="LoginConfig_1">
   <auth-method>FORM</auth-method>
   <realm-name>WebBank</realm-name>
   <form-login-config id="FormLoginConfig_1">
      <form-login-page>/login.html</form-login-page>
      <form-error-page>/error.html</form-error-page>
   </form-login-config>
</login-config>
```

See the HTML snippet given in Example 21-4 for a sample form login page implementation.

Example 21-4 Form login page implementation

```
...
<form method=POST action="j_security_check">
Userid <input type=text name="j_username" size=20><br>
Password <input type=password name="j_password" size=20><br>
<input type=submit name=action value="Submit Login">
</form>
...
```

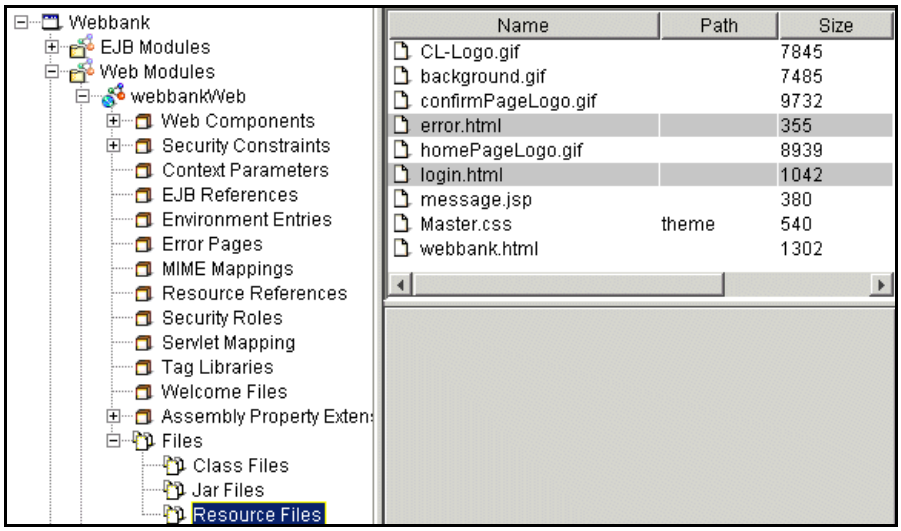


Figure 21-30 Resource files added for form authentication

When an unauthenticated user attempts to access a protected resource in the deployed Web module, the form authentication method will challenge for authentication details using the specified login form, as shown in Figure 21-31.

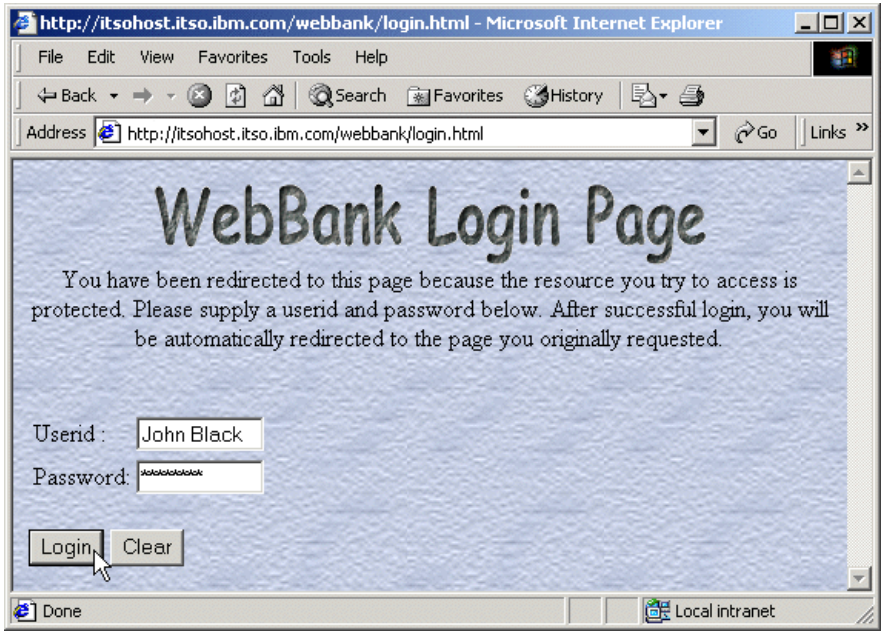


Figure 21-31 Form authentication challenge

Note: Make sure your login page is not protected by the resource constraint. If it is, the user will be redirected to the login page in an endless loop as WebSphere tries to authenticate the user accessing the login page!

Note: Form authentication uses the servlet sendRedirect() method. This method is used twice during form login:

1. To display the form login page in the Web browser
2. To redirect the browser back to the originally requested protected page

The sendRedirect(String URL) tells the Web browser to GET the page specified in the URL argument. The implication is that if an HTTP POST is the first request to a protected servlet or JSP, and no authentication or login has occurred previously, then the POST request will not be delivered to the protected page. An HTTP GET will be delivered instead.

This scenario can happen when you have an unprotected HTML form that collects information from a user who has not yet logged in. If this information is then posted (as in a POST HTML form action) to a protected servlet or JSP for processing, the sendRedirect() will deliver an HTTP GET instead.

To avoid this situation, structure your Web application or permissions such that login is required and performed prior to any POST actions to any protected servlets or JSPs.

21.3.5 Configuring client certificate authentication

To configure client certificate authentication for a Web module:

1. First, configure your Web server to use client certificate authentication. See 21.8, “Using client certificate-based authentication” on page 792 for details on IBM HTTP Server and IBM SecureWay Directory Server.
2. In AAT, right-click your Web module and select **Properties** from the pop-up menu, as shown in Figure 21-26 on page 764. We are using the webbankWeb Web module.
3. In the Web Module Properties window click the **Advanced** tab to set login configuration properties:
   a. Check the **Login configuration** box.
   b. Set Authentication method to Client cert.
   c. Specify a realm name.

   Our login configuration settings are shown in Figure 21-32.

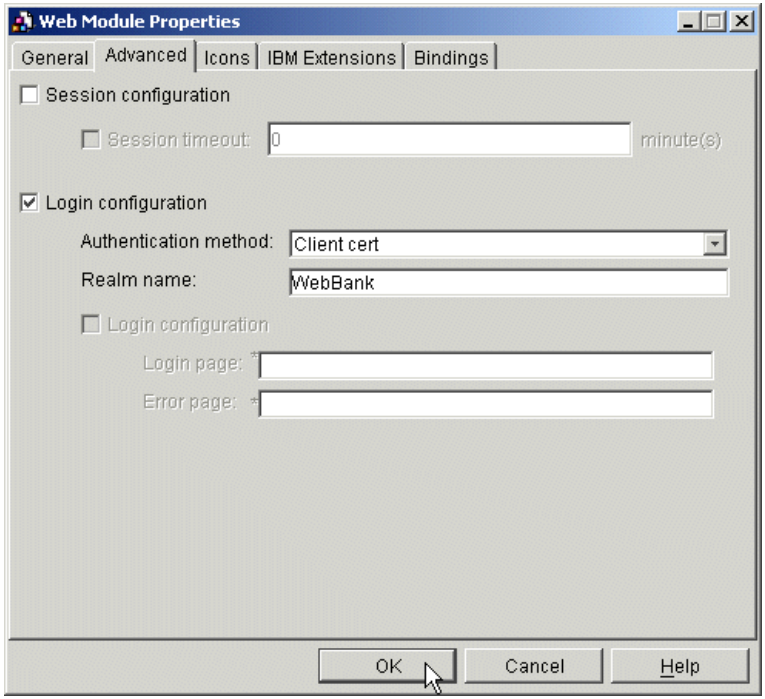


Figure 21-32 Web Module Properties for client certificate authentication

4. Click **OK** to update the login configuration settings.

Example 21-5 shows the login-config element from the web.xml deployment descriptor, when client certificate authentication is specified.

Example 21-5 login-config element for client certificate authentication in web.xml

```
<login-config id="LoginConfig_1">
   <auth-method>CLIENT-CERT</auth-method>
   <realm-name>WebBank</realm-name>
</login-config>
```

When an unauthenticated user attempts to access a protected resource in the deployed Web module, the Web server will challenge for client certificate authentication as shown in Figure 21-33.

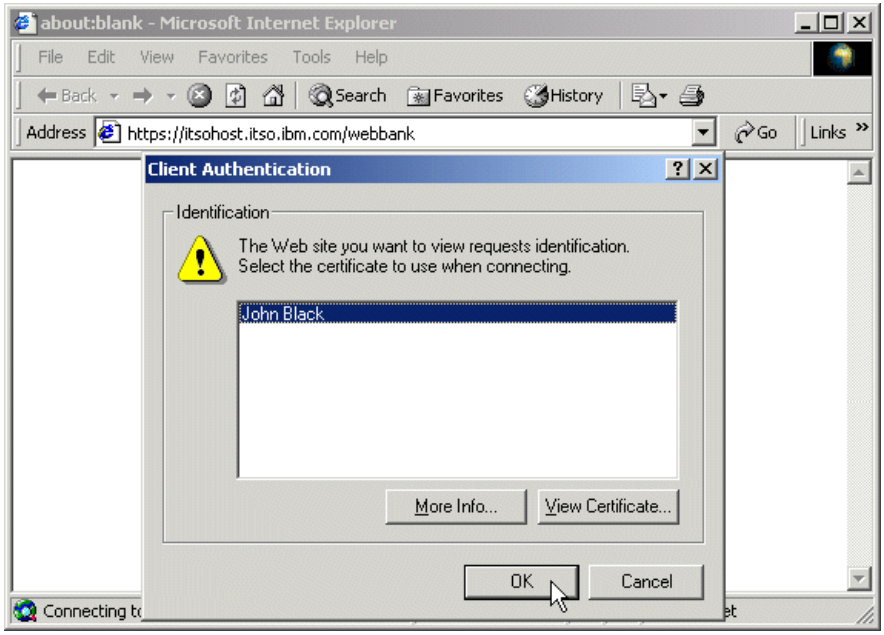


Figure 21-33 Client certificate authentication challenge

21.4 Configuring EJB method permissions

An EJB method permission is a mapping between one or more security roles and one or more methods that a member of the role can invoke. The security roles must be defined, and the methods must be in the enterprise bean's remote or home interfaces.

To add an EJB method permission:

1. Open your enterprise application archive in AAT. We are using webbank.ear in our example. You can find webbank.ear from Appendix D, “Additional material” on page 1083.
2. Double-click to expand **EJB Modules**.
3. Double-click to expand the EJB module you want to work with. We are using the webbankEJBs module.
4. Right-click **Method Permissions** and select **New** from the pop-up menu, as shown in Figure 21-34.

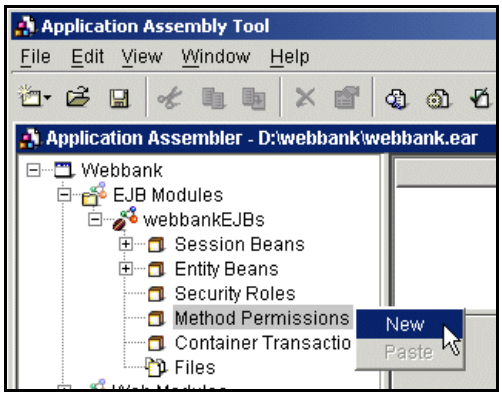


Figure 21-34 Adding a new EJB method permission

5. In the New Method Permission window:
   a. Set the method permission name.
   b. Optionally add a description of the method permission.
   c. Click Methods: **Add...** and select the EJB methods that the method permission allows, as shown in Figure 21-35.

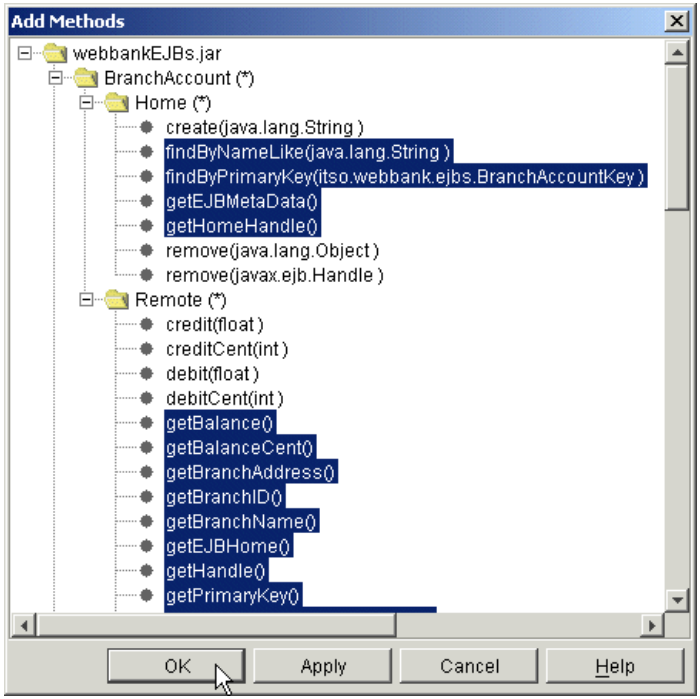


Figure 21-35 Adding EJB methods to the method permission

d. For the full list of methods we selected, see Example 21-6.

Example 21-6 Methods added to "Allow read for AllAuthenticated" method permission

```
ejb-name/method-intf/method-name
BranchAccount/Home/findByNameLike
BranchAccount/Home/findByPrimaryKey
BranchAccount/Home/getEJBMetaData
BranchAccount/Home/getHomeHandle
BranchAccount/Remote/getBalance
BranchAccount/Remote/getBalanceCent
BranchAccount/Remote/getBranchAddress
BranchAccount/Remote/getBranchID
BranchAccount/Remote/getBranchName
BranchAccount/Remote/getEJBHome
BranchAccount/Remote/getHandle
BranchAccount/Remote/getPrimaryKey
BranchAccount/Remote/isIdentical
Consultation/All methods/*
CustomerAccount/Home/findByPrimaryKey
CustomerAccount/Home/getEJBMetaData
CustomerAccount/Home/getHomeHandle
CustomerAccount/Remote/getAccountNumber
```

```
CustomerAccount/Remote/getBalanceCent
CustomerAccount/Remote/getCustomerID
CustomerAccount/Remote/getCustomerName
CustomerAccount/Remote/getEJBHome
CustomerAccount/Remote/getHandle
CustomerAccount/Remote/getPrimaryKey
CustomerAccount/Remote/isIdentical
Transfer/Home/*
Transfer/Remote/getEJBHome
Transfer/Remote/getHandle
Transfer/Remote/getPrimaryKey
Transfer/Remote/isIdentical
Transfer/Remote/remove
```

You can use asterisk (*) to denote all the methods of an enterprise bean's remote and home interfaces.

e. Click Roles: **Add...** and select the security role(s) that must be granted in order to invoke the method, as shown in Figure 21-36.

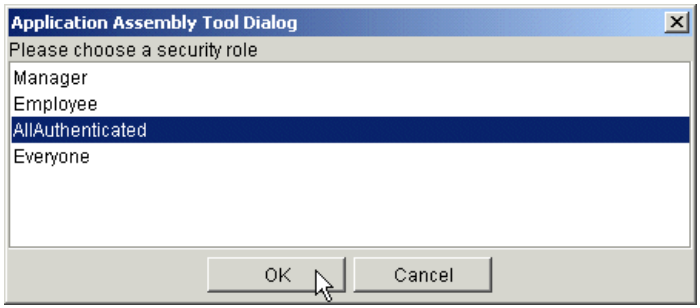


Figure 21-36 Selecting security roles needed to invoke method permissions

Our new method permission properties are shown in Figure 21-37.

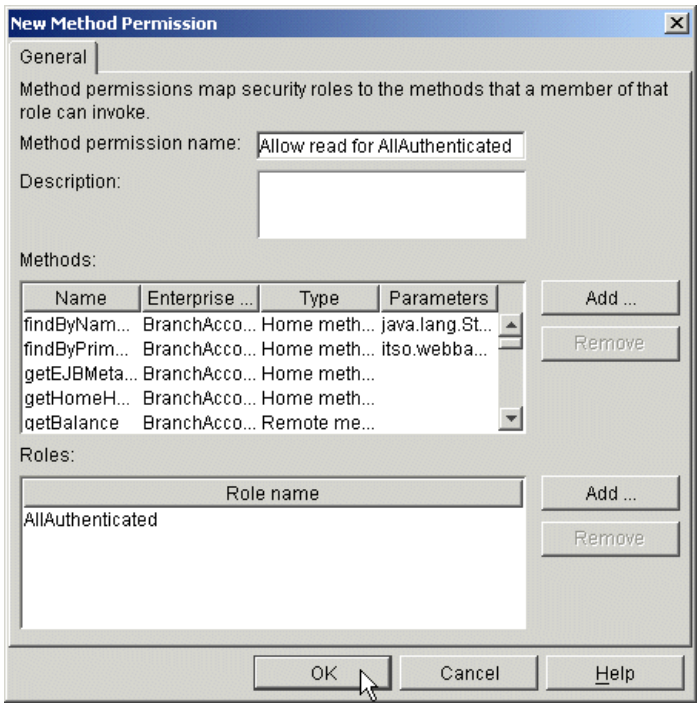


Figure 21-37 New Method Permission properties - selected methods allowed

6. Click **OK** to create the new EJB method permission.

For our example application, we created a second EJB method permission. We added all EJB methods, as shown in Figure 21-38, and called it “Allow all for Employees”.

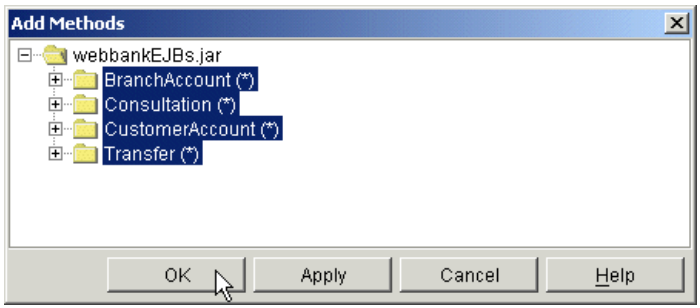


Figure 21-38 Adding all methods from EJBs to the method permission

We specified that the Employee and the Manager roles are needed to invoke the method. The method permission properties are shown in Figure 21-39.

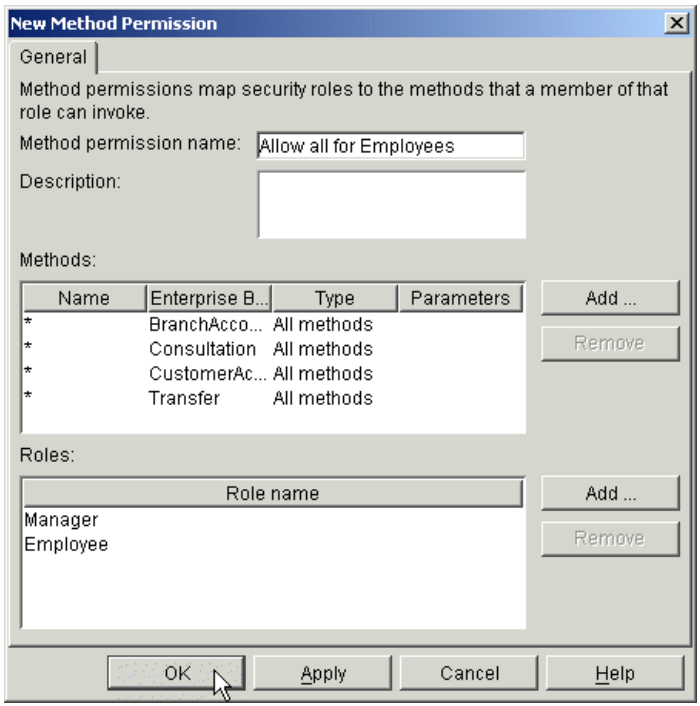


Figure 21-39 New Method Permission properties - all methods allowed

21.5 Assigning users/groups to security roles

Now that we have restricted Web module security constraints and EJB method permissions to users in specific security roles, we need to grant these security roles to individual users or groups of users. You have a number of options on *when* you assign users or groups to security roles. You can make this assignment:

- When packaging your application, with the Application Assembly Tool
- When deploying your application, with the administrative console
- When maintaining your application, with the security center

When you use the administrative console or the security center, you can browse and select users directly from the user repository. When using AAT to package an application, AAT does not provide this access to the user repository. You need to manually enter the required users or groups.

To improve application maintainability, you may consider first setting up user groups in your user repository. You can then assign user groups to roles when packaging or deploying your application. In this way, ongoing maintenance of individual users and group memberships is done in the user repository, thereby minimizing application server maintenance.

In our example, we assign users and groups to security roles when deploying the application with AAT, as follows:

1. First, set up your application users and groups in your user repository. We set up our application users and groups as described in 21.7.2, “Adding users to IBM SecureWay Directory” on page 788 and 21.7.3, “Adding groups to IBM SecureWay Directory” on page 790.
2. Install your enterprise application using the WebSphere Administrative Console. Right-click Enterprise Applications and select **Install Enterprise Application** from the pop-up menu.

   In the Install Enterprise Application Wizard:

   a. Specify your enterprise application archive path and click **Next**. We are using D:\webbank\webbank.ear.

   b. Click **Yes** to deny all access to unprotected methods, when prompted as shown in Figure 21-40.

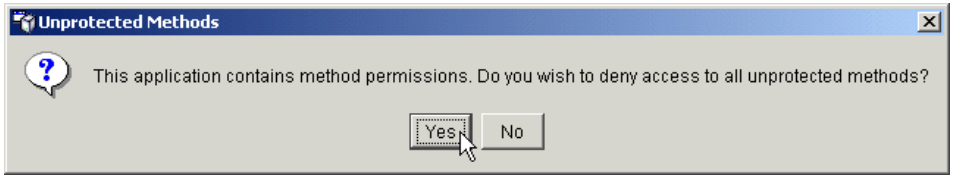


Figure 21-40 Denying access to unprotected methods

   c. In the Mapping Users to Roles window:

      - Select the **Employee** role and click **Select...**, as shown in Figure 21-40.

        In the Select Users/Groups window, check **Select users/groups**, set Search pattern to * and click **Search**. In Available Users/Groups, choose **employee** then click **Add** to add the employee to Selected Users/Groups, as shown in Figure 21-41.

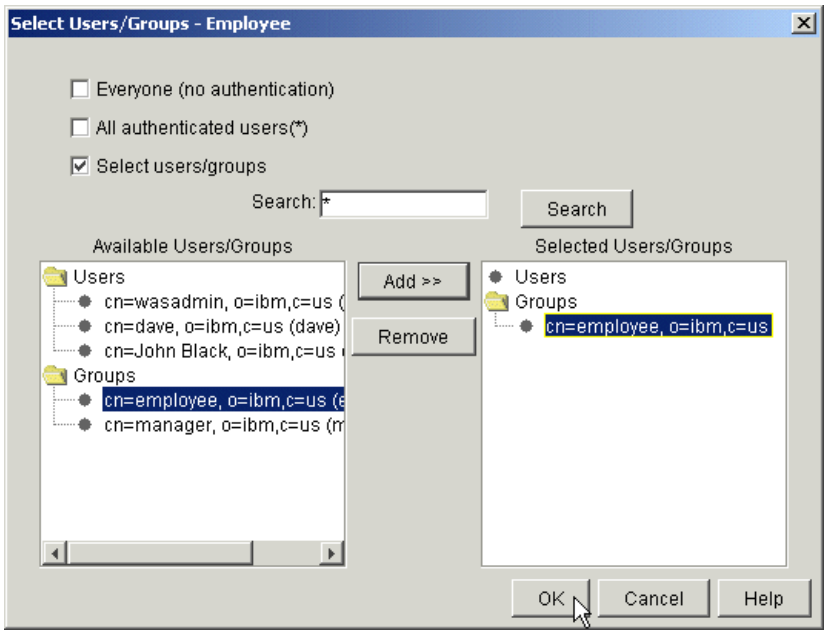


Figure 21-41 Select User/Groups for employee role

Click **OK** to return to the Mapping Users to Roles window.

- Click the **Manager** role and select the **manager** group, as we did with the employee group.
- Our role mappings are shown in Figure 21-42 on page 780.
- Click **Next**.

d. It should just be a matter of clicking **Next** on each of the remaining windows to complete the wizard. See for Chapter 19, “Deploying an application” on page 687 details.

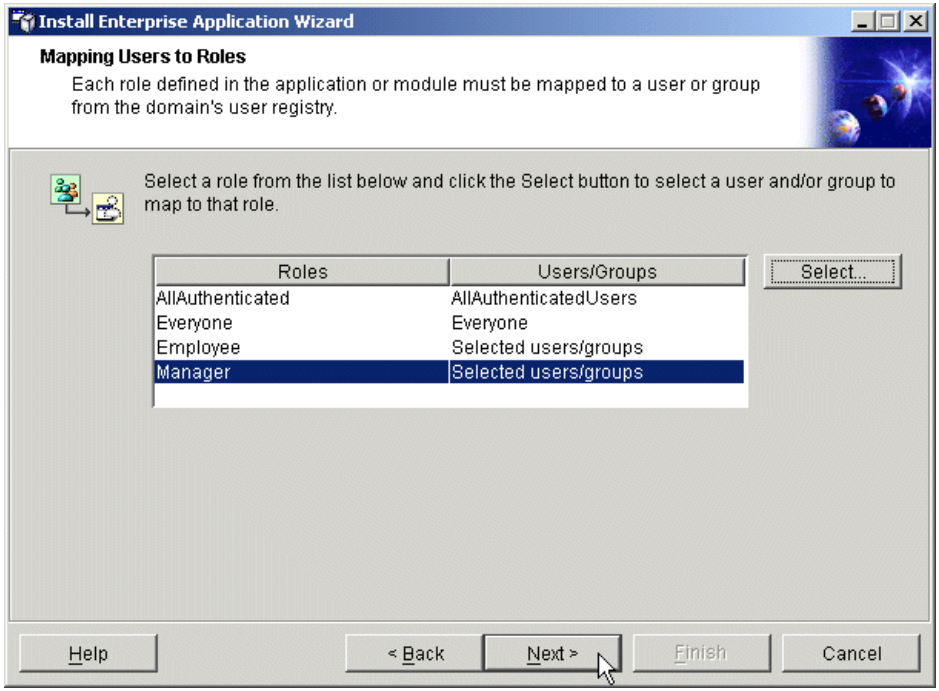


Figure 21-42 Mapping Users to Roles

3. Once you have deployed the WebBank sample application, you need to do the following:

   a. Start the enterprise application. Right-click the enterprise application and select **Start** from the pop-up menu.

   b. Regenerate the Web server plug-in configuration. Right-click the node and select **Regen Webserver Plugin** from the pop-up menu.

   c. Make sure module visibility is “Application”. Do this under Application Server Properties. Restart the application server if you have to change module visibility.

   See Chapter 19, “Deploying an application” on page 687 details.

To finish, check that your security role assignments work as expected:

1. Open URL `http://<your.server>/webbank` from your Web browser.
2. Log in as a user granted the Employee role, such as John Black. Try a customer to branch transfer, as shown in Figure 21-43.

Figure 21-43 WebBank customer to branch transfer

You should get a message indicating that the transfer was completed successfully, as shown in Figure 21-44.

Figure 21-44 WebBank successful transfer

3. Restart your browser and log in as a user not granted the Employee role, such as wasadmin (the WebSphere administration account). You should get a message indicating that the transfer failed.

   If you check the event viewer in the administrative console, you should see an EJS container exception indicating that the user is not granted any of the required roles:

   ```
   CNTR0019E: Non-application exception occurred while processing method
   create: com.ibm.websphere.csi.CSIException: SECJ0053E: Authorization failed
   for itsohost.itso.ibm.com:389/wasadmin while invoking
   (Home)webbank/Transfer create:0 securityName:
   itsohost.itso.ibm.com:389/wasadmin;accessID:
   user:itsohost.itso.ibm.com:389/cn=wasadmin, o=ibm,c=us is not granted any
   of the required roles: Manager Employee DenyAllRole
   ```

21.6 Accessing protected EJBs from a J2EE application client

When your application client uses the default sas.client.props security configuration, which has com.ibm.CORBA.securityEnabled set to true, it will assume that application security is enabled. If application security is not enabled, the server ignores the security request, and the client works as expected.

You can start our WebBank J2EE sample client as follows:

1. Start the launchClient utility <WAS_HOME>\bin\launchClient <userapp.ear>.

   ```
   D:\WebSphere\AppServer\bin\launchClient D:\webbank\webbank.ear
   ```

2. Enter your user ID and password when prompted, as shown in Figure 21-45.

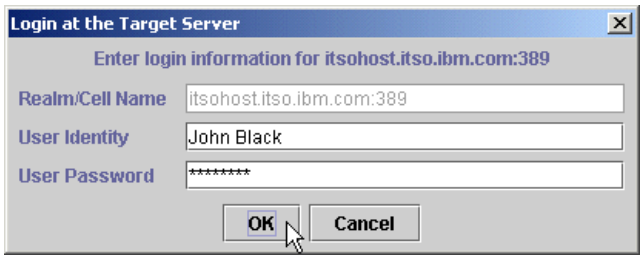


Figure 21-45 J2EE client login to target server

3. Once you log in successfully, try to get the branch or customer balance. Click **Init**, then click **Get Branch Balance** or **Get Customer Balance**, as shown in Figure 21-46.

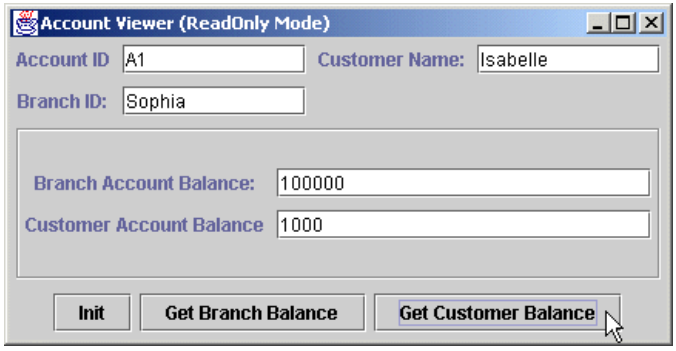


Figure 21-46 WebBank Account Viewer J2EE client

4. Close the application client and log in as a user not granted the Employee role, such as wasadmin (the WebSphere administration account). You should still be able to get the branch or customer balance, as we granted the “Allow

read for AllAuthenticated" method permission to all authenticated uses in 21.4, "Configuring EJB method permissions" on page 772.

21.7 Setting up an LDAP directory

To allow WebSphere to authenticate users via an LDAP directory, you first need to configure your LDAP directory. In this section we describe how you can set up the IBM SecureWay Directory for use with WebSphere security.

21.7.1 Setting up IBM SecureWay Directory

This section shows how to prepare a new IBM SecureWay Directory Server installation so that you can begin adding users.

1. First install the IBM SecureWay Directory Server, as described in its installation documentation. We used the following product versions for our installation:
   - IBM SecureWay Directory Server Version 3.2.1 for Windows
   - IBM DB2 Universal Database V7.2.1, Enterprise Edition for Windows
   - IBM HTTP Server 1.3.19 for Windows

   We used the same local database server and Web server for SecureWay and WebSphere.
2. Once you have installed SecureWay, use Windows Services to make sure the following services are started:
   - DB2 - LDAPDB2
   - IBM SecureWay Directory V3.2
   - IBM HTTP Server
3. The SecureWay Directory server has a Web-based server administration interface. Start your Web browser and open `http://<your.server>/ldap`.
4. When prompted for the admin ID and password, as shown in Figure 21-47, enter the distinguished name and password for the administrator that you chose during the installation, for example `cn=root`.

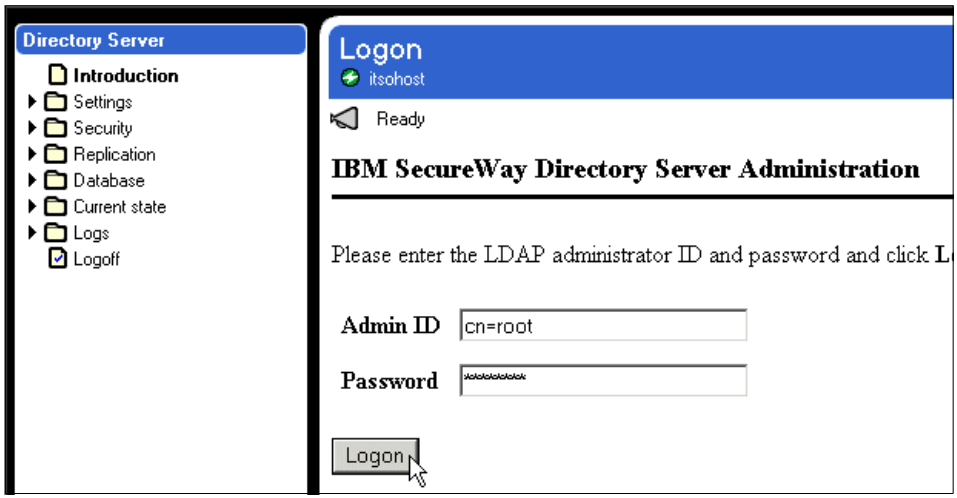


Figure 21-47 Log on to IBM SecureWay Directory Server Administration

Tip: If you do not know the admin ID and password set at installation time, you can reset them by starting the Directory Configuration tool, and selecting *only* the **Set the directory administrator name and password** option.

Before using the directory, you need to add a suffix. A suffix is a root from where searches are performed, or an entry point into the directory. You can then add entries to the directory tree, starting from that suffix.

5. Once logged on, you can add a suffix by clicking the **add suffixes** link at the top of the page, or by expanding **Settings** -> **Suffixes** on the left.

   Enter the **Suffix DN**. We set the Suffix DN to o=ibm,c=us, where o=organisation, c=country as shown in Figure 21-48. Click **Update**, then click the **restart the server** prompt for the change to take effect.

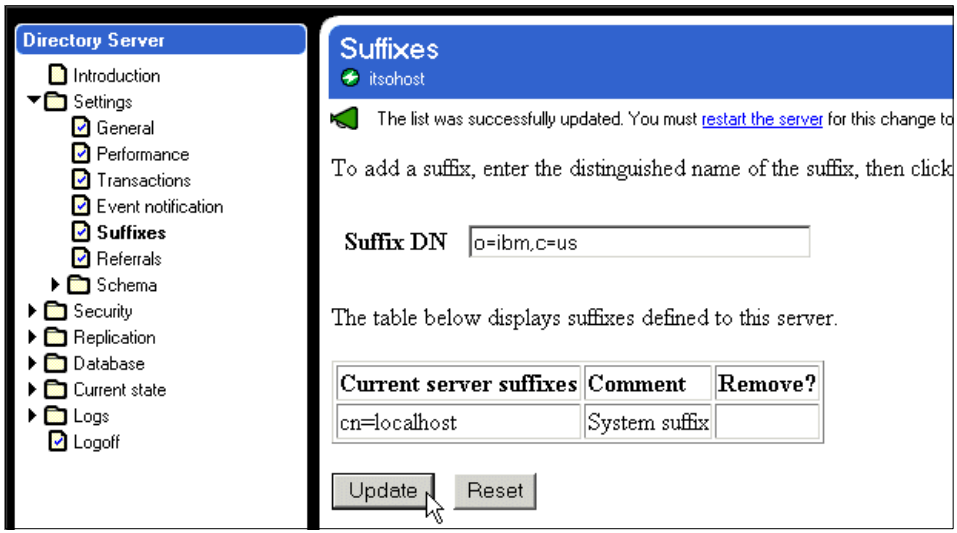


Figure 21-48 Adding a suffix distinguished name

You can now add entries to the directory tree using the Directory Management Tool.

6. In Windows start the Directory Management Tool by clicking **Start** -> **Programs** -> **IBM SecureWay Directory** -> **Directory Management Tool**.

   You will receive a warning that the entry in the tree you just created (`o=ibm,c=us` in our case) does not contain any data. Each entry in the directory tree must have an object attached to it. By adding a suffix, you have created an entry, but you have not attached any data to it yet. This is the next step.

7. First, you need to bind to the directory tree as an authenticated user. Select the **Server** -> **Rebind** option on the left. Choose the Authenticated option, and enter your administrator ID and password (don't forget to specify "`cn=`" in front of the user ID), as shown in Figure 21-49.

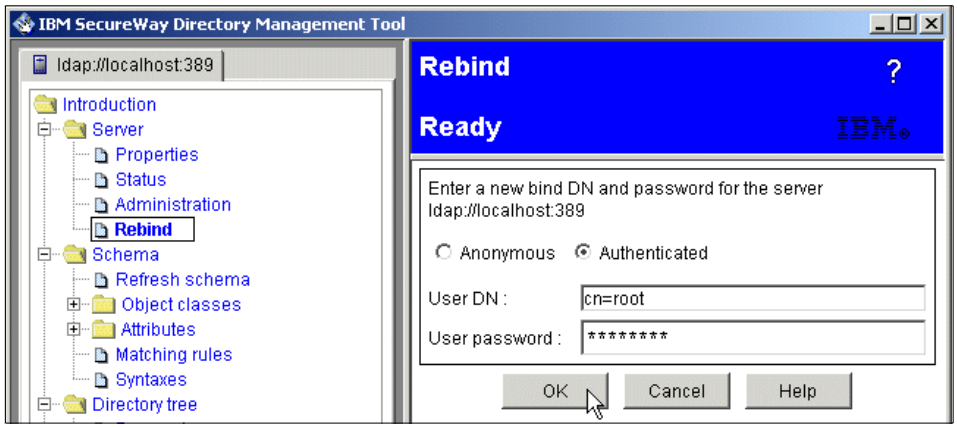


Figure 21-49 Rebinding as an authenticated user

Note: An anonymous bind allows you to browse the data, but does not allow you to modify it.

8. The directory tree browser is now open. You can see in Figure 21-50 the suffix does not yet appear in the tree. To attach an entry to it click the **Add** button in the tool bar.

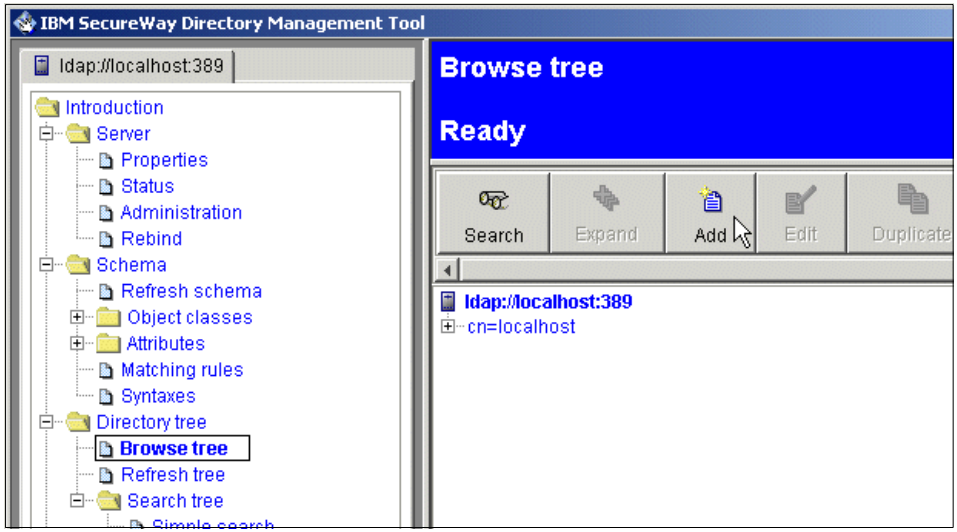


Figure 21-50 Adding an LDAP entry

9. In the Add an LDAP Entry window, shown in Figure 21-51, select Organization as the Entry type, type the suffix name (`o=ibm,c=us`) in the Entry RDN (relative distinguished name) field, and click **OK**.

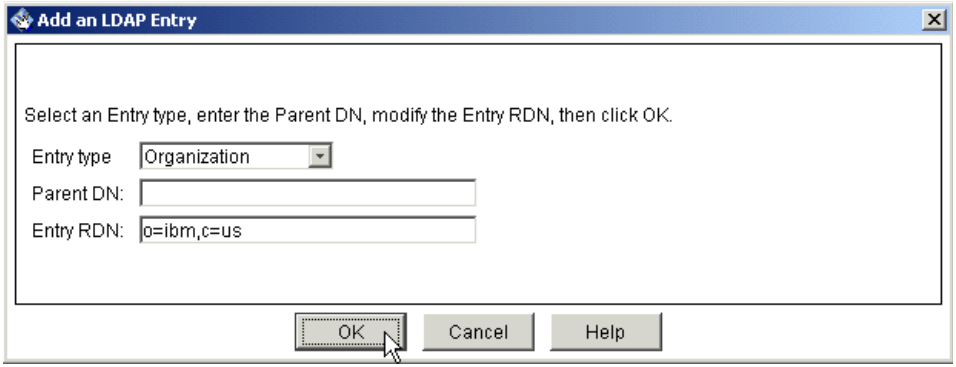


Figure 21-51 Adding an Organization RDN

10. In the next window you can set the entry attributes, as shown in Figure 21-52. All fields in bold face, such as **o:**, are mandatory. Click **Add** to create the new entry.

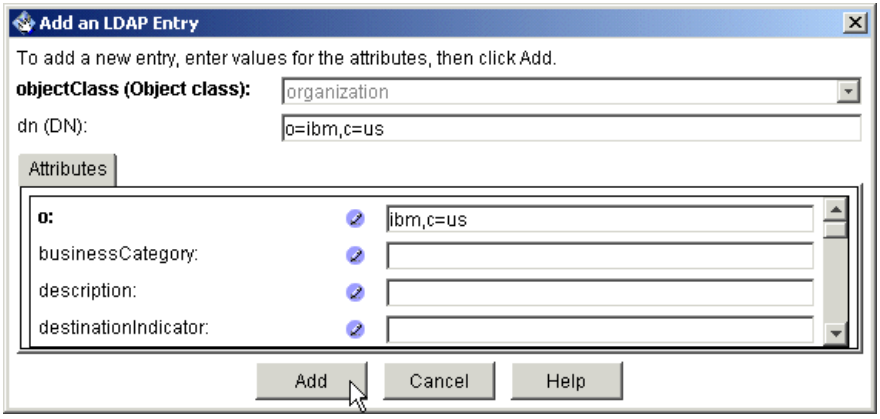


Figure 21-52 Setting organization attributes

11. The new entry should appear in the directory tree, as shown in Figure 21-53. Click **Directory Tree** -> **Refresh Tree** on the left if needed.

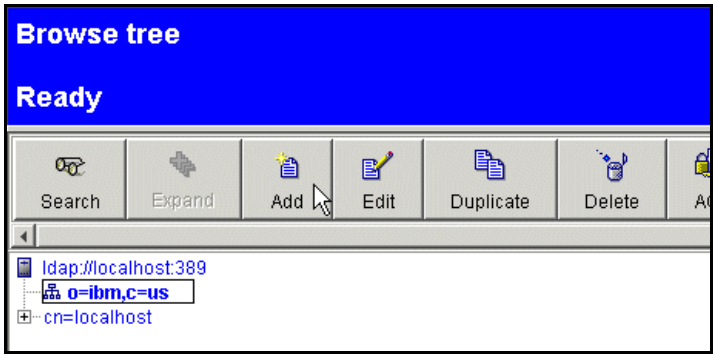


Figure 21-53 Directory tree with the new organization

21.7.2 Adding users to IBM SecureWay Directory

You are now ready to add users to the directory. Typically, you want to add users to the new organization you just created. We will add a user for the WebSphere administrator next.

1. Select the **o=ibm,c=us** entry in the directory tree, and click **Add** on the toolbar, as seen in Figure 21-53. This entry is your Parent DN.
2. Select **User** as the Entry type, and specify an Entry RDN such as `cn=wasadmin`, as shown in Figure 21-54. The full name of the new user is cn=wasadmin,o=ibm,c=us. This is called the full distinguished name (or full DN). Click **OK** to continue.

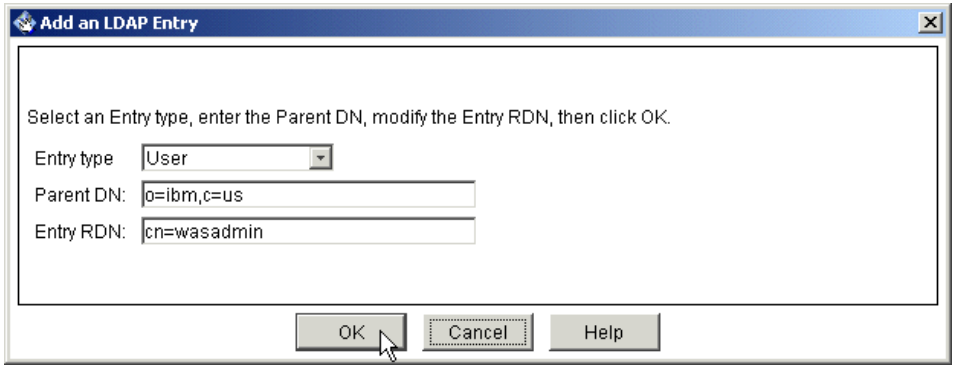


Figure 21-54 Adding a User RDN

3. In the next window you can set the entry attributes, as shown in Figure 21-55. Fill in the mandatory fields, which are in boldface, then provide a userPassword under the Business tab.

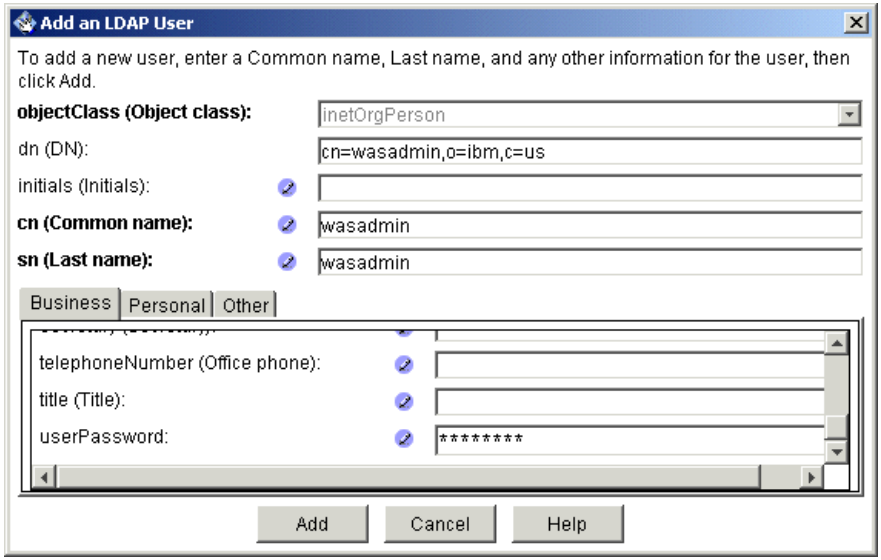


Figure 21-55 Setting User Business attributes

4. Click the **Other** tab and supply a uid, as shown in Figure 21-56. Click **Add** to create the new entry.

> **Note:** The uid is a (short) alternative to the full DN. If you do not supply it, users would have to authenticate themselves using the full DN.

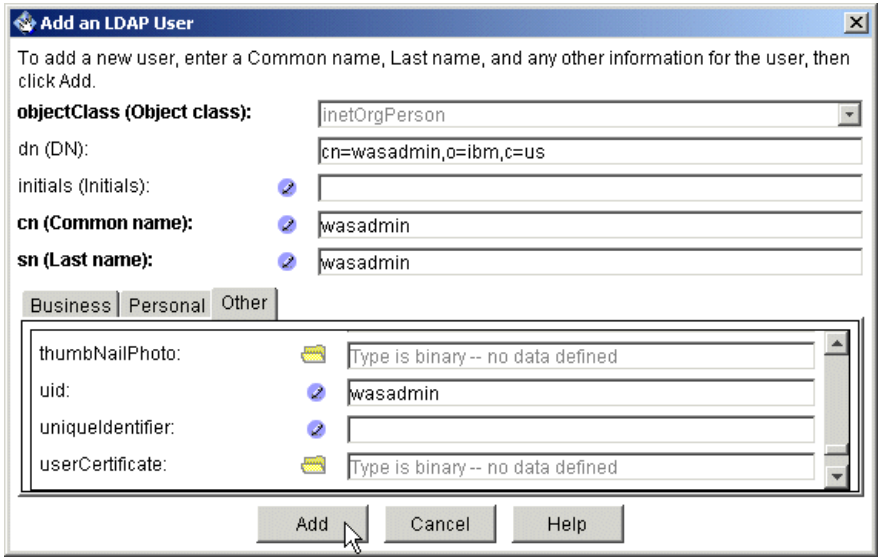


Figure 21-56 Setting User Other attributes

5. The new entry should appear in the directory tree, as shown in Figure 21-57. Click **Directory Tree** -> **Refresh Tree** on the left if needed.

Figure 21-57 Directory tree with the new user

6. You can repeat the steps in this section to create more users.

   We also created an application user:

Full DN:	cn=John Black,o=ibm,c=us
Common name:	John Black
Last name:	Black
userPassword:	********
uid:	John Black

21.7.3 Adding groups to IBM SecureWay Directory

Next we add groups to the directory. We will add a group for users who are employees.

1. Select the **o=ibm,c=us** entry in the directory tree, and click **Add** on the toolbar, as seen in Figure 21-53 on page 788.
2. Select **Group** as the Entry type, and specify an Entry RDN such as `cn=employee`, as shown in Figure 21-58. Click **OK** to continue.

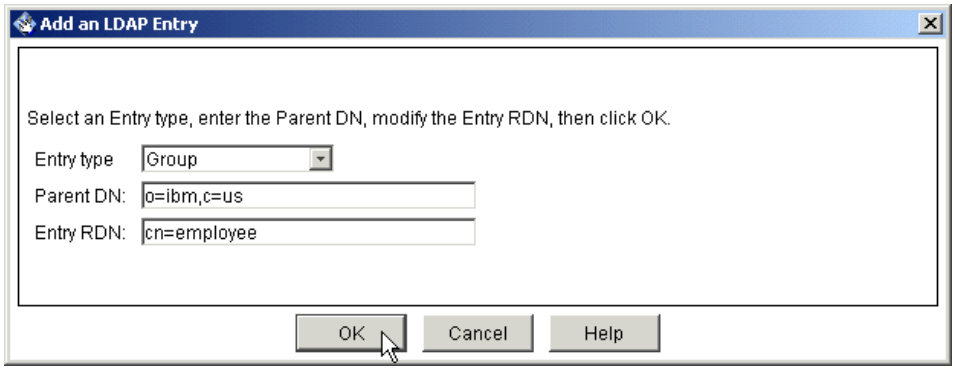


Figure 21-58 Adding a Group RDN

3. In the next window you can set the entry attributes. We added John Black as Group member, as shown in Figure 21-59. Click **Add** to create the new entry.

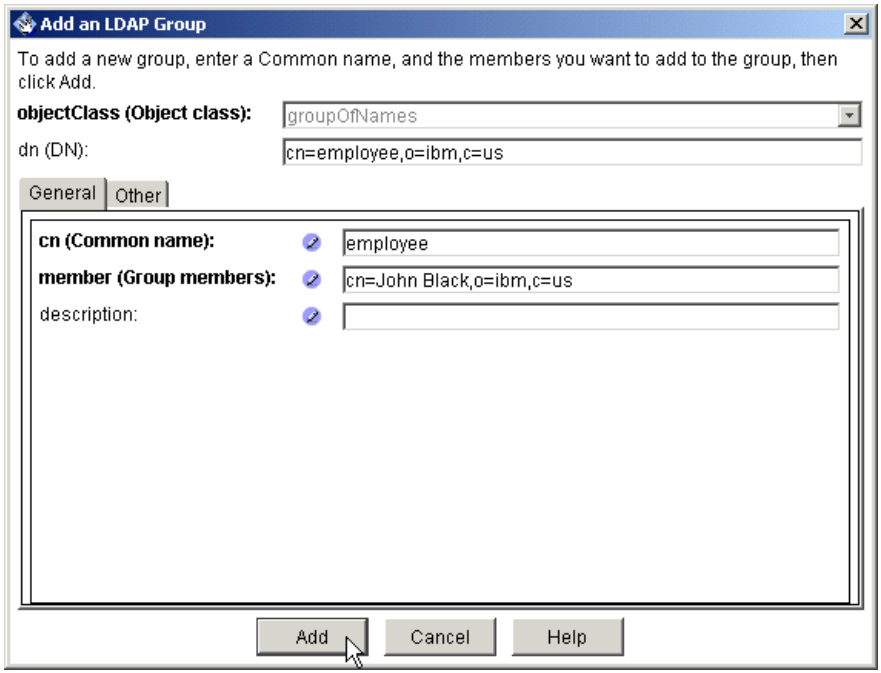


Figure 21-59 Setting Group attributes

4. The new entry should appear in the directory tree, as shown in Figure 21-60. Click **Directory Tree** -> **Refresh Tree** on the left if needed.

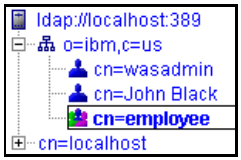


Figure 21-60 Directory tree with the new group

5. You can repeat the steps in this section to create more groups.

   We also created a manager group:

   Full DN: cn=manager,o=ibm,c=us

   Common name: manager

   Group members: John Black

21.8 Using client certificate-based authentication

In this section we discuss certificates and technologies related to certificates, and provide an example of using certificates with WebSphere.

Certificate-based authentication requires the specification of LTPA as the WebSphere authentication mechanism. LTPA provides the means for the WebSphere Application Server to access an LDAP directory that contains the authentication information. For our certificate-based authentication example, we will be using the IBM SecureWay Directory, which is an LDAP conforming registry.

In our example, the following products are used:

- IBM SecureWay Directory Server Version 3.2.1 for Windows
- IBM DB2 Universal Database V7.2.1, Enterprise Edition for Windows
- IBM HTTP Server 1.3.19 for Windows
- Microsoft Internet Explorer V5.5

Our example assumes that WebSphere, SecureWay, DB2, IBM HTTP Server, and Internet Explorer have all been previously installed on a single Windows 2000 server. It also assumes that the default server and Web application are available in WebSphere (that is, the **Samples** component was selected during WebSphere installation).

Note: This chapter is concerned with communication between the Web client and Web server. For an overview on encrypting communication between the Web server plug-in and WebSphere, see 8.4.7, “Encrypt communication between plug-in and WebSphere” on page 155. For full details refer to one of the platform-specific installation chapters.

21.8.1 Web client security flow with certificates

We start our discussion with a scenario that describes the flow of certificate-based authentication in a WebSphere environment. The basic flow is shown graphically in Figure 21-61 on page 794 and is as follows:

1. The request for the IncServlet servlet comes from the browser user to the Web server.
2. The Web server determines that the request is for a resource that requires client certificate authentication, so it challenges the browser to return a certificate.
3. The browser recognizes the challenge and returns the certificate.
4. The Web server authenticates the client certificate. It then determines that it does not control the servlet request, so it passes the client certificate along with the request to the application server.
5. The application server determines that it controls the servlet and, through the security collaborator, determines that the servlet is secure.
6. The application server, through the security collaborator, determines that the required authentication method is client certificate.
7. Using the security collaborator, which in turn works with the security application, the application server determines the credentials of the certificate. The LTPA server component of the security application works with the LDAP server to perform a credential mapping of the certificate to the contents of the LDAP directory.
8. Using the security collaborator, the application server determines that the user is authorized to access the servlet being requested.
9. The application server invokes the servlet for the user.

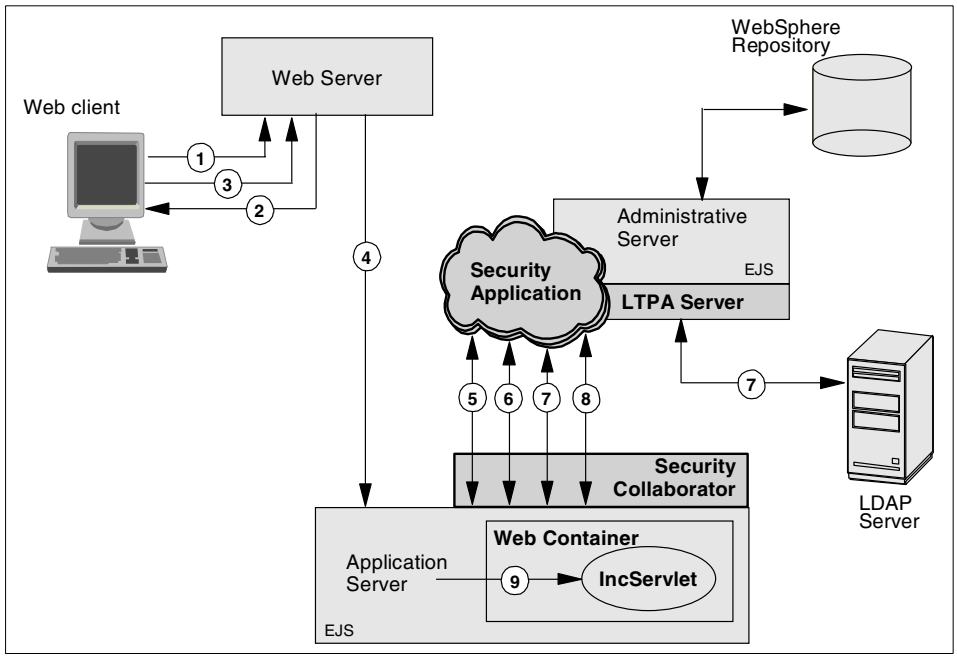


Figure 21-61 WebSphere security flow using client certificates

21.8.2 Managing certificates

In our example, we will use certificates for two purposes: first, to allow an SSL connection to be established between the browser client and the Web server, and second, to support WebSphere authentication of the browser client. We will use a signer certificate from a trusted Certificate Authority (CA), a personal certificate that identifies the server, and a personal certificate that identifies the client.

A key database file located on the WebSphere server machine is created and holds the certificates needed by the server. The server needs:

- The signer certificate of the trusted CA
- A server certificate signed by the trusted CA
- A client certificate signed by the trusted CA

The browser security specifications will also need to be updated to contain the signer certificate of the trusted CA and the client certificate.

WebSphere provides two means to manage certificates:

- A graphical tool called IKeyMan, the IBM Key Management tool

- A package of Java command-line tools, com.ibm.cfwk.tools (CFWK tools)

The CFWK tools support scripting of certificate management, which is useful for administrators who do a lot of this work or who want to automate the work. The IKeyMan tool is much easier to use for small tasks, so that is what we will be using for our example.

Create a key database for the server

We will create the key database file using the IBM Key Management utility. To create the key database file, do the following:

1. Create a working directory in the Windows 2000 machine to be used during certificate creation and management (we'll use <WAS_HOME>\etc in our example).
2. Start the IBM Key Management Utility by selecting Windows **Start** -> **Programs** -> **IBM HTTP Server** -> **Start Key Management Utility**. The window shown in Figure 21-62 will appear.

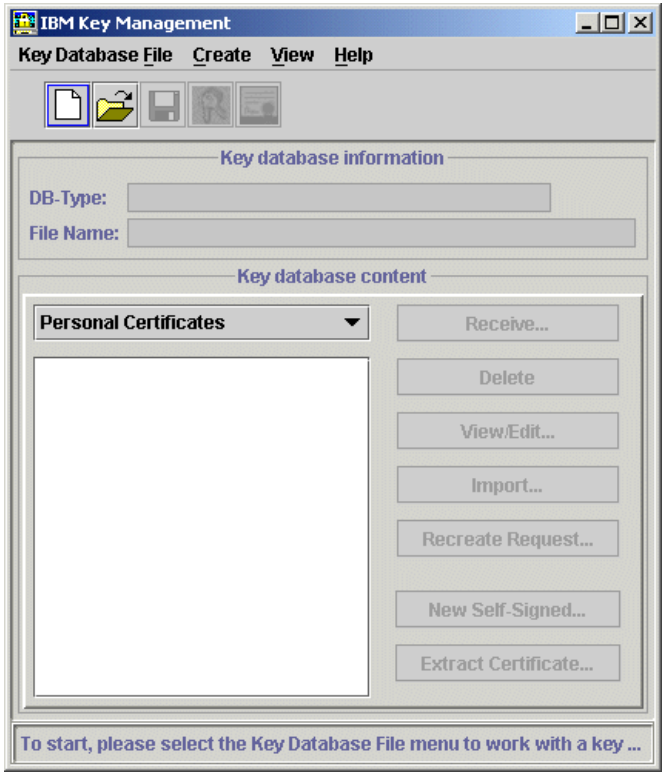


Figure 21-62 IKeyMan initial window

3. From the menu bar, select **Key Database File** -> **New...**. In the New window (Figure 21-63):
   - Keep the Key database type as CMS key database file
   - Type a file name for the key database (in our case, Serverkey.kdb)
   - Make the location the working directory (in our case, <WAS_HOME>\etc\)
   - Click **OK**

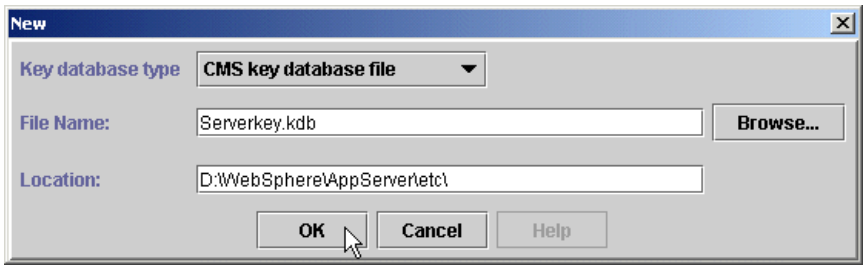


Figure 21-63 Creating a new key database

4. In the Password Prompt window (Figure 21-64):
   - Type a password and confirm the password used to access the key database. In our case, the password used was WebAS.
   - Check the **Stash the password to a file?** box.
   - Click **OK**.

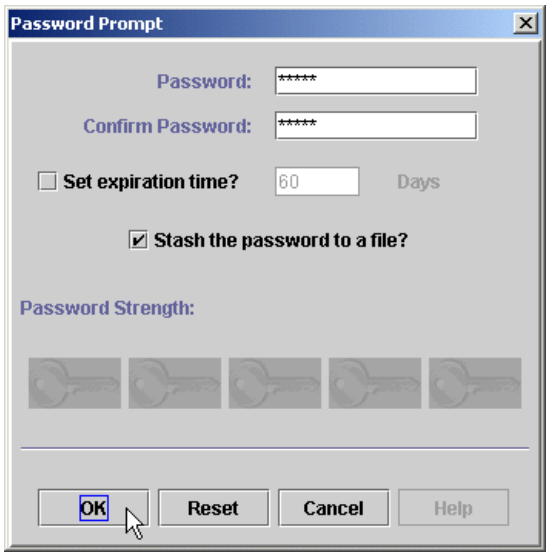


Figure 21-64 Password Prompt window

5. In the Information window (Figure 21-65), click **OK**.

Figure 21-65 Information window

The key database has now been created. See Figure 21-66.

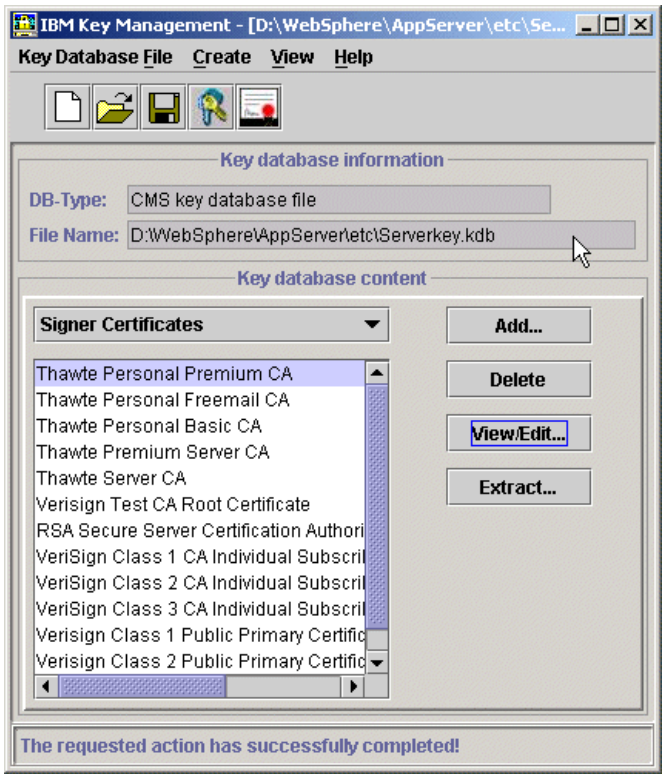


Figure 21-66 The key database has been created

Obtain a signer certificate of a trusted CA

When IKeyMan creates a key database, it automatically includes signer certificates from some well-known CAs. You can view the list of CAs by selecting signer certificates from the drop-down list in the Key database content area of the main IKeyMan window.

Since we were unable to find a free and easy way to obtain test client and server certificates from these CAs, we choose to use another CA. So, the first thing to do is obtain a new signer certificate.

Certificates will be obtained using a set of Web-based services provided by Microsoft. To obtain a new CA signer certificate do the following:

1. From a browser, go to URL `http://marting.develop.com/certsrv/`. In our case, we are using Microsoft Internet Explorer.
2. From the Welcome page (Figure 21-67), select **Retrieve the CA certificate or certificate revocation list** and click **Next**.

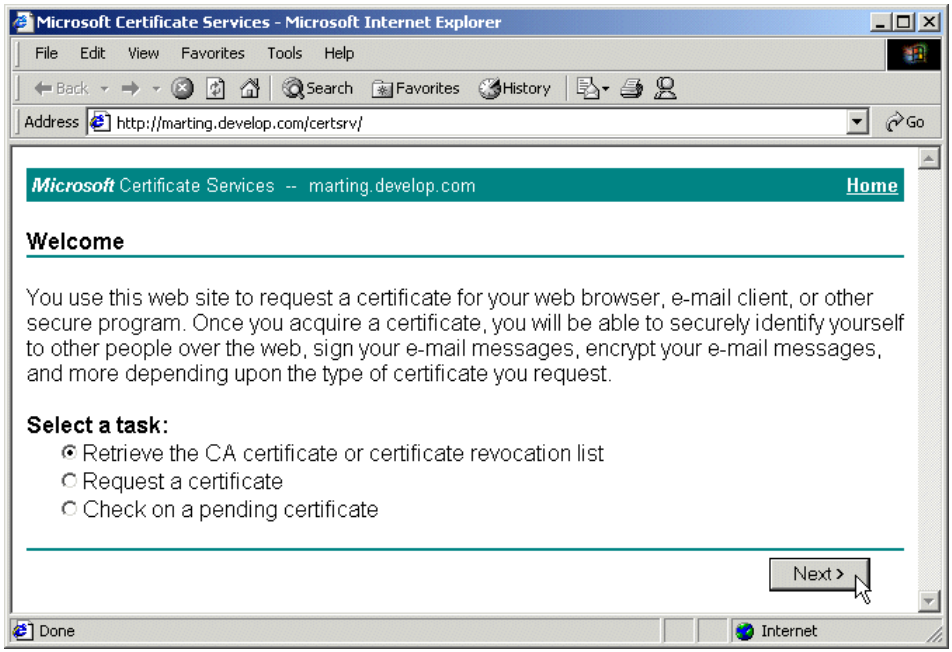


Figure 21-67 Retrieve the CA certificate

3. Select **Base 64 encoded** and click **Download CA certificate**.The window shown in Figure 21-68 will appear.

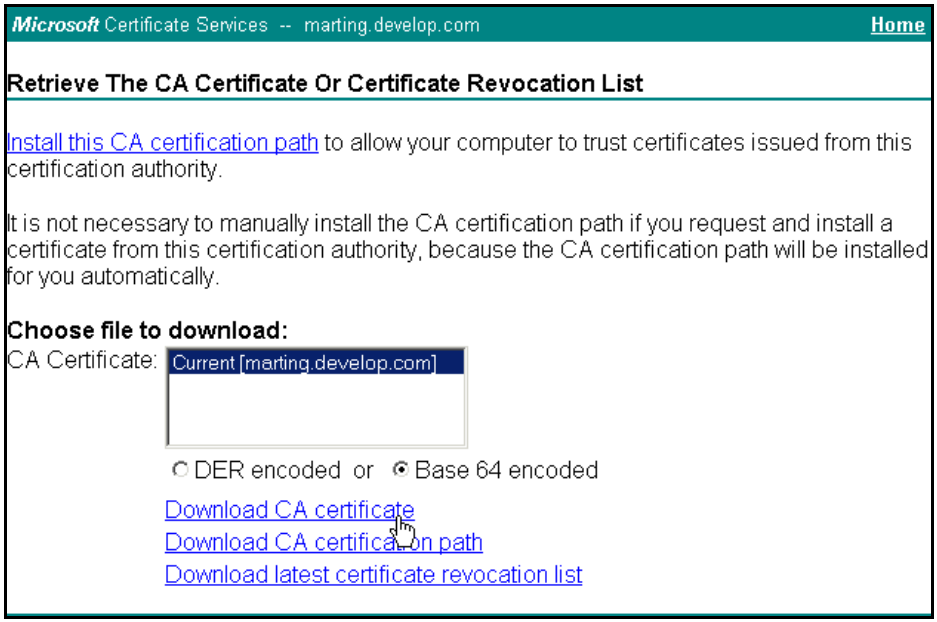


Figure 21-68 Download CA certificate

4. In the File Download window, make sure that **Save this file to disk** is selected, then click **OK**.
5. Specify a directory to save in and a file name. Then click **Save**. In our case the directory is <WAS_HOME>\etc and file name is CAcertnew.cer.

Now, add the CA certificate to the key database. Using IKeyMan, do the following:

1. From the main IKeyMan window (Figure 21-69), select **Signer Certificates** from the drop-down list in the Key database content area.
2. Click the **Add...** button.

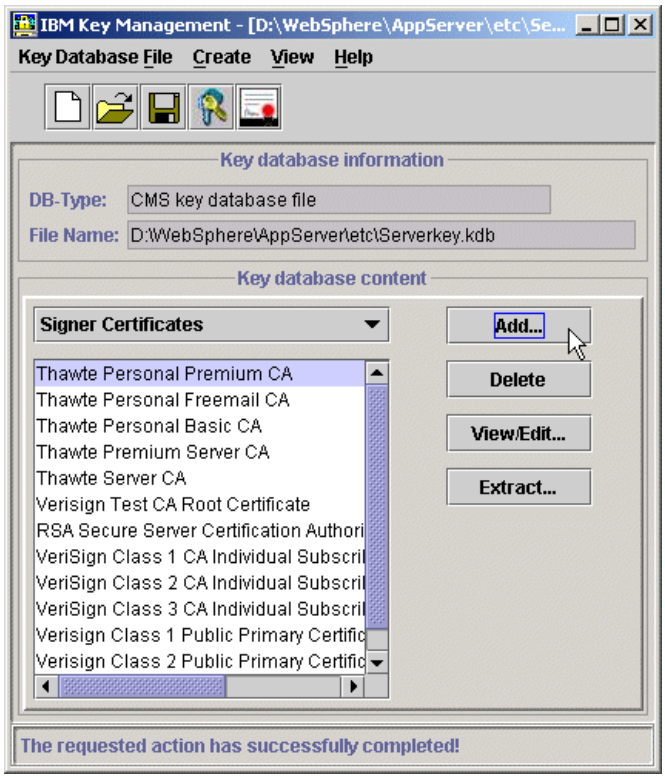


Figure 21-69 Add a signer certificate

3. In the Add CA's Certificate from a File window (Figure 21-70):
   - Leave the Data type as Base64-encoded ASCII data.
   - Type the Certificate file name. In our case, the name is CAcertnew.cer.
   - Type the Location. In our case, it is <WAS_HOME>\etc\.
   - Click **OK**.

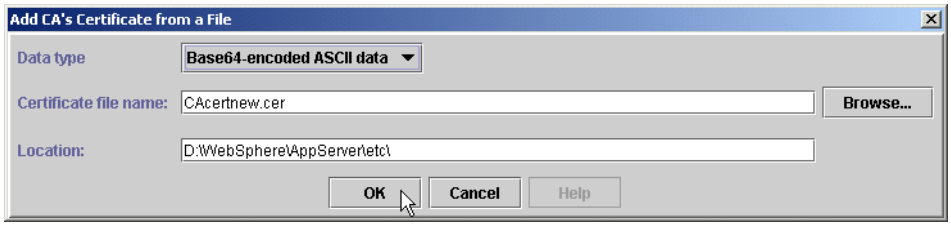


Figure 21-70 Add CA's Certificate from a File

4. In the Enter a Label window (Figure 21-71), type the name to be used to identify the certificate as it is listed in IKeyMan and click **OK**. In our case, the label is MS Test CA.

Figure 21-71 Enter a label for the certificate

The IKeyMan main window with signer certificates selected should now include the MS Test CA in the certificate list as shown in Figure 21-72.

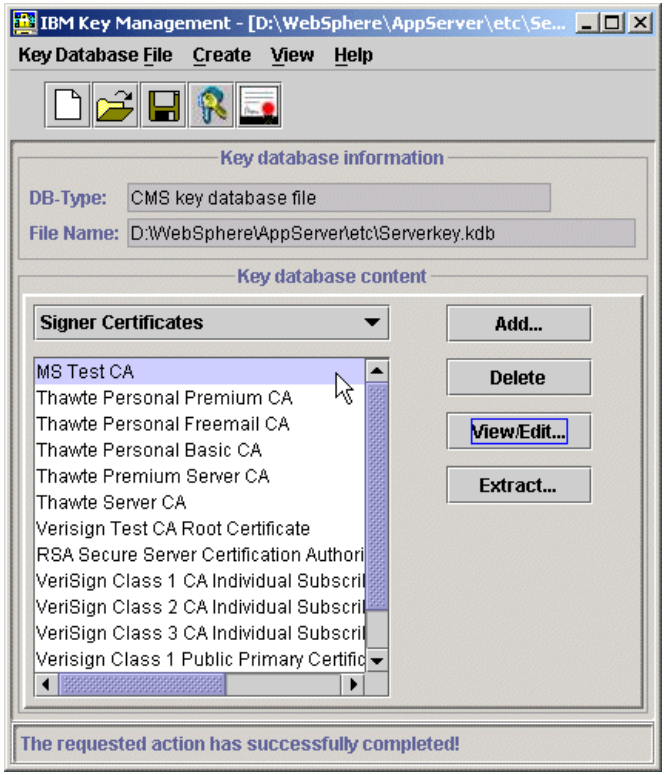


Figure 21-72 IKeyMan - signer certificates list

Obtain a server certificate

The first step to obtaining a server certificate is to create a certificate request using IKeyMan. That request will then be sent on to the Web-based certificate services, which will return a server certificate signed by our new CA. To create the certificate request, do the following:

1. From the main IKeyMan window, select **Personal Certificate Requests** from the drop-down list in the Key database content area, as shown in Figure 21-73.

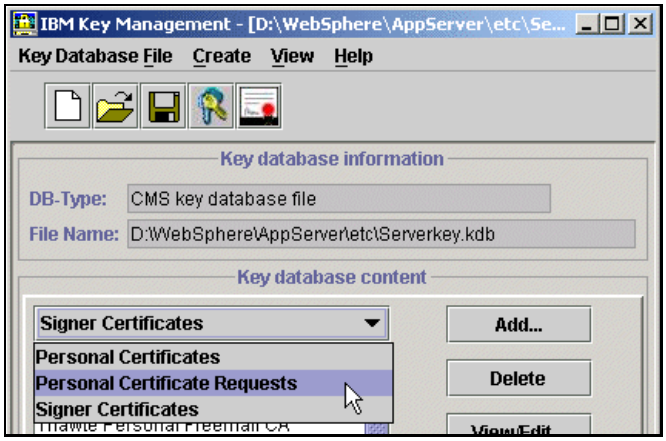


Figure 21-73 Select personal certificate requests

2. Click **New...**, as shown in Figure 21-74.

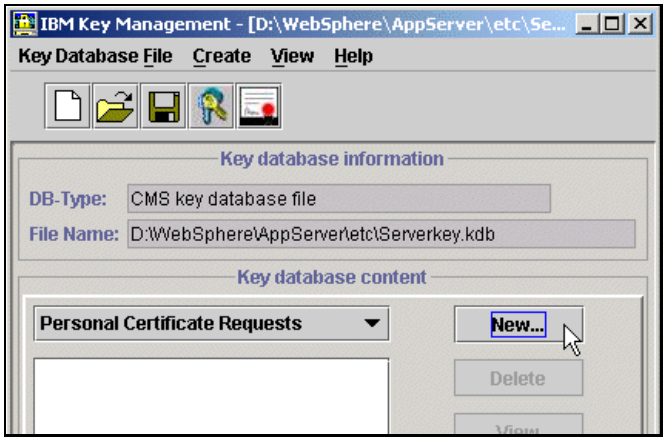


Figure 21-74 Create new personal certificate requests

3. In the Create New Key and Certificate Request window (Figure 21-75), enter the appropriate values and click **OK**. In our case, we are using the following values:
   - For Key Label, choose Server. The Key Label is used to identify the certificate in an IKeyMan list.
   - For Key Size, choose 1024.
   - For Common Name, choose itsohost.itso.ibm.com. The common name is the host name of the server.
   - For Organization, choose ibm.
   - For Country, choose US.
   - For the name of a file in which to store the certificate request, choose <WAS_HOME>\etc\certreq.arm.

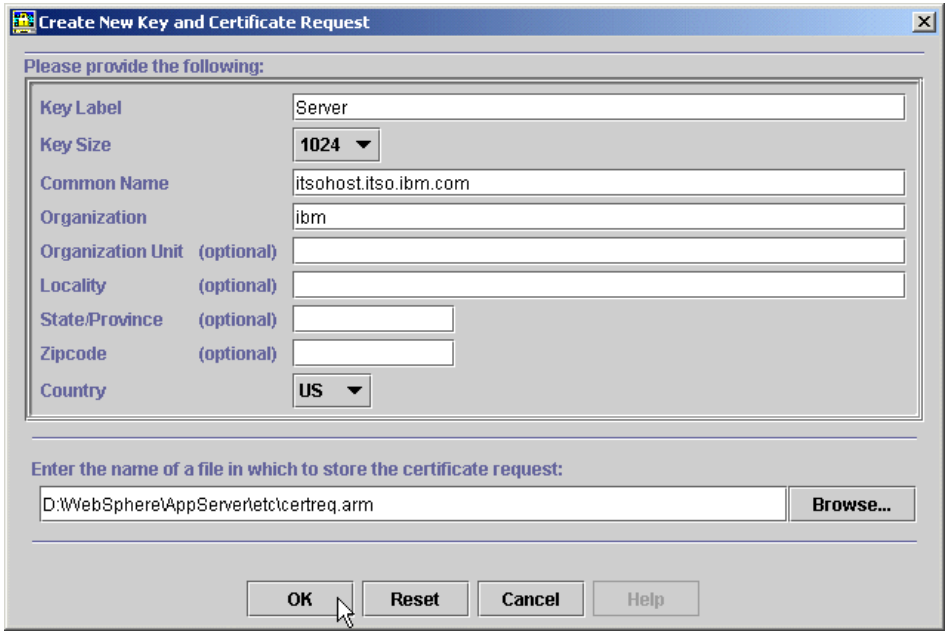


Figure 21-75 IKeyMan - create a request for a new certificate

4. Click **OK** in the Information window.

The Web-based services will now be used to obtain the server certificate. To obtain a new server certificate do the following:

1. From a browser, go to URL `http://marting.develop.com/certsrv/`. In our case, we are using Microsoft Internet Explorer.

2. From the Welcome page (Figure 21-76), select **Request a certificate** and click **Next**.

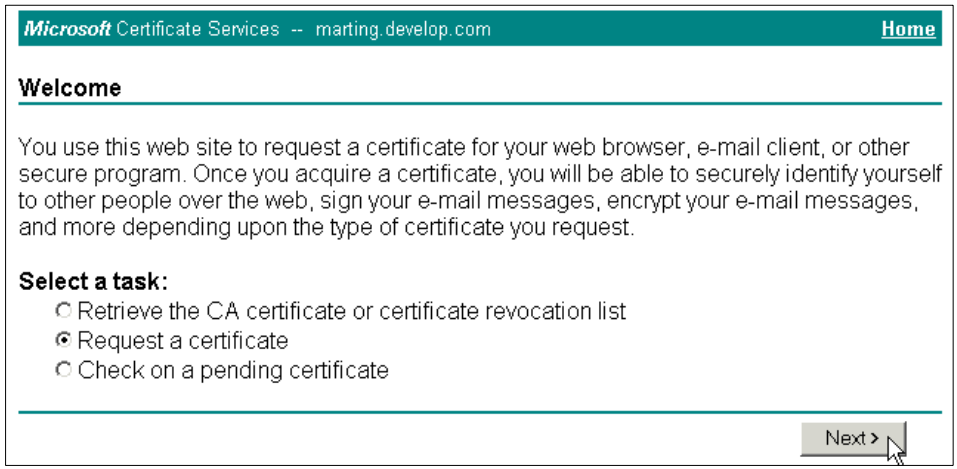


Figure 21-76 Request a certificate

3. In the window shown in Figure 21-77, select **Advanced request** and click **Next**.

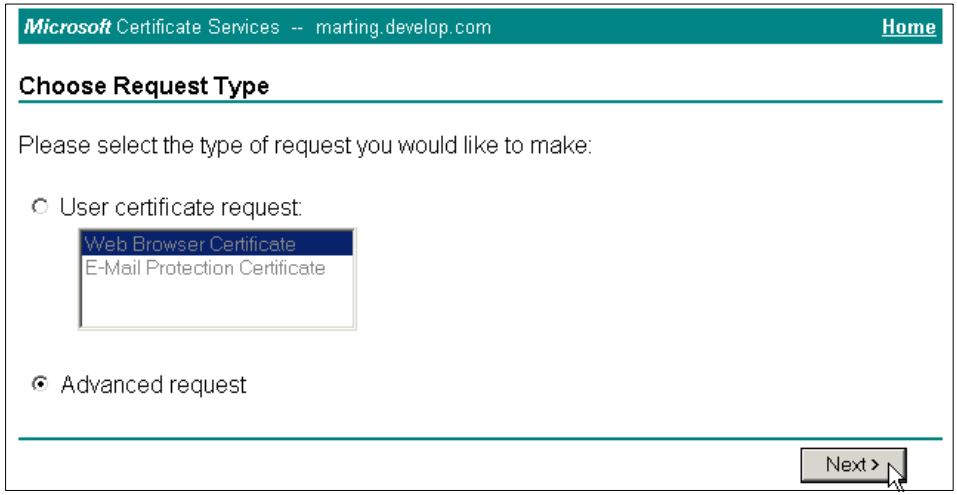


Figure 21-77 Advanced request

4. In the window shown in Figure 21-78, select **Submit a certificate request using a base64 encoded PKCS #10 file...** and click **Next**.

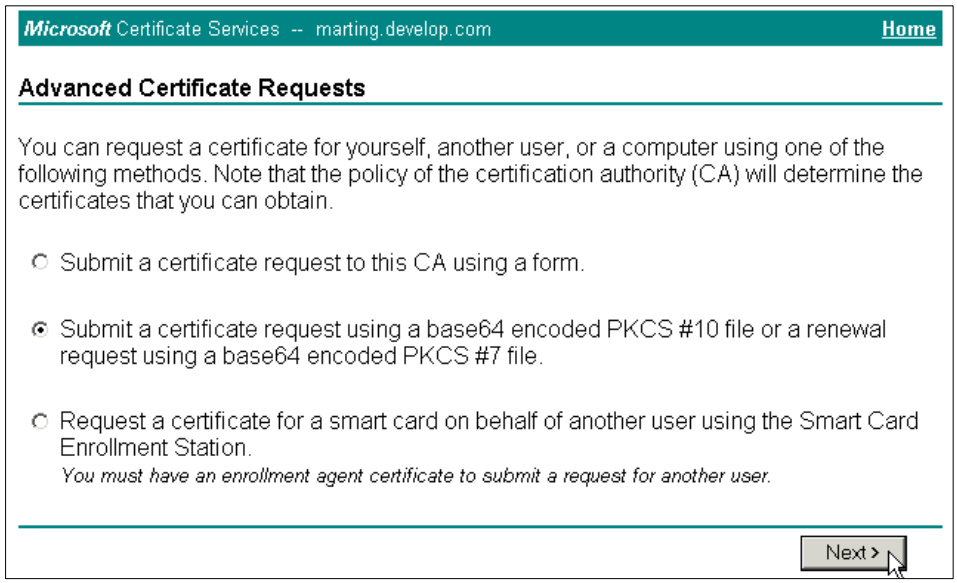


Figure 21-78 Submit a certificate request using a base 64 encoded PCKS #10 file

5. Open the newly created certificate request file (certreq.arm) in Notepad (see Figure 21-79) and copy the entire contents of the file to the clipboard.

Figure 21-79 Copy the entire contents of the certificate request file

6. Back in the certificate services page, paste the clipboard contents into the Saved Request edit box. The results should look something like those shown in Figure 21-80.

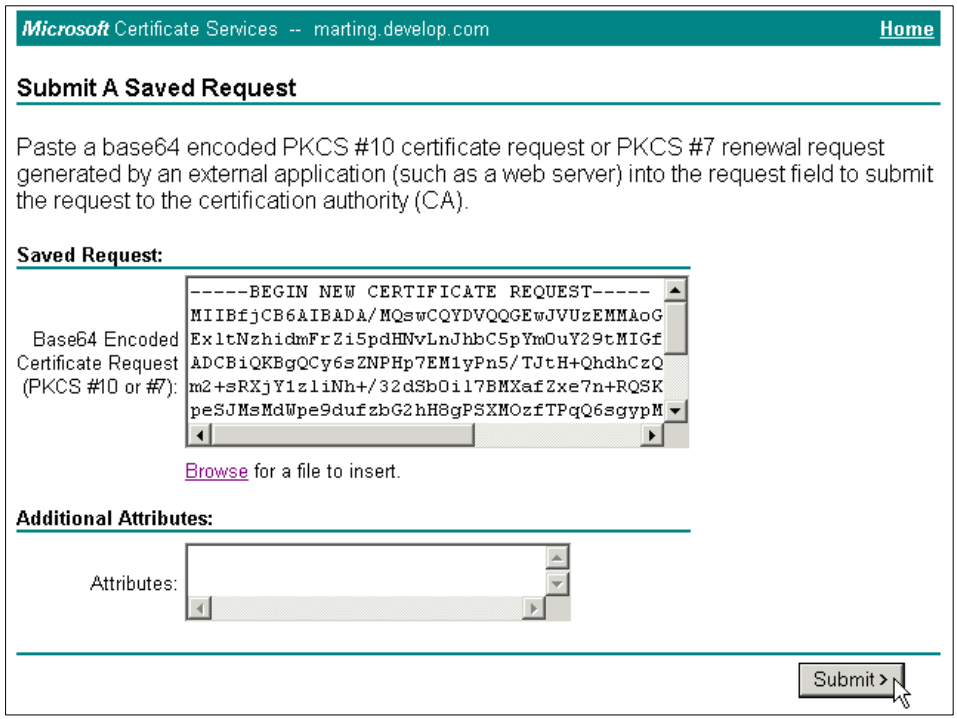


Figure 21-80 Microsoft Certificate Services - submit a request for a new certificate

7. Click the **Submit** button.
8. In the Certificate Issued page (Figure 21-81), select **Base 64 encoded** then click **Download CA certificate**.

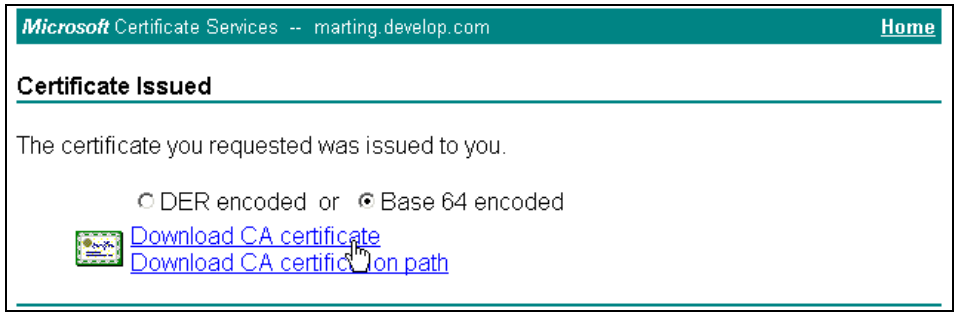


Figure 21-81 Download CA certificate

9. In the File Download window, make sure that **Save this file to disk** is selected, then click **OK**.

10. Specify a directory to save in and a file name, then click **Save**. In our case the directory is <WAS_HOME>\etc and file name is Srvcertnew.cer.

Now, IKeyMan will be used to receive the newly obtained server certificate into the server key database. To receive the certificate, do the following:

1. From the main IKeyMan window, select **Personal Certificates** from the drop-down list in the Key database content area.
2. Click **Receive...**, as shown in Figure 21-82.

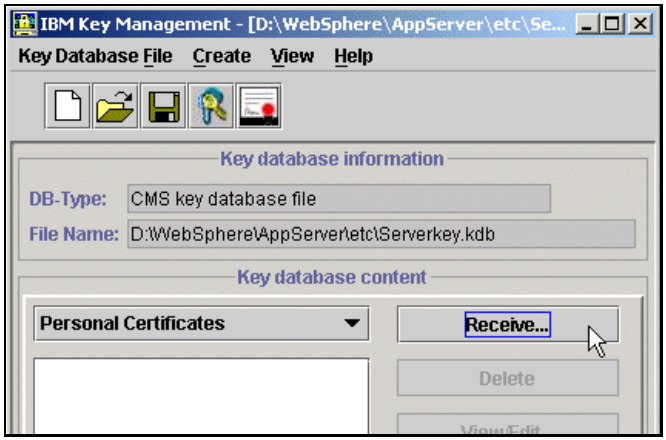


Figure 21-82 Receive a new personal certificate

3. In the Receive Certificate from File window (Figure 21-83):
   - Leave the data type as Base64-encoded ASCII data.
   - Type the Certificate file name. In our case, the name is Srvcertnew.cer.
   - Type the Location. In our case, it is <WAS_HOME>\etc\.
   - Click **OK**.

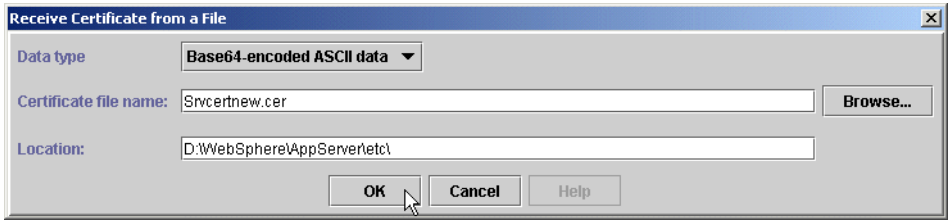


Figure 21-83 Receive Certificate from a File window

4. You should now see a single entry in the Personal Certificates list called Server. See Figure 21-84.

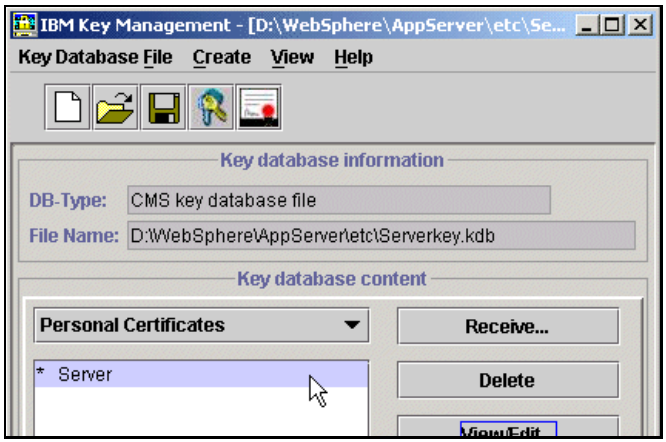


Figure 21-84 The server's certificate in the Personal Certificates list

Obtain a client certificate

The Web-based services will now be used to obtain the client certificate. To obtain a new client certificate do the following:

1. From a browser, go to URL `http://marting.develop.com/certsrv/`. In our case, we are using Microsoft Internet Explorer V5.5.
2. From the Welcome page, select **Request a certificate** and click **Next**.
3. Select **User certificate request**, **Web Browser Certificate**, and click **Next**, as shown in Figure 21-85.

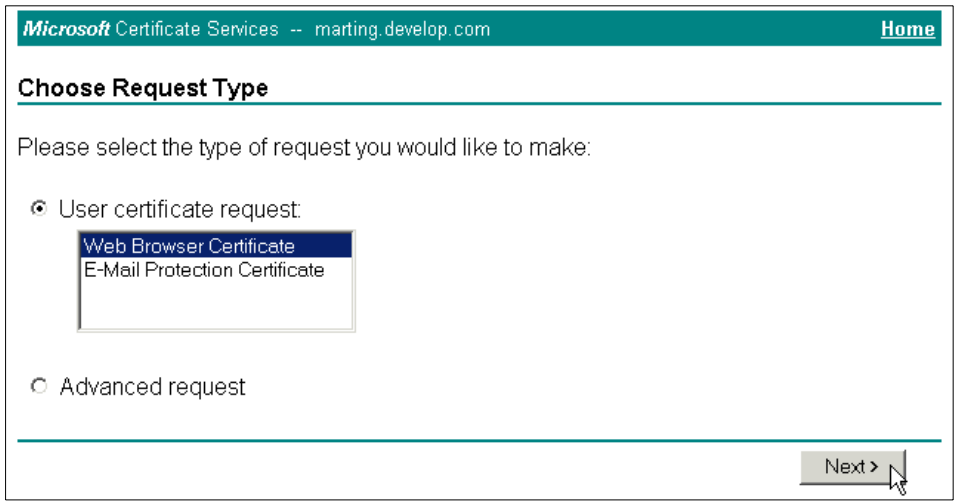


Figure 21-85 User certificate request

4. Click the **More Options** button, then **Advanced Certificate Request** to open the Advanced Certificate Request page.
5. In the Advanced Certificate Request page (Figure 21-86), enter the appropriate values and click **OK**. In our case, we are using the following values:
   - For Name, we entered `John Black`. The name must match the common name of the LDAP user entry. See 21.8.5, “Using WebSphere security certificate mapping filters” on page 827 for detailed information.
   - For Company, we entered `ibm`.
   - For Country, we entered `US`.
   - For Key Size, we entered `1024`.

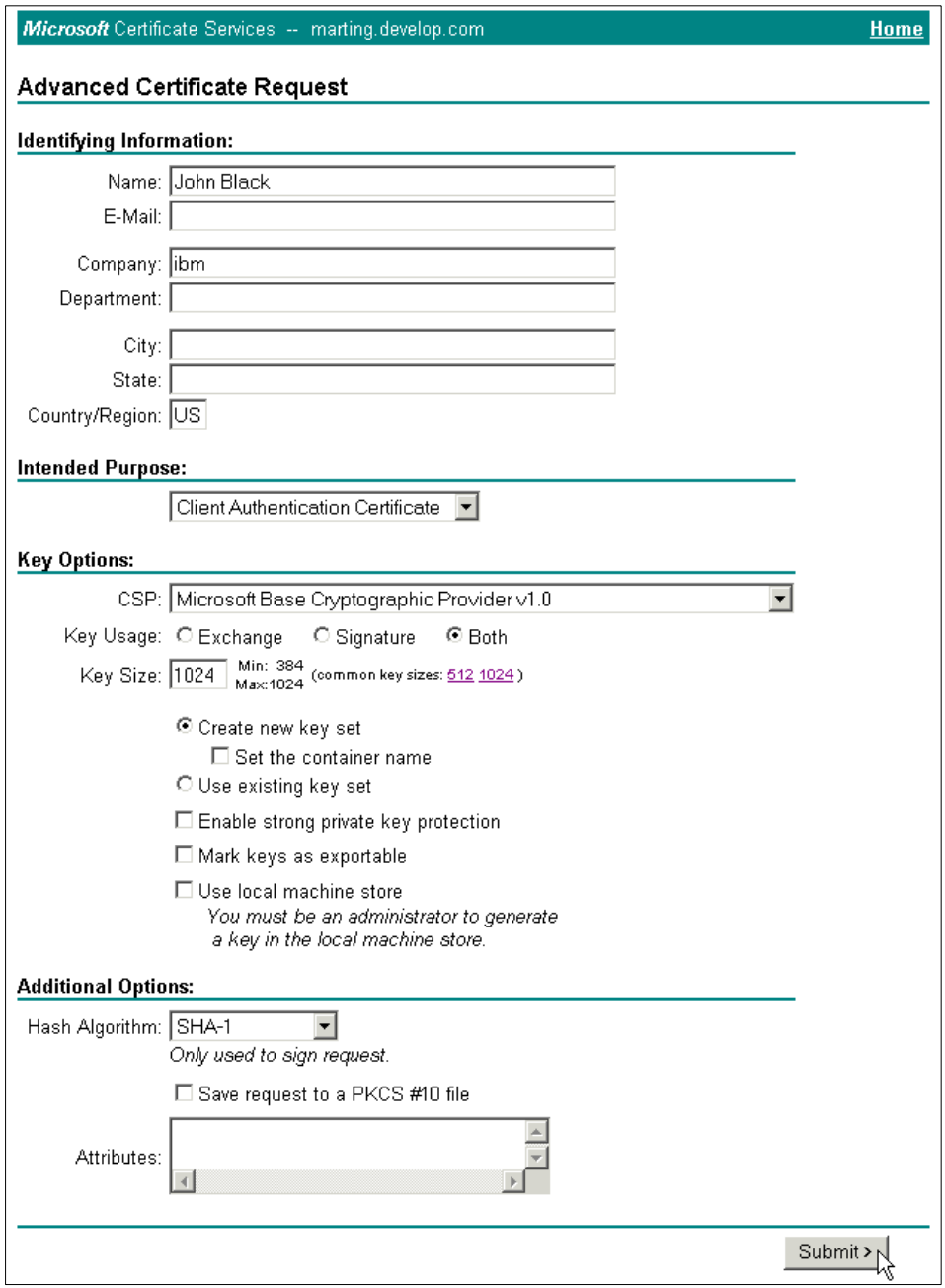


Figure 21-86 Advanced Certificate Request

6. Click the **Submit** button.

7. In the Certificate Issued page click **Install this certificate**, as shown in Figure 21-87.

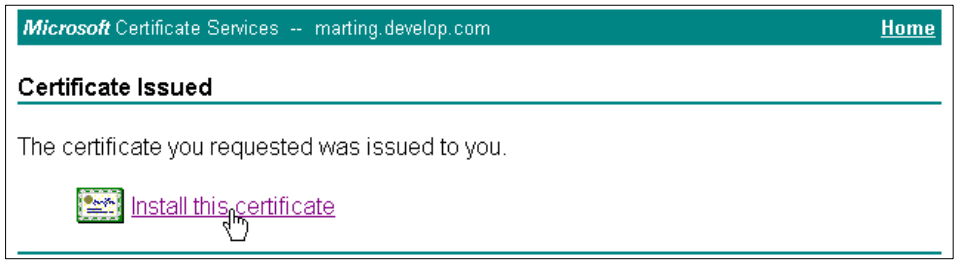


Figure 21-87 Certificate Issued

8. You will be asked if you want to add the following certificate to the Root Store, as shown in Figure 21-88. Click **Yes**.

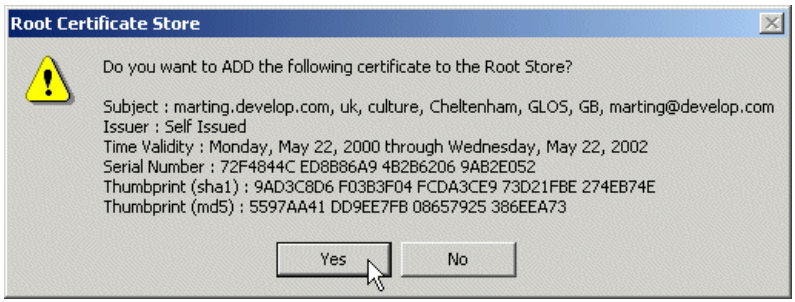


Figure 21-88 Adding the certificate to the Root Store

9. Your new certificate should be successfully installed, as indicated in Figure 21-89.

Figure 21-89 Certificate Installed

Check the browser certificate store

Let's take a look at the John Black client certificate and CA signer certificate that have been to be imported into the Web browser certificate store. We are using Internet Explorer (IE) V5.5 in our example.

To check the client certificate and CA signer certificate imported into the browser, do the following:

1. From the IE menu bar, select **Tools** -> **Internet Options...**.
2. Click the **Content** tab.
3. Click the **Certificates** button, as shown in Figure 21-90.

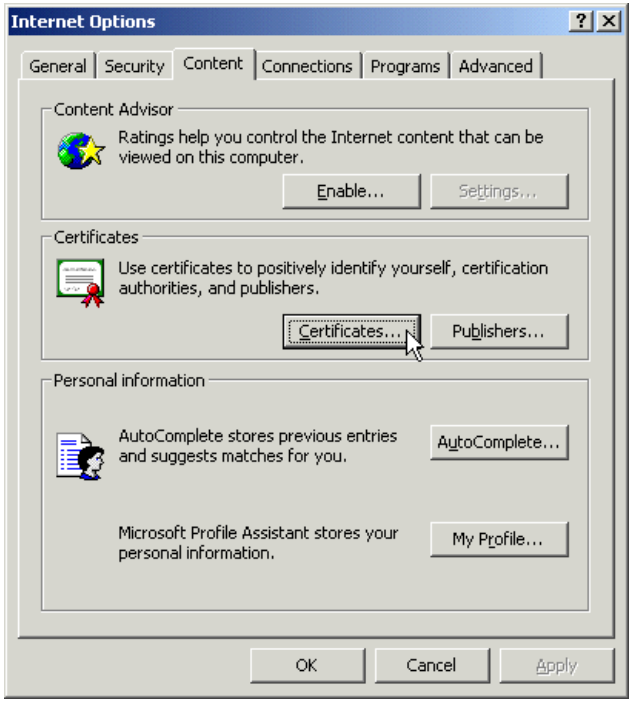


Figure 21-90 Internet Options in Internet Explorer

4. In the Personal tab of the Certificates main window, there should be an entry in the list for John Black, as shown in Figure 21-91. This is the newly imported client certificate.

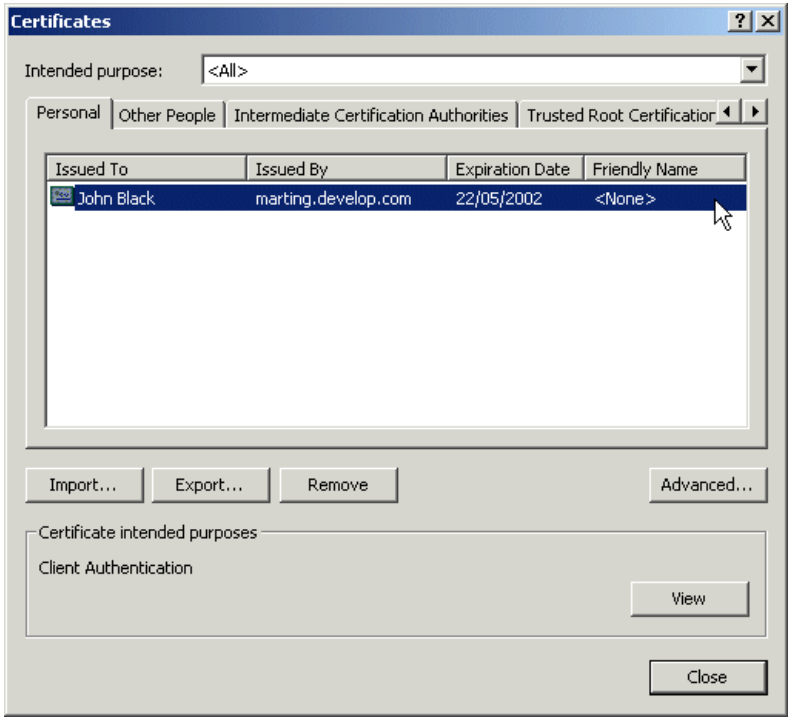


Figure 21-91 IE Certificates - Personal certificates list

5. In the Trusted Root Certification Authorities tab, scroll down to the marting.develop.com entry. This is the newly imported CA signer certificate. You should see something similar to Figure 21-92.

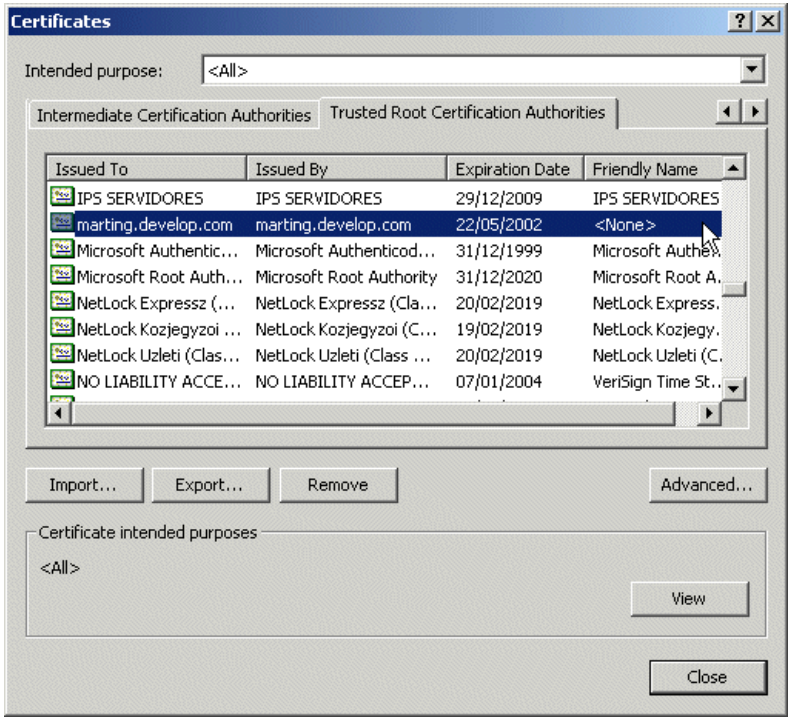


Figure 21-92 IE Certificates - Trusted Root CA list

6. Close the Certificates window and click **OK** in the Internet Options window, shown in Figure 21-93.

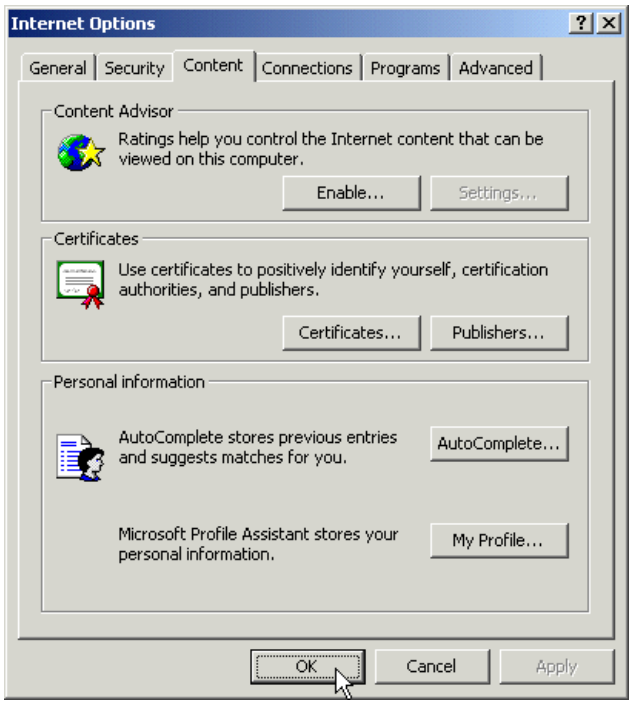


Figure 21-93 Internet Options completed

We have now completed the process for creation of our user John Black certificate.

21.8.3 Configuring the IBM HTTP Server to support HTTPS

An SSL link must be established in order for the browser and server to exchange certificates. The IBM HTTP Server must be configured to allow SSL communications.

To enable SSL for the IBM HTTP Server (IHS), do the following:

1. Make sure that the IBM HTTP Server and the IBM HTTP Administration Server are running.
2. Also make sure that the IHS Administration user ID and password have been set. If not, use the htpasswd utility shipped with the product to set the ID and password.
3. Using a browser, access the IHS welcome page by specifying `http://hostname` as the URL and substitute the host name of your machine. In our case, the URL is `http://localhost`. See Figure 21-94.

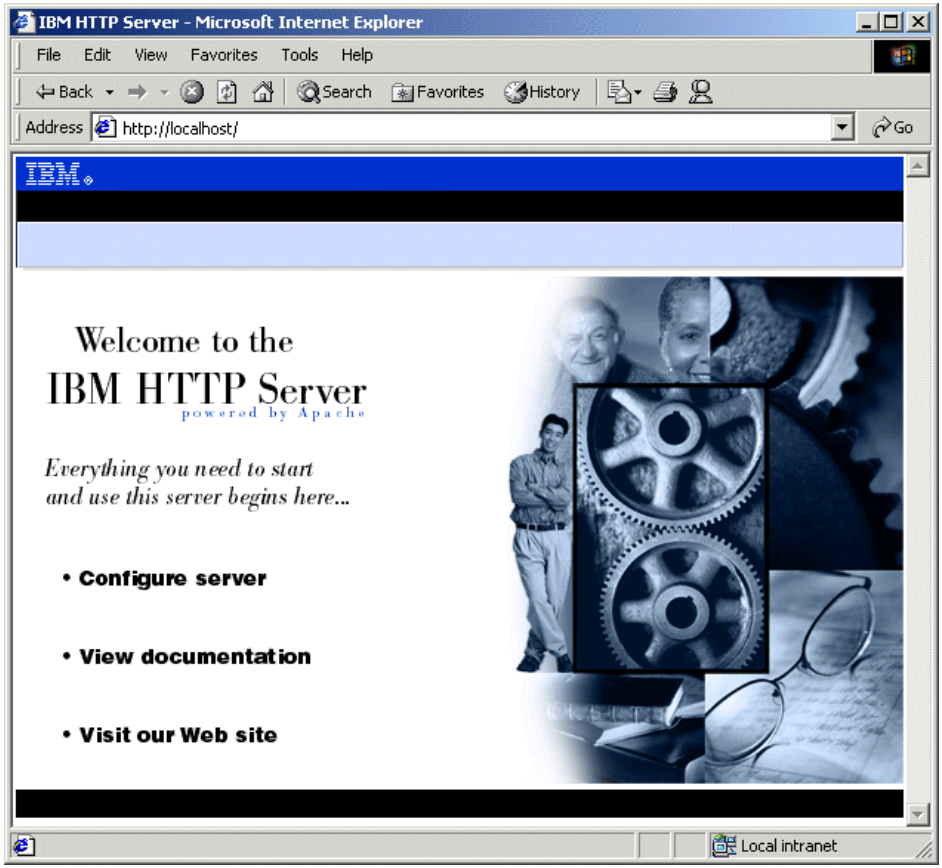


Figure 21-94 IBM HTTP Server (IHS) welcome page

4. Click **Configure server**. Then, type the IHS administration user name and password and click **OK**. You should come up in the IHS Administration window shown in Figure 21-96. Be patient. It may take several seconds for the left navigation frame to appear.

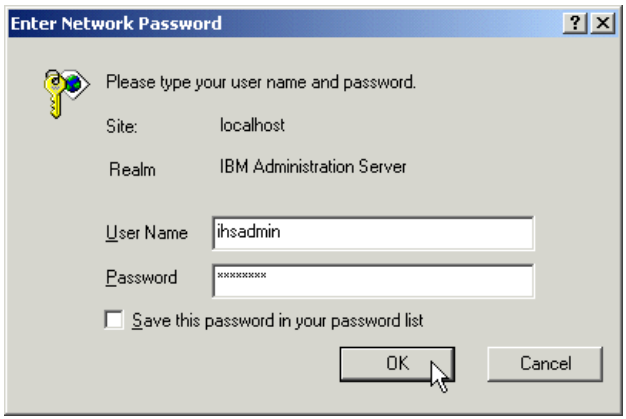


Figure 21-95 IHS - authorization for server configuration

5. To configure the security module, expand **Basic Settings** -> **Module Sequence** in the left navigation frame.

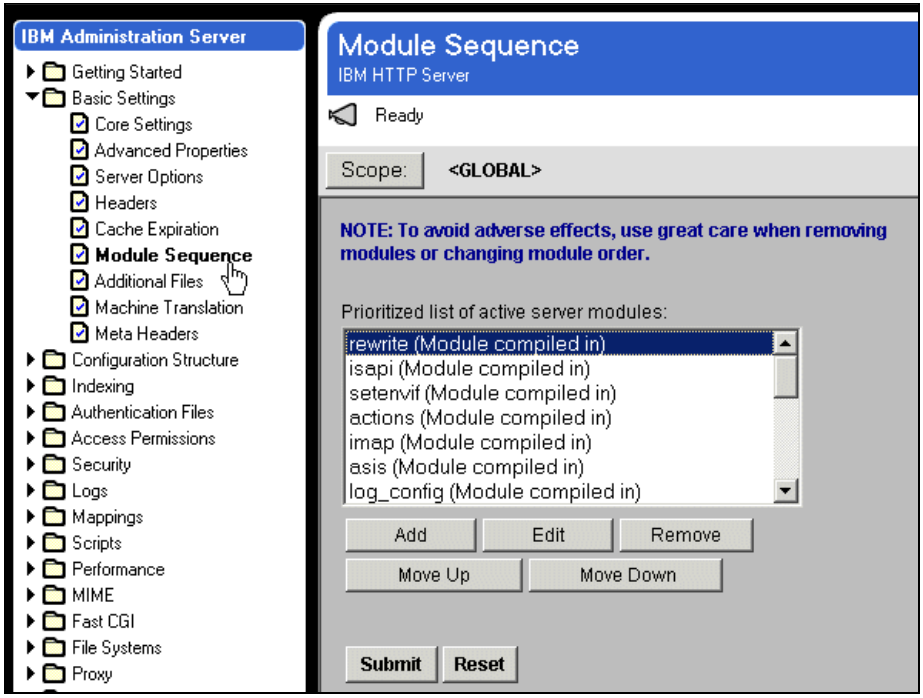


Figure 21-96 IHS - adding a module to the module sequence

6. Make sure the Scope is GLOBAL. If the value to the right of the Scope: button is not <GLOBAL>, click the button and select **<GLOBAL>**.

7. In the Module Sequence frame, click the **Add** button. You will see Figure 21-97.
8. Select **Select a module to add:** and open the drop-down list. Scroll to near the bottom of the list and select **ibm_ssl (IBMModuleSSL128.dll)**. The Module DLL will be placed to the right.

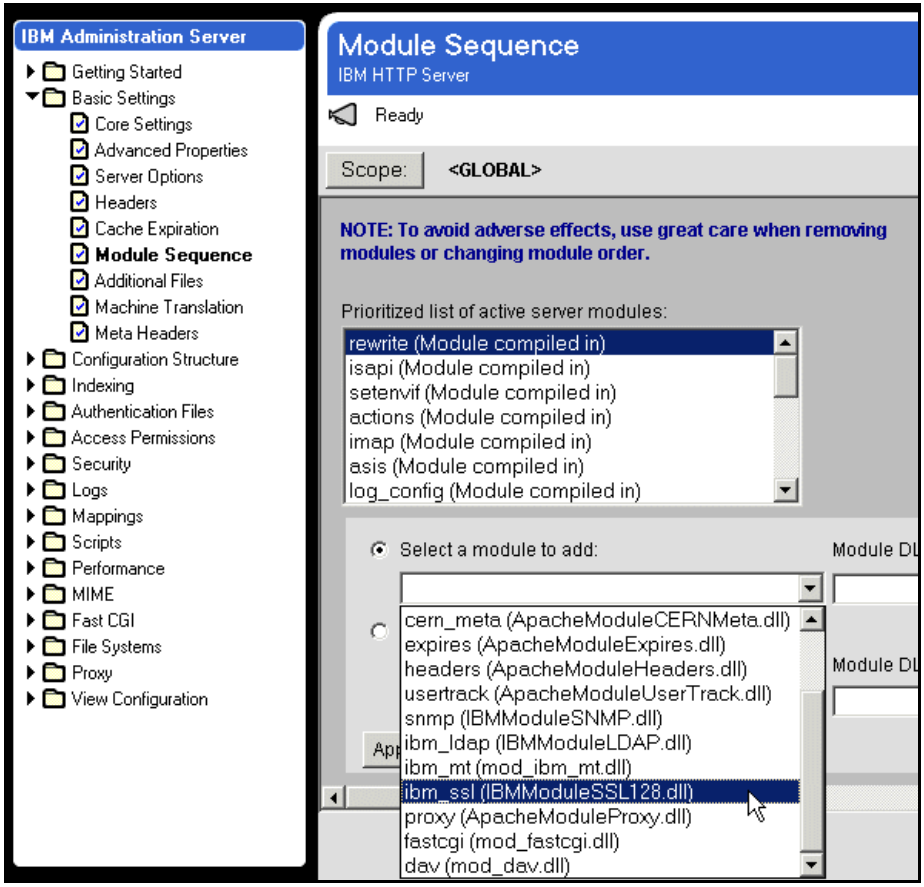


Figure 21-97 IHS - selecting a module to add to the module sequence

9. Click **Apply** -> **Close** -> **Submit**.

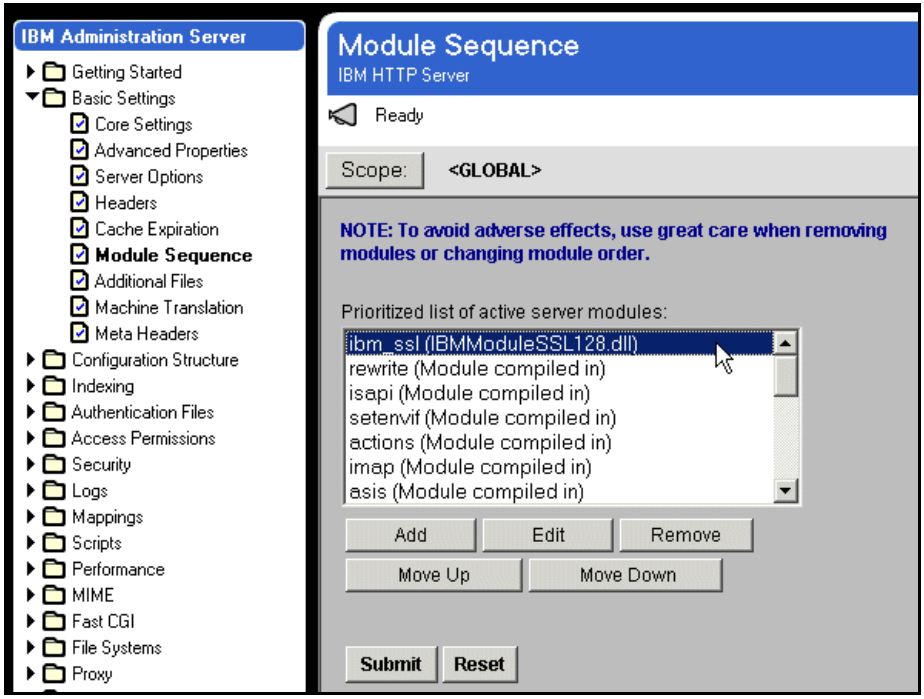


Figure 21-98 IHS - submitted module sequence

10. To set up the secure host IP and an additional port for the secure server, first select **Basic Settings** -> **Advanced Properties** in the left navigation frame of the window shown in Figure 21-99. Then, make sure the Scope is GLOBAL.

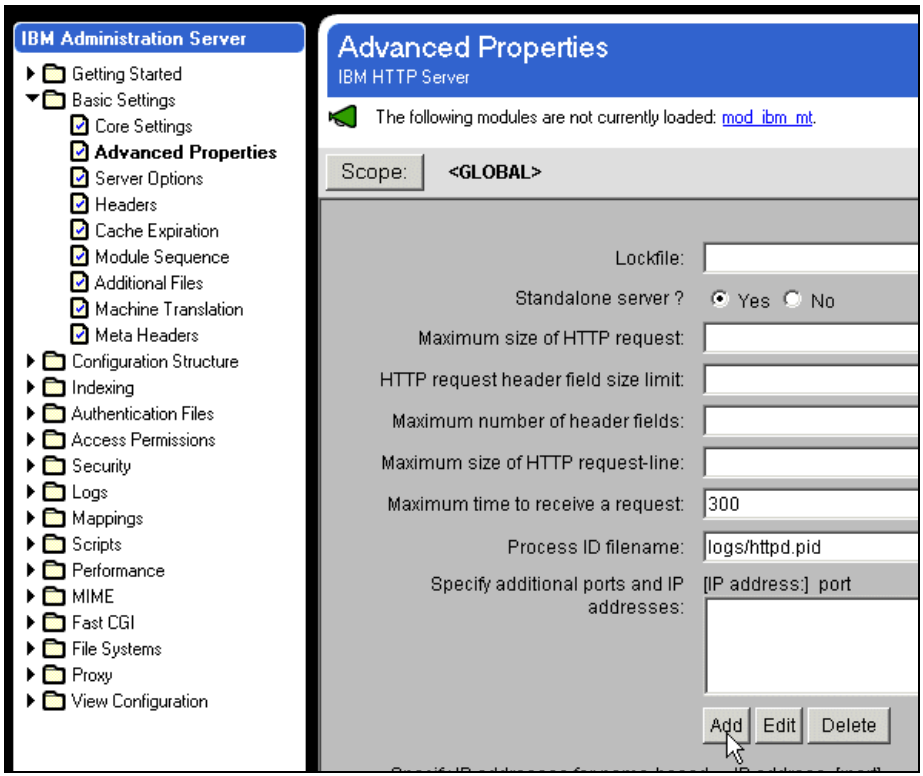


Figure 21-99 IHS - adding the SSL port

11. Click the **Add** button for the Specify additional ports and IP addresses field. Leave the IP address field empty and enter 443 in the port field.

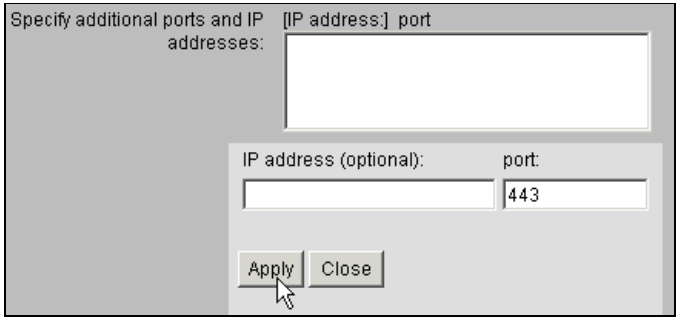


Figure 21-100 IHS - adding the SSL port

12. Click **Apply** -> **Close** -> **Submit**.

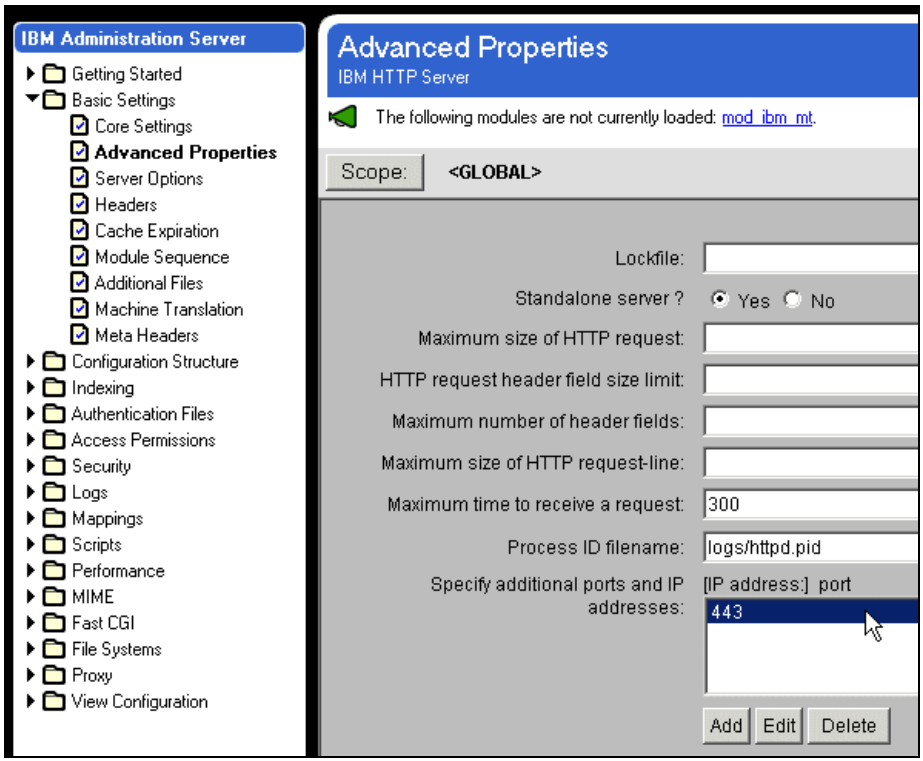


Figure 21-101 IHS - submitted SSL port

13. To set up the virtual host structure for the secure server, do the following:
    - In the left navigation frame, select **Configuration Structure** -> **Create Scope**. Then, make sure the Scope is GLOBAL. In this case, <Global> in the tree structure in the right panel should be highlighted.
    - Select **VirtualHost** in the Select a valid scope to insert within the scope selected in the right panel field.
    - Type the virtual host IP address or fully qualified domain name. In our case, the value is itsohost.itso.ibm.com.
    - Type the virtual host port value as 443.
    - Type the server name. In our case, the name is itsohost.itso.ibm.com.
    - Leave the alternate name(s) for host blank.
    - Click the **Submit** button.

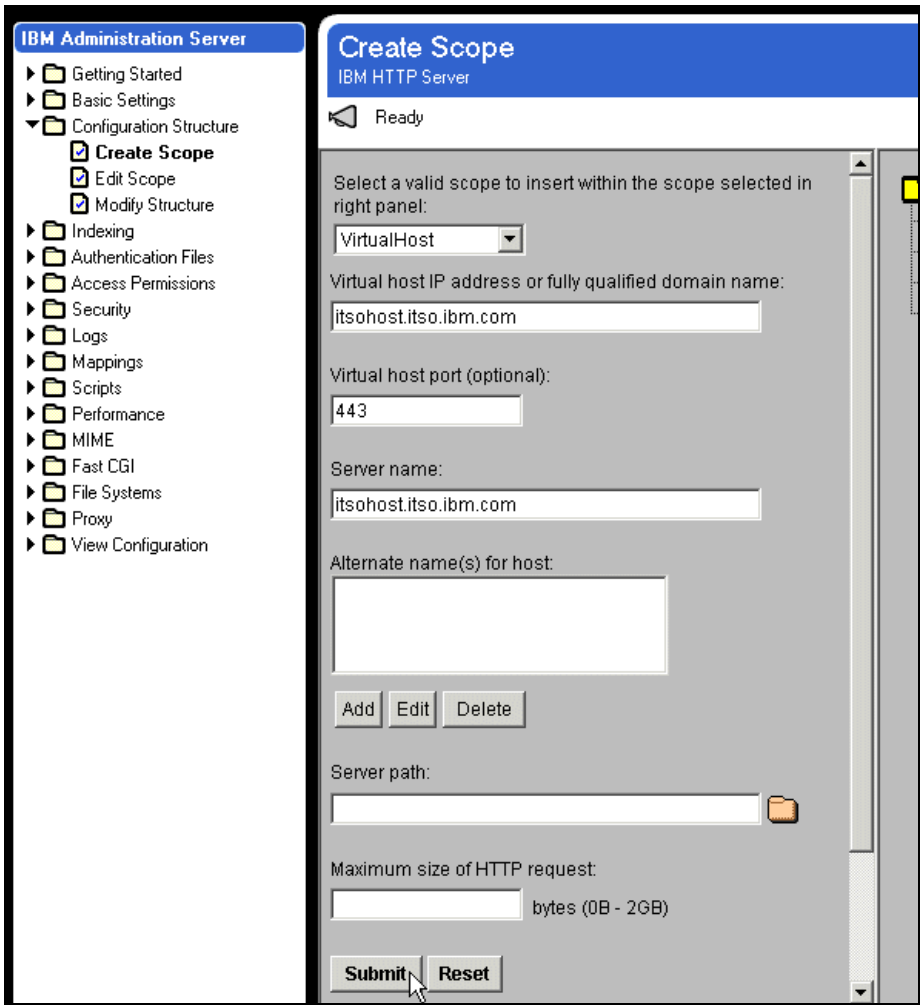


Figure 21-102 IHS - creating a virtual host

14. To set up the virtual host document root for the secure server, do the following:
    - In the left navigation frame, select **Basic Settings** -> **Core Settings**.
    - Make sure the Scope is set to the virtualhost you are now working with. In our case, click the **Scope** button, then select <VirtualHost itsohost.itso.ibm.com:443>.
    - Type the Server name as a fully qualified domain name. In our case, the name is itsohost.itso.ibm.com.

– Type the Document root directory name. In our case, the name is D:\IBM HTTP Server\htdocs.

– Click the **Submit** button.

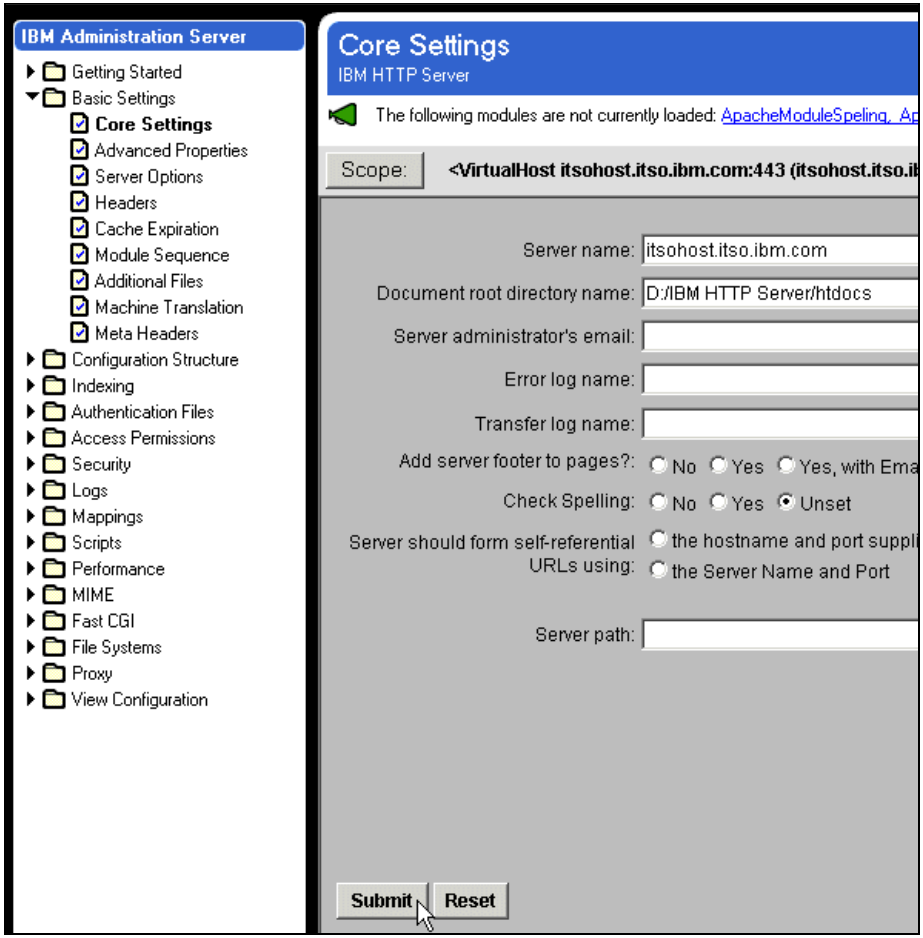


Figure 21-103 IHS - setting the virtual host document root

15. To set the keyfile and the SSL timeout values for the secure server, do the following:

– In the left navigation frame, expand **Security** -> **Server Security**.

– Make sure the Scope is GLOBAL.

– Select **No** for Enable SSL. This will disable SSL globally, but we will soon enable SSL for the virtual host we are working with.

- Type the path and file name for the keyfile. In our case, the value is <WAS_HOME>\etc\Serverkey.kdb.
- Type a Timeout value for SSL Version 2 session IDs (100 seconds).
- Type a Timeout value for SSL Version 3 session IDs (1000 seconds).
- Click the **Submit** button.

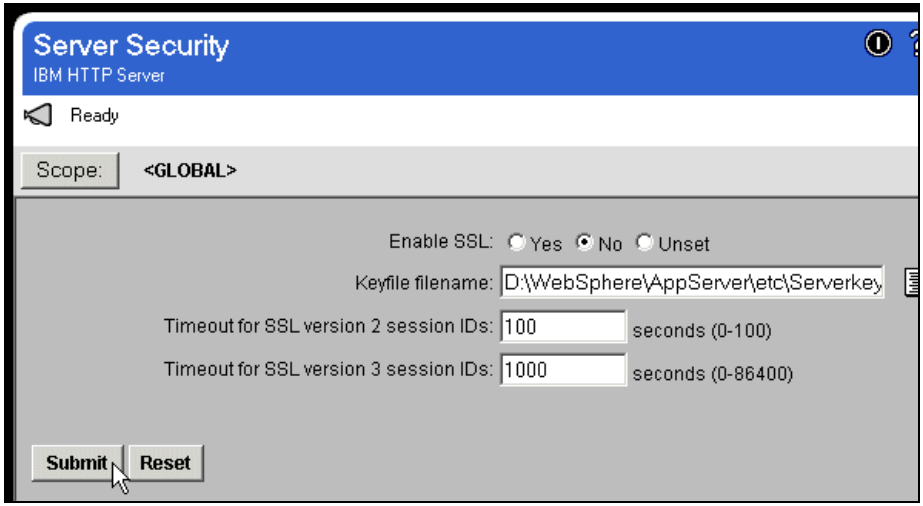


Figure 21-104 IHS - setting Global security values

16. To enable SSL and set the mode of operation of client authorization for the virtual host, do the following:
    - In the left navigation frame, select **Security** -> **Host Authorization**.
    - Make sure the Scope is set to the virtual host you are working with. To do this, click the **Scope** button and select the virtual host. In our case, the Scope is <VirtualHost itsohost.itso.ibm.com:443>.
    - Select **Yes** for Enable SSL.
    - Select **required** for the mode of client authentication to be used. This will require a certificate exchange to be made between the browser and the Web server.
    - Click the **Submit** button.

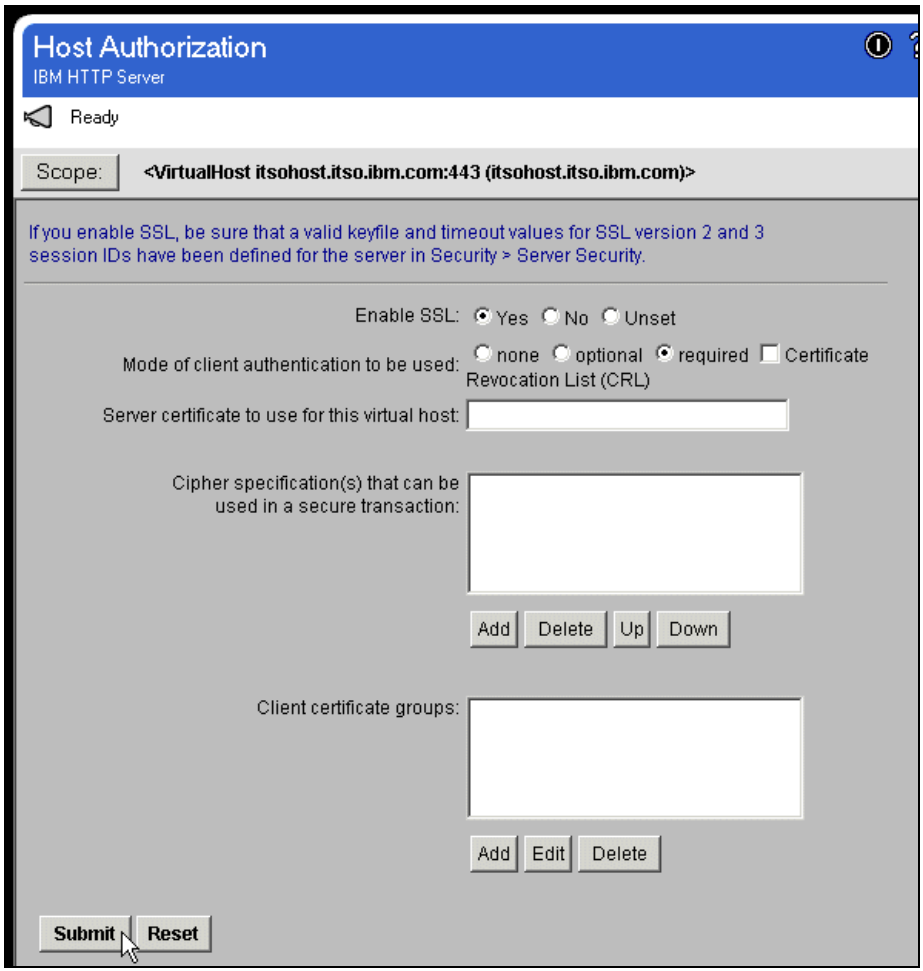


Figure 21-105 IHS - setting security values for the virtual host

17. Restart the server.

Note: In some cases there may be a problem loading modules on the restart after the IHS configuration changes have been made on Windows 2000. You may see the following messages:

```
Could not start the IBM HTTP Server service on \\...
Error 1067. The process terminated unexpectedly.
```

You can try to circumvent the problem by commenting out the following line in the IHS httpd.conf file:

```
ClearModuleList
```

21.8.4 Configuring the WebSphere virtual host alias for SSL

We need to define a WebSphere virtual host alias for our SSL connection. To do this, do the following:

1. Make sure that the WebSphere administrative server and administrative console are up and running.
2. In the administrative console select **Virtual Hosts**.
3. Click the **Add** button in the General tab.
4. Type in the alias name in the Host Aliases list, being sure to append the SSL port number. In our case, the alias name is *:443.
5. Click the **Apply** button, as shown in Figure 21-106 on page 827.

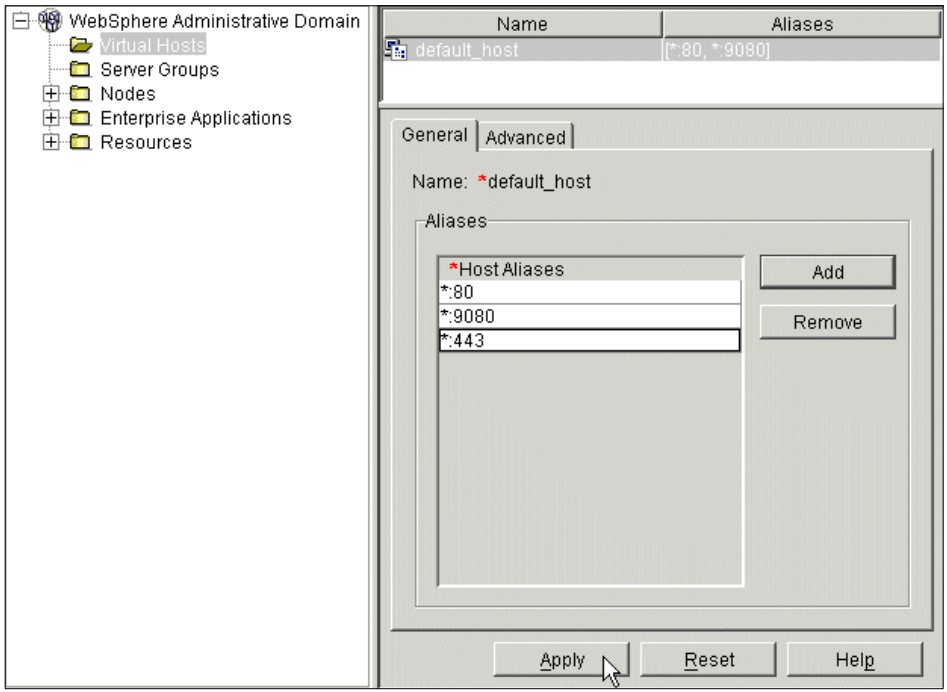


Figure 21-106 Adding the WebSphere virtual host alias for SSL

6. Regenerate the plug-in configuration. You can do this by right-clicking the node and selecting **Regen Webserver Plugin** from the pop-up menu.

21.8.5 Using WebSphere security certificate mapping filters

In our example, we want authentication to succeed if the distinguished name value in the client certificate exactly matches a distinguished name in the LDAP registry. This is the default certificate mapping technique used by WebSphere. However, there may be some cases where you want authentication to succeed based on other comparisons. You can do this by specification of a certificate filter.

Suppose you want authentication to succeed if the value of the common name (CN) in the certificate matches (maps to) the value of the uid attribute in the LDAP registry. To define the certificate filter to make this happen, do the following:

1. Start the Security Center by selecting **Console** -> **Security Center...** from the WebSphere Administrative Console main menu.

2. From the Authentication tab of the Security Center, seen in Figure 21-6 on page 745, click the LDAP Settings **Advanced...** button.
3. In the Certificate Mapping field, select **Certificate Filter:** from the drop-down menu (the default value is Exact Distinguished Name).
4. In the Certificate Filter field, type in the filter value. In our case, the filter value is `(uid=${SubjectCN})`. This specification will cause WebSphere authentication to “map” the uid attribute in the LDAP directory to the CN value in the certificate. Figure 21-107 shows the specification values. For other properties that can be used as filter values, see “Advanced properties for configuring LDAP support” in the WebSphere InfoCenter.
5. Click the **OK** button.
6. Then, back in the User Registry window, click the **Finish** button.

Remember that our sample scenario uses Exact Distinguished Name mapping, so if you want to try certificate filters, consider doing so after you have tested the sample scenario, as described in 21.8.6, “Testing with a secured application” on page 828.

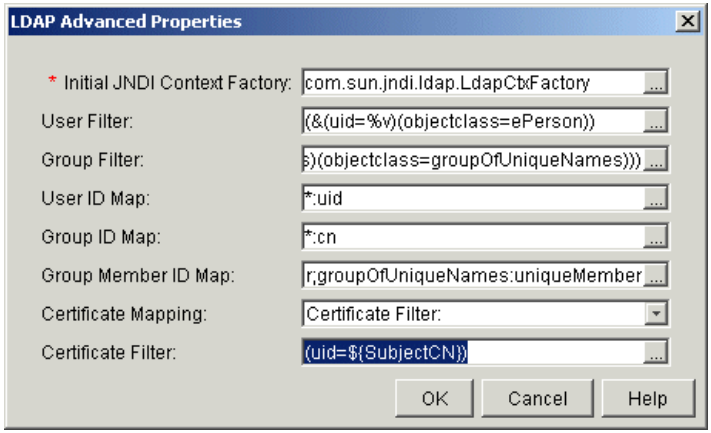


Figure 21-107 LDAP Advanced Properties window

21.8.6 Testing with a secured application

You are now ready to test client certificate-based authentication with a secured Web application.

You can use the WebBank sample application with the client certificate authentication method enabled, as described in 21.3.5, “Configuring client certificate authentication” on page 770.

Alternatively, you can simply change the deployment descriptor for the default_app Web module, so it uses the client certificate authentication method instead of the basic authentication method.

To test client certificate-based authentication by changing the default_app deployment descriptor:

1. Open the default_app deployment descriptor for editing:

   `<WAS_HOME>\installedApps\sampleApp.ear\default_app.war\WEB-INF\web.xml`

   Note: Do not edit the deployment descriptor in the <WAS_HOME>\installedApps directory in a production environment. It is described here as a convenience, for use only in a test or development environment.

2. Change the auth-method property from BASIC to CLIENT-CERT.
3. Save the deployment descriptor.
4. Restart the Default Server application server to reload the new deployment descriptor.
5. Try to access a protected resource in the default_app Web module. Try the Snoop Servlet at URL `https://<your.server>/servlet/snoop`.
   a. You should be challenged by your Web browser to select a client certificate.

      In our case, we select the **John Black** client certificate.

   b. The Snoop Servlet should be displayed. You should see the following in the Snoop Servlet Request Information:

      - Remote User is John Black
      - Authorization scheme is CERT

      If you get a message stating `You are not authorized to view this page` without being challenged for your client certificate, make sure you used `https` in the URL.

      If you can't connect at all, make sure you used the correct Web server virtual host name in the URL, set up for SSL port 443 in 21.8.3, "Configuring the IBM HTTP Server to support HTTPS" on page 815.

21.9 Global default SSL configuration

Out of the box, WebSphere provides a predefined global default SSL configuration. The default SSL configuration is used by the following WebSphere Application Server components to secure their communications:

- HTTPS transport.
- ORB: the application server's client and server Object Request Broker.
- LDAPS: the administrative server's secure connection to the LDAP registry used for authentication.

These settings allow the SSL properties to be managed using one centrally controlled configuration. Alternatively, the settings can be overridden by an individual application server's configuration. Remember, changing the global settings may affect all of the application servers in the WebSphere domain. Also, the administrator must restart the application server before changes to the SSL settings take affect.

Note: The default key and trust files in the default global SSL configuration contains a test certificate. These are only intended for use in a *test* environment. The default key files should *never* be used in a production environment because the private keys are the same on all the WebSphere installations. For an example of generating a key file, see Part 3, "Installing WebSphere" on page 125.

To see the current global default SSL settings:

1. Select **Console -> Security Center...** from the WebSphere Administrative Console main menu, as shown in Figure 21-108 on page 831.

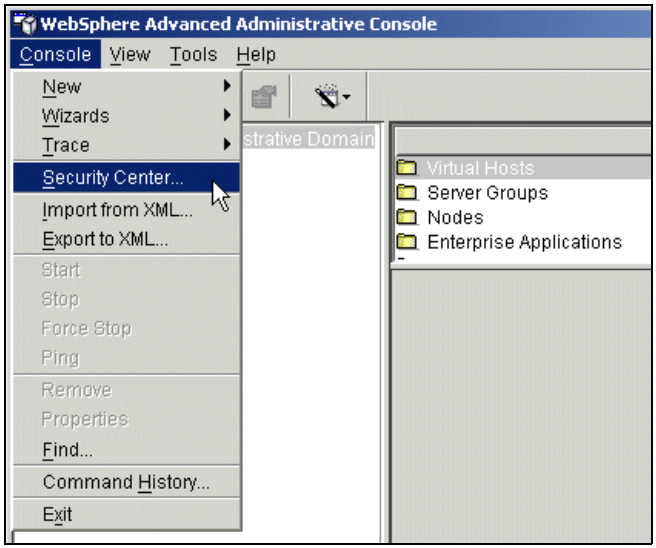


Figure 21-108 Starting the Security Center

2. Click the **Default SSL Configuration** button, as shown in Figure 21-109.

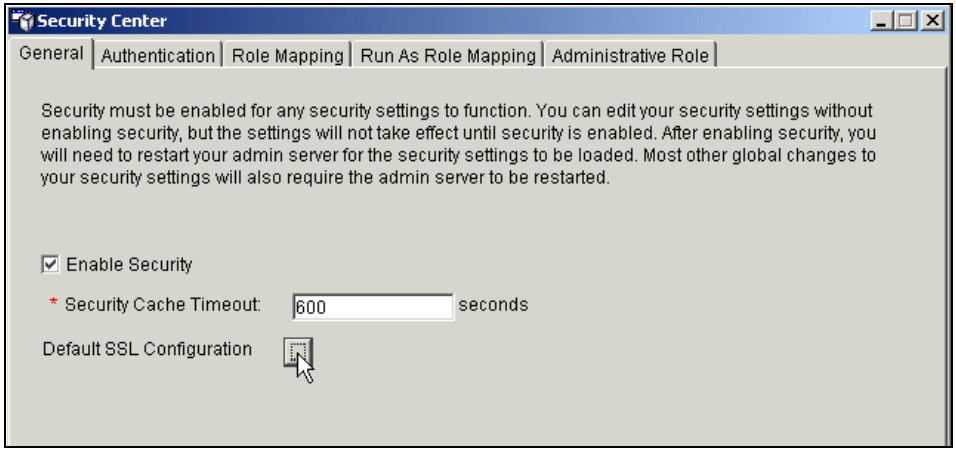


Figure 21-109 Accessing the current default SSL configuration

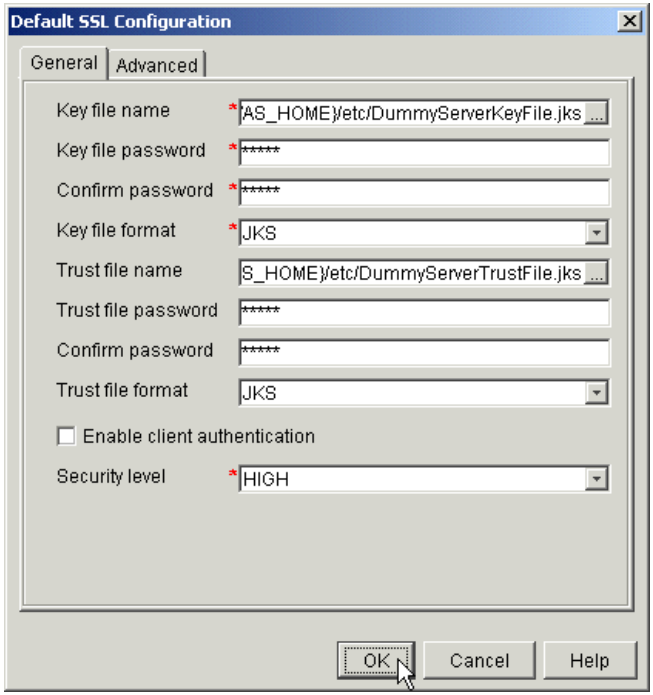


Figure 21-110 The global default SSL settings

Note: The default SSL configuration managed by the Security Center is shared by multiple nodes in the same security domain. The WebSphere Application Server installation path can be different on different host machines. Hence it is not always possible to use an absolute file path when specifying the location of the key store and the trust store.

IBM WebSphere Application Server uses a symbolic link {WAS_HOME} (which equates to *product_installation_root*) to locate the key store and trust store. For example, the key file name can be defined as ${WAS_HOME}/etc/ServerKeyFile.jks.

The ServerKeyFile.jks must exist on all the host machines in the "etc" subdirectory of the product_installation_root. The contents in the key files can be different on different nodes, but the file names should match.

The fields shown in Figure 21-110 on page 832 are:

- Key file name

  The full path name to the WebSphere key file, which can use the {WAS_HOME} symbolic link, as noted above.

  The key store may contain both personal certificates and signer certificates.

  The administrator can choose to use the key file generated using the iKeyMan utility or, if a crypto token hardware or software device is available, the key file associated with that device. If the iKeyMan utility is used, it is important to remember that the iKeyMan supplied with WebSphere and the iKeyMan supplied with the IBM HTTP Server (used with the WebSphere plug-in) are *not* the same. They are designed as access key files of different formats so be careful to use the correct tool when creating or populating a key file or trust file. For an example of creating and populating the key file for the WebSphere plug-in and the key file used by WebSphere, see Part 3, “Installing WebSphere” on page 125.

- Key file password/confirm password

  The password on the key file. The DummyServerKeyFile password is “WebAS”.

- Key file format

  The format of the WebSphere key file. By default it is the JKS format.

- Trust file name

  The fully qualified path to a trust file containing the public keys, which can use the {WAS_HOME} symbolic link, as noted above.

  As with the SSL key file, this can be created with the IKeyMan utility, or it may correspond to a hardware or software device. Unlike the key file, no personal certificates are referenced; only signer certificates are retrieved. If a trust file is not specified but the key file is specified, then the SSL key file is used for the retrieval of signer certificates as well as personal certificates.

- Trust file password/confirm password

  The password for the trust file. The DummyServerTrustFile password is “WebAS”.

- Trust file format

  The format of the WebSphere trust file. By default it is JKS format.

- Enable client authentication

  Whether the server and client should prove their identities through an exchange of keys. The SSL server is always authenticated to the client. If client authentication is enabled, the SSL client is also authenticated to the server. By default, client authentication is disabled. For an example of how to set up client authenticated SSL, see Part 3, “Installing WebSphere” on page 125. Also see the InfoCenter article, “Configuring SSL in WebSphere Application Server”.

- Security level

  The security level can be High, Medium, or Low and is a user-friendly way of enabling a certain set of cipher suites. The security level can be overridden by giving an explicit value to the dynamic property named com.ibm.ssl.EnabledCipherSuites (see Dynamic Properties on the Advanced tab in Figure 21-111 on page 834). Click the **Help** button to see the list of cipher suites enabled for each security level.

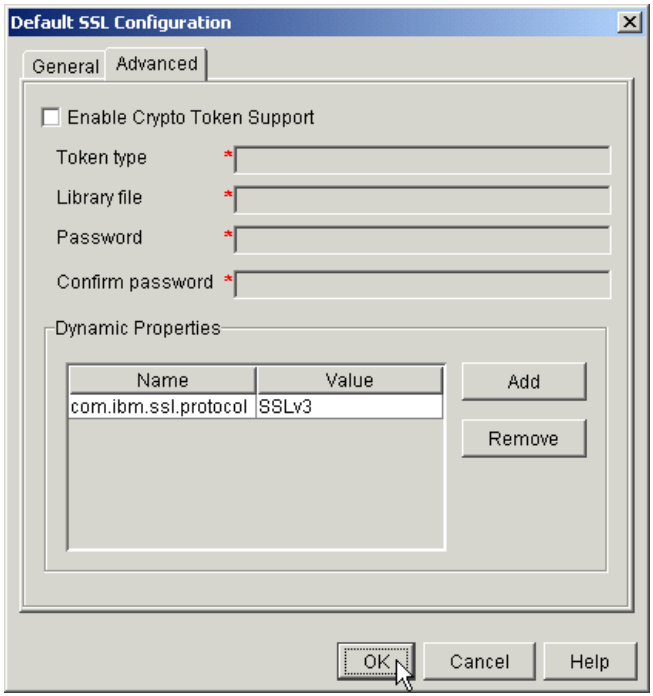


Figure 21-111 Global default SSL advanced settings

3. To see the advanced settings for the global default SSL configuration click on the Advanced tab, as seen in Figure 21-111 on page 834.

   There are two set of properties on this screen. These are:

– Crypto token support:

  The first set enables the use of crypto hardware or software devices.

  • Enable Crypto Token Support

    If this is selected cryptographic token support is enabled and the values on the Crypto Token panel are used. After enabling Crypto Token support, stop your application server and start it again for the change to take effect.

    A crypto token is a hardware or software device that has a built-in key store implementation. The exact values for the following fields should be specified in the documentation of the cryptographic device.

    For more information on crypto token support, see the InfoCentre article, “Cryptographic token support”.

  • Token Type

    The type of token, such as PKCS#11.

  • Library File

    The DLL or shared object that implements the interface to the cryptographic device.

  • Password/confirm password

    The password for the cryptographic device.

– Dynamic Properties

  These are the second set of parameters in this window and are not directly related to the Crypto Token fields. The Dynamic Properties are name-value pairs that the administrator can use to configure additional SSL settings beyond those available in the General panel, seen in Figure 21-110 on page 832. Possible values are:

  • com.ibm.ssl.protocol

    This is the SSL protocol to be used (including its version). The possible values are SSL, SSLv2, SSLv3, TLS, or TLSv1. The default value, SSL, is backward-compatible with the other SSL protocols.

  • com.ibm.ssl.keyStoreProvider

    The name of the key store provider to use. Specify one of the security providers listed in your java.security file that has a key store implementation. The default value is IBMJCE.

  • com.ibm.ssl.keyManager

    The name of the key management algorithm to use. Specify any key management algorithm that is implemented by one of the security providers listed in your java.security file. The default value is IbmX509.

- com.ibm.ssl.trustStoreProvider

  The name of the trust store provider to use. Specify one of the security providers listed in your java.security file that has a trust store implementation. The default value is IBMJCE.

- com.ibm.ssl.trustManager

  The name of the trust management algorithm to use. Specify any trust management algorithm that is implemented by one of the security providers listed in your java.security file. The default value is IbmX509.

- com.ibm.ssl.trustStoreType

  The type or format of the trust store. The possible values are JKS, PKCS12, JCEK. The default value is JKS.

- com.ibm.ssl.enabledCipherSuites

  The list of space-separated cipher suites to enable. By default, this is not set and the set of cipher suites used are determined by the value of the SecurityLevel (High, Medium, or Low) on the General tab, seen in Figure 21-110 on page 832. A cipher suite is a combination of cryptographic algorithms used for an SSL connection. Click the **Help** button to see the list of available cipher suites.

Part 5

Managing WebSphere

In this part we cover other administrative tasks, including using administrative tools, monitoring and tuning your environment, command-line administration, troubleshooting, and migration.

22

Monitoring and tuning your runtime environment

In this chapter we first describe the new Performance Monitoring Infrastructure (PMI) that comes with IBM WebSphere Application Server V4.0. We also describe the monitoring tools that use PMI:

- WebSphere Resource Analyzer
- Performance monitoring servlet

A short introduction to using the Java Virtual Machine Profiling Interface (JVMPI) and monitoring the IBM HTTP Server follows.

The remainder of the chapter provides steps and explanations of how to tune your WebSphere Application Server runtime environment. We describe the Performance Tuner wizard that can be accessed from the WebSphere Administrative Console.

22.1 What is Performance Monitoring Infrastructure?

The *Performance Monitoring Infrastructure* (PMI) is a set of packages and libraries designed to assist with gathering, delivering, processing and displaying performance data in WebSphere Application Server V4.0 domains.

PMI uses a client/server architecture. In PMI terms, a server is any application that uses the PMI API to collect performance data. Servers can include application servers, Web servers, and Java applications. IBM WebSphere Application Server V4.0 provides a service named *PmiService* that is responsible for retrieving performance data from other servers in the domain and making the data available to interested clients. A client is an application that receives performance data from a server or servers and processes the data. Clients can include graphical user interfaces (GUIs) that display performance data in real time, applications that monitor performance data and trigger different events according to the current values of the data, or any other application that needs to receive and process performance data. The role of PmiService in collecting and distributing performance data is shown in Figure 22-1.

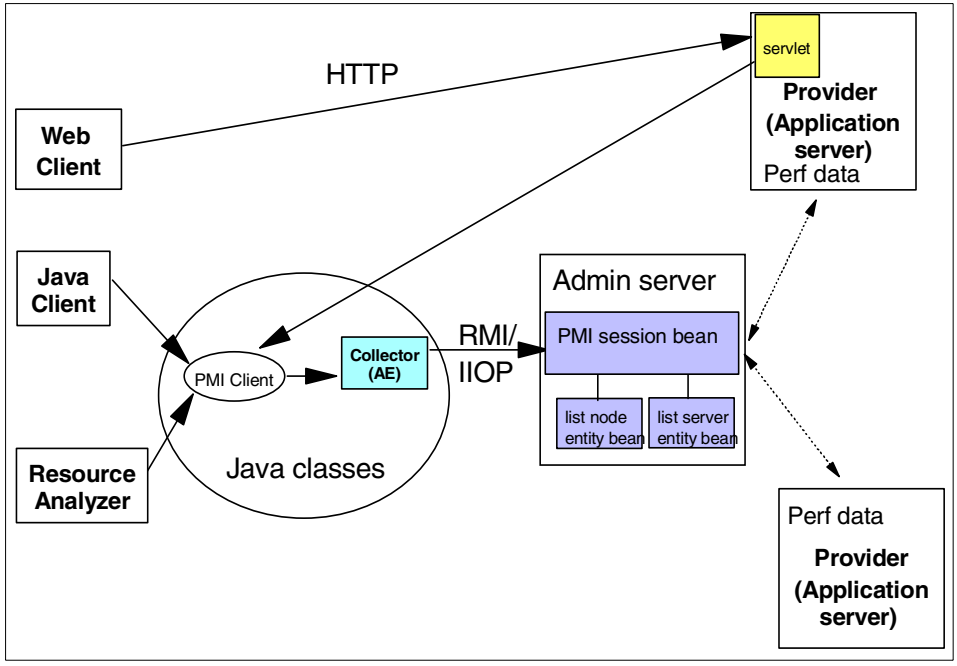


Figure 22-1 Performance Monitoring Infrastructure

Each piece of performance data has two components, a static component and a dynamic component. The static component consists of a name and an ID to identify the data, as well as other descriptive attributes that assist the client in processing and displaying the data. The dynamic component consists of information that changes over time, such as the current value of a counter and the time stamp associated with that value.

In the following sections we give a high-level view on how the PMI and PMI client interface is organized and works together with Resource Analyzer.

22.1.1 Performance data classification

Performance data is classified into the following four types:

- **Numeric** data consists of a single numeric value such as an integer, a long, or a double. It is used to represent data such as counts and sizes.
- **Statistical** data on a sample space consists of the number of elements in the sample set, the sum of the elements, and the sum of squares. These values can be used to obtain the mean, the variance and the standard deviation of the mean. An example of statistical data is the response time for each invocation of an enterprise bean.
- **Load** data monitors a value as a function of time. Example uses include tracking the number of threads or the number of service requests in a queue. Load data tracks the current value, the time the value was reached and the integral over time of the value. These values can be used to obtain the weighted average for the level over a period of time. An example of load data is the average size of a database connection pool during a specified time interval.
- **Group** data is a collection of performance data intended to be used by groups. It enables servers to create sets of performance data that can be retrieved by clients with a single call.

22.1.2 Performance data hierarchy

Performance data is provided in a centralized hierarchy of the following objects:

- **Node**. A node represents a physical machine in the WebSphere administrative domain. No performance data is collected for the node itself.
- **Server**. A server is a functional unit that provides services to the clients over a network. No performance data is collected for the server itself.
- **Module**. A module represents a resource category for which performance data is collected. For Resource Analyzer these are Enterprise Java Beans, database connection pools, J2C connectors, JVM runtime, JVMPI runtime,

servlet session manager, thread pools, transaction manager, and Web applications.

- **Method**. A method is software that implements a specified operation within a Java class. For the Resource Analyzer, information is collected on instance methods.
- **Servlet**. A servlet is a Java program that extends the functionality of the Web server. It generates dynamic data in response to client requests. For the Resource Analyzer, information is collected on the servlets in a Web application.
- **Counter**. A counter is a data type used to hold performance information for analysis.

Each resource category (module) has a related set of counters. The counters contain values for performance data that can have an impact on system performance. Counters measure some aspect of the running system, for example the number of active enterprise beans, the time spent in responding to a servlet request, or the number of kilobytes of available memory.

Modules can have instances, which are single instantiations of an object of a class. Counters can be assigned to any number of modules and instances. Figure 22-2 shows the counter *Avg Method RT* assigned to both the enterprise beans module, and the methods of the Container1.Bean1 instance. Figure 22-2 also shows a hierarchy of data collections that are organized for reporting to the Resource Analyzer.

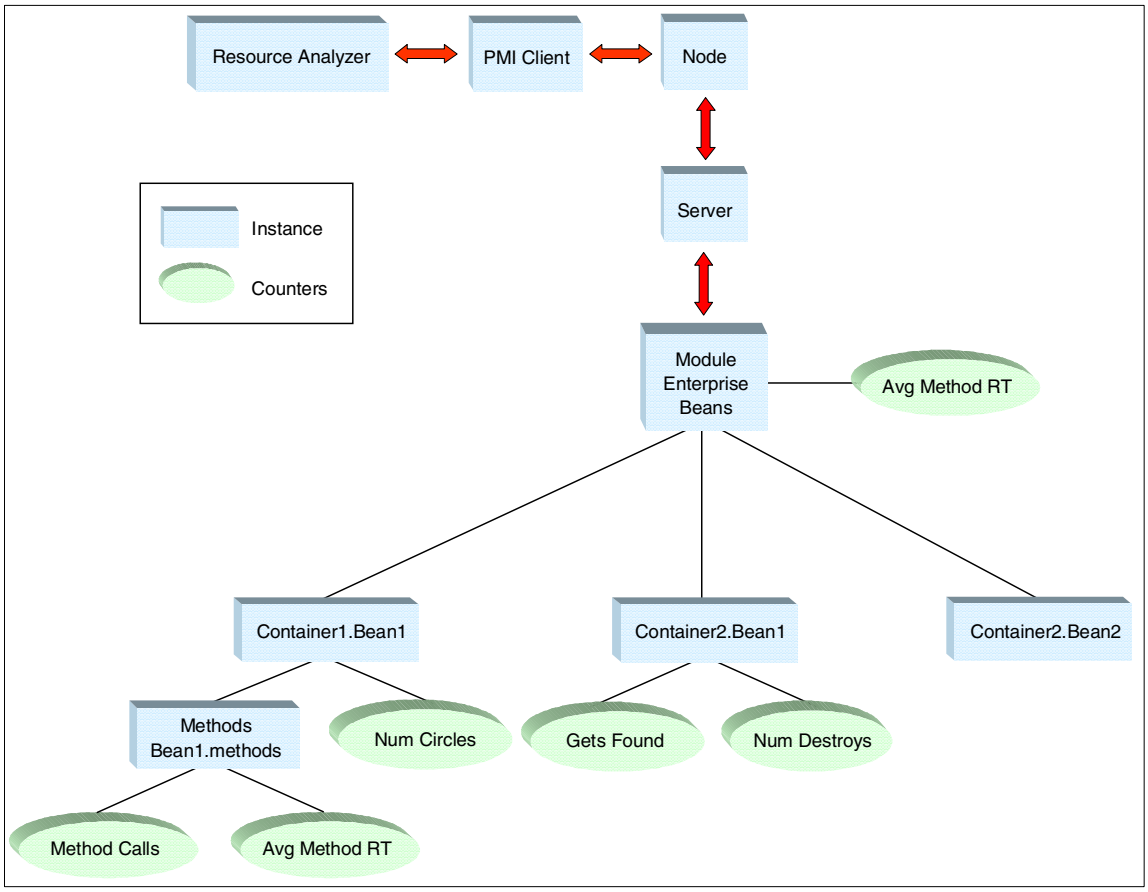


Figure 22-2 Example performance group hierarchy

You can activate or deactivate counters for each module (for example, Enterprise Beans category). Each module has its own set of counters. A subset of counters is available based on the instrumentation level chosen for the particular module or instance. See 22.2.5, “Understanding instrumentation levels” on page 850.

22.1.3 Performance data organization

PMI data is provided to clients in a hierarchical structure and organized into modules (resource categories) based on runtime components. Each module has a configuration file in XML format that determines its organization. It specifies unique identifiers for each performance data item per module. A client can use this unique identifier to fetch the performance data’s static and dynamic information.

Performance data is collected for each of the following modules (resource categories):

- Enterprise beans

  Reports load values, response times, and life cycle activities for EJBs. Examples include the average number of active beans and the number of times bean data is loaded or written to the database. It also reports information about the size and the usage of a cache of bean objects (EJB object pool).

- Database connection pools

  Reports usage information about connection pools for a database. Examples are the average size of the connection pool, the average number of threads waiting for a connection, the average waiting time in milliseconds for a connection and the average time a connection was in use.

- J2C connectors

  Reports usage information about the J2EE Connector Architecture that enables EJBs to connect and interact with procedural back-end systems such as CICS (Customer Information Control Systems) and IMS (Information Management System). Examples are the number of managed connections (physical connections) and the total number of connections (connection handles).

- JVM runtime

  Reports memory used by a process as reported by the JVM. Examples are the total memory available and the amount of free memory for the JVM.

- JVMPI runtime

  In addition, the Resource Analyzer makes use of a Java Virtual Machine Profiling Interface (JVMPI) to enable a more comprehensive performance analysis. This profiling tool enables the collection of information about the JVM that runs the application server. See 22.4.2, “Enabling JVMPI from administrative console” on page 868.

- Servlet session manager

  Reports usage information for HTTP sessions. Examples include the total number of sessions being accessed, the average session life time in milliseconds, and the number of sessions being invalidated.

- Thread pools

  Reports information about the pool of ORB (Object Request Broker) threads that an application server uses to process remote methods and the Web container pools that are used to process HTTP requests coming into the application server. Examples include the average pool size, the number of threads created and destroyed and the number of concurrently active threads.

- Transaction manager

  Reports transaction information for the container. Examples include the average number of concurrently active transactions (local and global), the average duration of transactions and the number of transactions committed, rolled back and timed out.

- Web applications

  Reports load information for the selected Web application and the installed servlets. Examples are the number of loaded servlets, the number of servlet reloads, the total requests that a servlet has processed and the response time in milliseconds for servlet requests.

Applications that use the PMI facility are:

- WebSphere Resource Analyzer, discussed in 22.2, “Using WebSphere Resource Analyzer” on page 846.
- Performance Servlet, discussed in 22.3, “Using performance monitoring servlet” on page 863.

22.1.4 PMI client package

The Performance Monitoring Infrastructure client package is part of the PMI application programming interface (API). This can be used to develop your own performance monitoring client.

If you are interested in developing your own PMI clients, see the WebSphere InfoCenter. The “Performance monitoring infrastructure client package” article discusses the PMI client package and describes how to use it with WebSphere Application Server.

22.2 Using WebSphere Resource Analyzer

The WebSphere Resource Analyzer is a stand-alone runtime performance monitor for WebSphere Application Server V4.0, Advanced Edition. It provides a Graphical User Interface (GUI) console that is available on Windows and UNIX platforms. Resource Analyzer can also be used remotely and across platforms. Another ability of this tool is to record the collected information and replay it without connecting to WebSphere Application Server.

22.2.1 About Resource Analyzer

The Resource Analyzer retrieves performance data by periodically polling the PmiService inside the WebSphere Application Server. The performance data requested by Resource Analyzer is provided by the PmiService (session bean inside administrative server). PmiService fetches the performance data directly from the application servers without storing it.

You can regulate the impact incurred from data collection by using the Resource Analyzer or the WebSphere Administrative Console (see 22.2.5, “Understanding instrumentation levels” on page 850). The Resource Analyzer’s GUI provides controls that enable you to choose the particular resources and counters to include in the view. There are table and chart views available. You can also store retrieved data in a log file while viewing the data. This log file can later be used for replaying the scenario. See Figure 22-3 for an infrastructure overview.

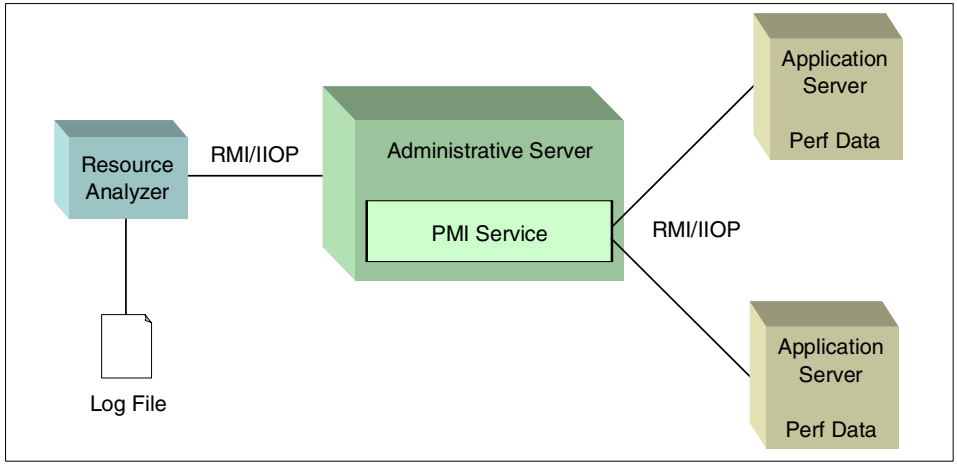


Figure 22-3 Resource Analyzer infrastructure overview

22.2.2 What can Resource Analyzer do?

The Resource Analyzer provides access to a wide range of performance data for two kinds of resources:

- Application resources (for example, enterprise beans and servlets).
- WebSphere runtime resources (for example, Java Virtual Machine (JVM) memory, application server thread pools, and database connection pools).

Performance data includes simple counters, statistical data (such as the response time for each method invocation of an enterprise bean), and load data (such as the average size of a database connection pool during a specified time interval). This data is reported for individual resources and aggregated for multiple resources. See 22.1, “What is Performance Monitoring Infrastructure?” on page 840 for more details about performance data organization.

Resource Analyzer functionality

Depending on which aspects of performance are being measured, you can use the Resource Analyzer to accomplish the following tasks:

- View data in real time or view historical data from log files.
- View data in chart form, allowing comparisons of one or more statistical values for a given resource on the same chart. In addition, different units of measurement can be scaled to enable meaningful graphic displays.
- Record current performance data in a log and replay performance data from previous sessions.
- Compare data for a single resource to an aggregate (group) of resources on a single node.

Given all this data, the Resource Analyzer can be used to do the following types of analysis:

- Monitor real-time performance, such as response times for servlet requests or enterprise bean methods.
- Detect trends by analyzing logs of data over time.
- Determine the efficiency of a configuration of resources (such as the amount of allocated memory, the size of database connection pools, and the size of a cache for enterprise bean objects).
- Gauge the load on application servers and the average wait time for clients.

22.2.3 About performance data counters

Each resource category (module) has his own set of performance data counters. Those counters have particular properties, for example the rating impact and data type. A complete list of all performance data counters for each resource category is included in the InfoCenter article, “Performance data reported with the Resource Analyzer“. There is a separate table for each resource category. As an example, the JVM runtime counters are shown in “Counters for the JVM runtime” on page 849.

Each set of counters are described in tables using the columns listed in Table 22-1.

Table 22-1 Description of the performance data counters

Column name	Description
Name	This column shows the name of the counter as it appears in the Counter Selection window.
Granularity	This column indicates the unit to which data collection is applied for that counter.
Impact	This column identifies the level of performance impact (low, medium, or high) that is expected when the counter is enabled.
Data type	This column shows the digital representation format used for expressing the measurable units that are collected by the counter. The following types are included: Long This is a signed integer. Stat This is basic statistical data that is used to calculate means, variations, and standard deviation information. The Resource Analyzer uses the following raw data, which is stored on the application server: the sum, the count, and the sum of squares (for standard deviation calculations). Load This value adds a time dimension to calculate time-weighted statistics.
Bean type (only for enterprise beans)	For the enterprise beans resource category, Bean type is also given. Values can be entity, stateful session, stateless session, and all.

To perform calculations for time-weighted statistics, the following raw data, which is stored on the application server, is used:

- The time-weighted sum, which is updated each time the counter is changed.
- The time when the counter is created.
- The time when the counter was last updated.
- The current value of the counter.

Counters for the JVM runtime

The performance data for the Java Virtual Machine (JVM) runtime is listed in Table 22-2.

Table 22-2 Counter information for JVM runtime memory

Name	Granularity	Impact	Data type	Description
Total memory (bytes)	per JVM	Low	long	The amount of used memory in JVM runtime.
Free memory (bytes)	per JVM	Low	long	The amount of free memory in JVM runtime.
Used memory (bytes)	per JVM	Low	long	The total amount of memory in JVM runtime.

22.2.4 Understanding data counter impact rating

Collecting data from WebSphere and runtime resources always affects the performance in some way and the impact itself varies depending on the counter. Resource Analyzer represents the overhead cost associated with each counter as a rating of *low*, *medium*, or *high*. If a counter has a low cost rating, its performance cost is minor and usually involves a single addition or subtraction. A counter with a high cost rating has a higher performance impact. A high cost rating usually indicates that several calculations, including multiplication, division, or both, are involved in gathering the data for the counter.

For example, collecting the number of times a bean instance is passivated has a low impact on performance. Calculating the average response time on all methods of an enterprise bean's remote interface has a medium impact on performance. Calculating the average number of methods being processed concurrently has a high impact. The associated cost rating (low, medium, or high) for each counter is described in 22.2.3, “About performance data counters” on page 848.

22.2.5 Understanding instrumentation levels

The resources in a WebSphere administrative domain are instrumented so that statistical data can be collected. Instrumentation refers to the mechanism by which some aspect of the running system is measured (analogous to a meter attached to a resource). Each resource category has an instrumentation level (different from the impact rating for the counters in that category). The instrumentation level determines which counters are available to be collected for that category.

For example, if a resource category has an instrumentation level setting of low, only counters having a low impact rating are available for selection. If the instrumentation level is set to medium, then counters having low impact and medium impact ratings are available for selection. Similarly, when the instrumentation level is set to high, all counters with low, medium, and high impact ratings are available for selection.

An instrumentation level also can be set to *maximum*, which enables the availability of all counters and, in addition, increases the level of granularity when reporting on enterprise methods. This setting has a higher impact on a system's performance. Conversely, the instrumentation level can be set to none, which disables performance reporting and eliminates any impact of monitoring on system performance. Initially, the instrumentation levels are set to none.

The instrumentation level can be set by using:

- Resource Analyzer's main menu, clicking **Actions** -> **Monitoring Settings...**.
- WebSphere Administrative Console on the application server properties pane.

See "Setting the instrumentation level" on page 853 for a detailed description.

22.2.6 Using Resource Analyzer to monitor an application

In this section we explain the steps we used to monitor one of the sample enterprise applications installed into the WebSphere administrative domain by default.

Starting Resource Analyzer

To start the Resource Analyzer on any platform:

- Select **Tools** -> **Resource Analyzer** from the WebSphere Administrative Console main menu, as shown in Figure 22-4.

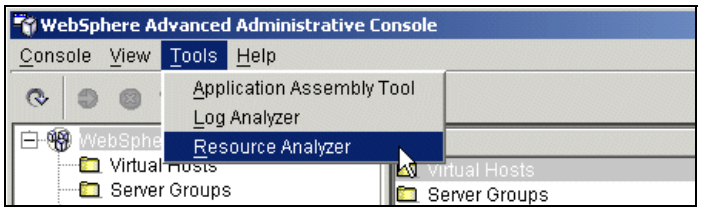


Figure 22-4 Start Resource Analyzer from WebSphere Administrative Console

- Using the start script on the command line.

On Windows platforms do one of the following:

- Click **Start** -> **Programs** -> **IBM WebSphere** -> **Application Server 4.0** -> **Resource Analyzer.**
- Use the command line by typing:

```
<WAS_HOME>\bin\ra.bat
```

On UNIX platforms do the following:

- Type into a shell:

```
<WAS_HOME>/bin/ra.sh
```

By default Resource Analyzer tries to access the administrative server on the local machine (localhost and port 900). However, if your are already connected to a remote administrative server via WebSphere Administrative Console, then Resource Analyzer connects to this machine. If no administrative server is running, you can use Resource Analyzer to replay a previously recorded session by selecting **File** -> **Open Log File...** from the main menu (see “Record a load scenario” on page 859). If you want to access a remote machine running WebSphere Application Server, the command usage is:

```
ra.bat|sh [host_name [port_number]]
```

Where:

`host_name`	Host name or IP address of the remote system (default: localhost).
`port_number`	Port number of the remote system (default: 900).

For example, if you want to access a remote UNIX machine running WebSphere Application Server from your Windows system, type in the following:

```
ra.bat mywas.mydomain.com 900
```

If the default port 900 is used, it can be left out.

After starting the Resource Analyzer successfully, you will get the main window shown in Figure 22-5. Resource Analyzer window consists of the menu bar, below the task bar, on the left-hand side the Resource Selection pane, on the right-hand side the data monitoring pane (can contain table and charts alternatively), and on the bottom you will find the status bar.

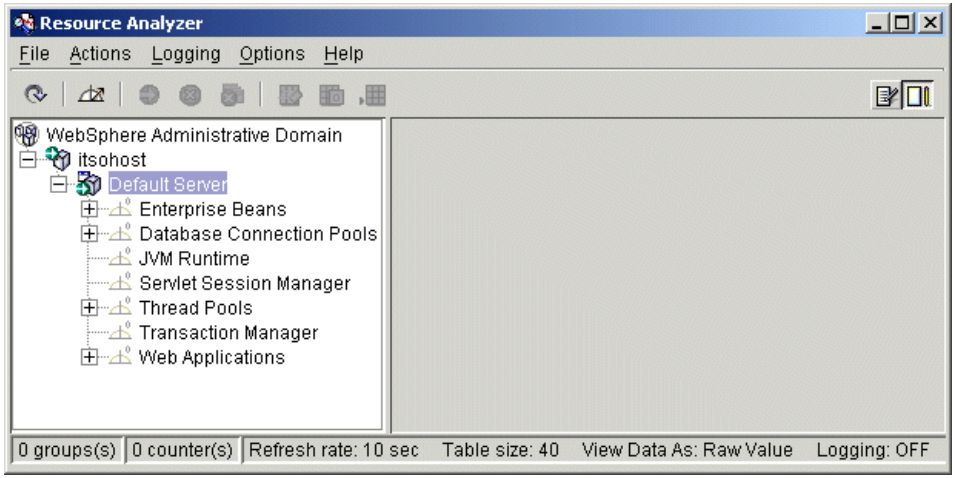


Figure 22-5 Resource Analyzer

Stop the Resource Analyzer by selecting **File** -> **Exit** from the main menu. This will bring up a window if the performance monitoring feature was not enabled before starting the Resource Analyzer. The window asks whether you want to save the monitoring settings in the administrative server or not. Choosing **Yes** will save the performance monitoring settings within the administrative server; choosing **No** will discard them.

Tip: Due to memory requirements, it is recommended that you run the Resource Analyzer on a separate machine from the administrative server

What happens if security is enabled?

If your WebSphere Application Server is configured for global security then you will get an user ID/password challenge window, as shown in Figure 22-6, before the main window. At this point do the following steps:

1. Enter a valid user ID and password.
2. Click **OK** to log on.

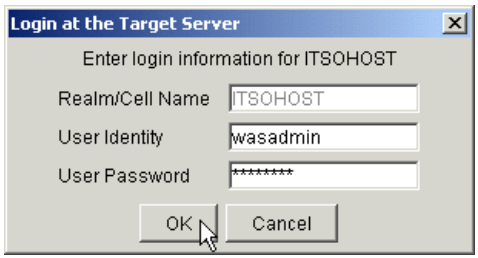


Figure 22-6 Resource Analyzer login challenge

Setting the instrumentation level

To specify the instrumentation level for resource categories, use the Performance Monitoring Settings window. Access the Performance Monitoring window either from:

- WebSphere Administrative Console
- Resource Analyzer

You can also use the **wscp** command to set the instrumentation level, as described in “Using the wscp command to set instrumentation level” on page 854.

Using administrative console to set instrumentation level

To open the Performance Monitoring Settings window using the WebSphere Administrative Console:

1. Highlight your application server (for example Default Server).
2. Choose the **Services** tab in the property pane.
3. Highlight the **Performance Monitoring Settings** in the service listing.
4. Click the **Edit Properties** button, as shown in Figure 22-7.

The Performance Monitoring Settings window pops up. To set the instrumentation level, see “Performance Monitoring Settings window” on page 855.

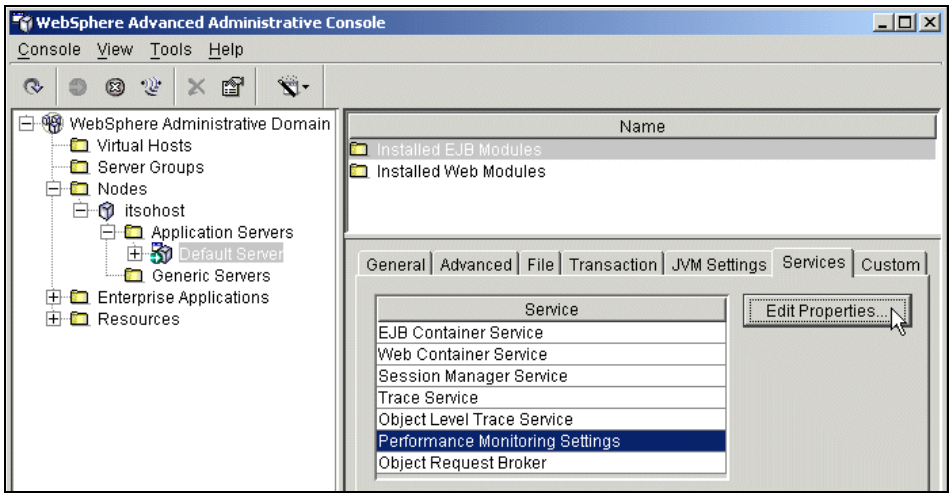


Figure 22-7 Setting instrumentation level from WebSphere Administrative Console

Using Resource Analyzer to set instrumentation level

To open the Performance Monitoring Settings window using the Resource Analyzer:

- Highlight the appropriate application server in the Resource Selection pane. Do one of the following:
  - Right-click the application server and select **Monitoring Settings...** from the pop-up menu.
  - Click the **Monitoring Settings...** button in the task bar.
  - Select **Actions** -> **Monitoring Settings...** from the main menu.

Also, if you click a resource with an instrumentation level of none, you will be prompted to set the monitoring level now. Click **Yes** to open the Performance Monitoring Settings window.

Using the wscp command to set instrumentation level

To enable performance data monitoring, use the following **wscp** command:

```
wscp> PmiService enableData \
   {/Node:itsohost/ApplicationServer:Default Server} \
   -dd {jvmRuntimeModule}
```

To disable the performance data collection, use the following **wscp** command:

```
wscp> PmiService disableData \
   {/Node:itsohost/ApplicationServer:Default Server} \
   -dd {jvmRuntimeModule}
```

To set the instrumentation level for JVM runtime to low (3 means low), use the following **wscp** command:

```
wscp> PmiService setLevel \
   {/Node:itsohost/ApplicationServer:Default Server} \
   -levelspec {{JVMRuntimeModule 3}}
```

To set the instrumentation level for JVM runtime to none, use the following **wscp** command:

```
wscp> PmiService setLevel \
   {/Node:itsohost/ApplicationServer:Default Server} \
   -levelspec {{pmi/JVMRuntimeModule 0}}
```

Performance Monitoring Settings window

The Performance Monitoring Settings windows from both tools have the same functionality. However, if you come from the WebSphere Administrative Console you have to check **Enable performance counter monitoring**, as shown in Figure 22-8.

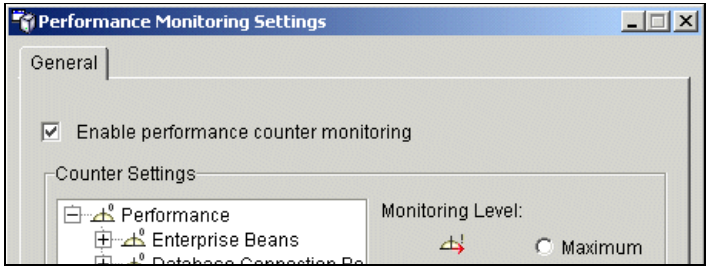


Figure 22-8 Enable performance counter monitoring

To set the instrumentation level for a resource category:

1. Locate the desired resource category in the hierarchy.
2. Highlight the desired resource category and select the required monitoring level on the right (either **Maximum**, **High**, **Medium**, **Low**, or **None**). Figure 22-9, for example, shows JVM Runtime set to Low.
3. Click **OK** to apply the settings and close the window.

Changing or setting the instrumentation level always starts the performance data reporting immediately for the selected resource category.

The counters available for the selected resource category will be shown on the lower left (in the Counters box) when you choose the monitoring level. Figure 22-9 shows the counters available for JVM Runtime with the instrumentation level set to **Low**.

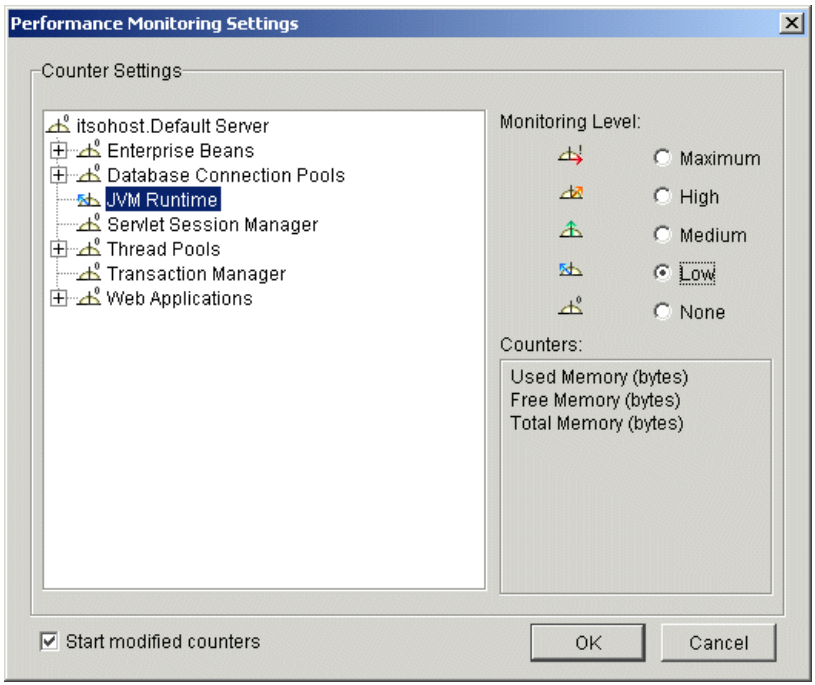


Figure 22-9 Performance Monitoring Settings for JVM Runtime

Setting a particular instrumentation level for a resource category applies the same level all subcategories. You can specify a different level for an individual resource subcategory by selecting it and specifying the desired level.

Expand the resource categories to make sure that only the desired level is applied; otherwise the performance impact may be higher than you expected.

Note: Instrumentation level settings are permanent even after restarting the application server or the administrative server. You have to disable them explicitly.

Setting instrumentation level for enterprise bean methods

To avoid the overhead of monitoring individual remote methods, individual methods in enterprise beans will not be displayed in the performance pane unless the methods monitoring level is set to maximum. To display individual methods and specify their instrumentation levels, do the following:

1. Open the Performance Monitoring Settings window.
2. Set the instrumentation level for the methods category to Maximum.

3. Click **OK** to close, then re-open the Performance Monitoring Settings window. Individual methods are now displayed as shown in Figure 22-10.
4. Select individual methods, as required.

Note that only methods that have been called by an application are displayed. If a remote method has not been called since the application server was started, it does not appear in the performance pane.

In Figure 22-10, for an example, the IncBean enterprise bean has been called at least once, but the BeenThere bean has not been called yet.

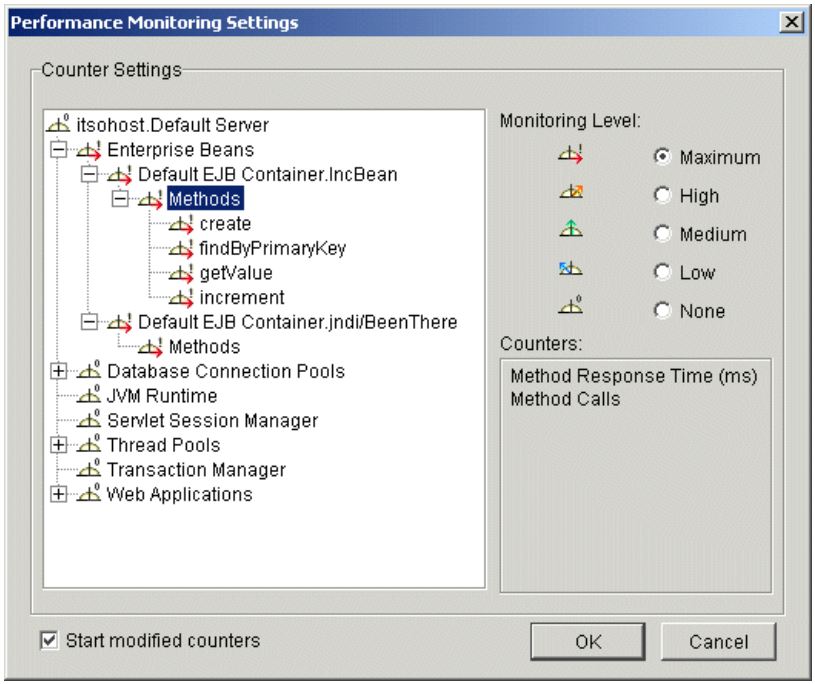


Figure 22-10 Enterprise bean methods available after the first method call

Starting data reporting

Resource Analyzer starts reporting immediately after changing the instrumentation level for resource categories, or by using the following steps:

1. Select one or more resource categories in the resource hierarchy. Multiple selections can be made when holding the Shift or Ctrl key.
2. Select **Actions** -> **Start** from the main menu, or click the **Start** icon on the task bar. You can also right-click a single resource category and select **Start** from the pop-up menu.

When new modules or instances (JSPs, servlets, enterprise beans) are loaded by WebSphere Application Server while Resource Analyzer is running, they are not automatically displayed and included in performance data reporting. Use the following steps to include newly loaded modules or instances:

1. Select the resource category that the module or instance belongs to.
2. Select **Actions** -> **Refresh** from the main menu, or click the **Refresh** icon on the task bar.
3. Set instrumentation level for this new module or instance as described in “Setting the instrumentation level” on page 853.

Tip: For load test scenarios it is wise to set up so-called preload tests. Preload tests make sure that all of your resources are loaded at least once. All caches inside WebSphere Application Server will be initialized. Another advantage is that you can configure Resource Analyzer in one step because all instances are available.

Stopping data reporting

Use the following steps to stop data reporting:

4. Select one or more resources in the resource hierarchy.
5. Select **Actions** -> **Stop** from the main menu. Alternatively, right-click a resource and select **Stop** from the pop-up menu. The resource's icon turns red.

If you want to stop data reporting for all resources, select **Actions** -> **Stop All** from the main menu.

When a running application server is stopped in the administrative console, data reporting for the server also stops, and its resource icons turn red.

Showing correlated counters

To analyze the performance data captured by Resource Analyzer it is necessary to set counters from different modules in correlation. Combinations of multiple counters from different sources is supported. To select multiple counters in one chart or table:

1. Hold down the Ctrl key and select the resource categories from which you want to see the counters.
2. Select the required counters using the Select column in the counter selection window on the lower left.

If you want to see if there is a correlation between *Created Sessions* (from Servlet Session Manager), *Num Allocates* (from Database Connection Pools), and *Average Response Time* (from Web Applications), then set up the Resource Analyzer as shown in Figure 22-11.

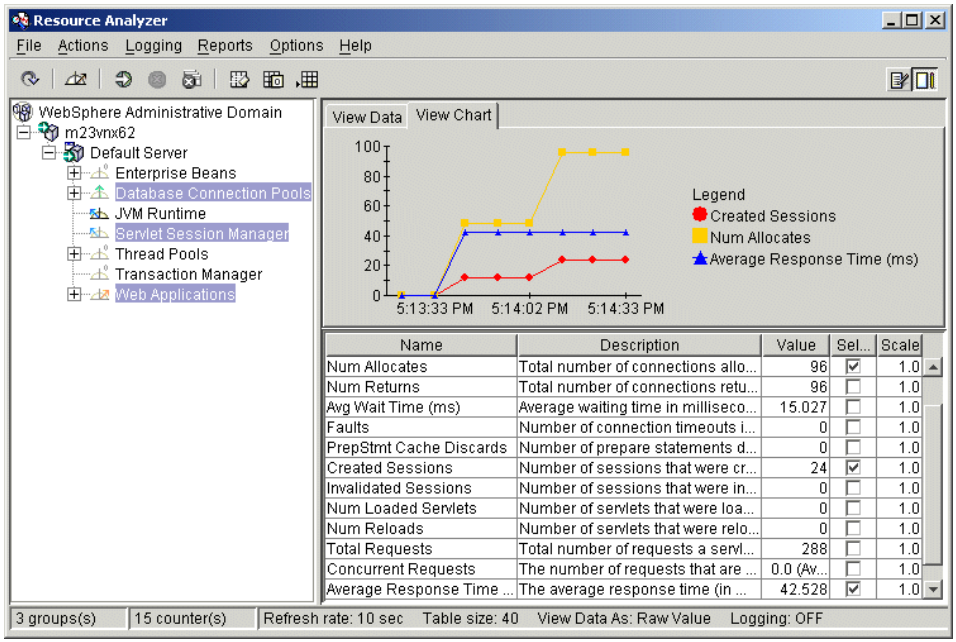


Figure 22-11 Selecting multiple counters from different resource categories

Record a load scenario

All data being reported by the Resource Analyzer can be saved in a log file. The data is written to the log as serialized Java objects or as an XML document. To start recording data, do the following:

1. Select **Logging** -> **On...** from the Resource Analyzer main menu.
2. In the Save log file window, specify the File name (and path), and Save as type. The Save as type field allows an extension of *.lra (Resource Analyzer logs) for binary files or *.xml for XML files.
3. Click **OK**.

To stop the recording of data, select **Logging** -> **Off** from the main menu.

Replaying a load scenario

Only log files saved in binary format (*.lra) can be replayed using Resource Analyzer. Log files saved in XML format (*.xml) provide the option of analysis using third-party tools. However XML files cannot be replayed using Resource Analyzer. To replay a log file, do the following:

1. Select **File** -> **Open Log File...** from the Resource Analyzer main menu.
1. In the Open window, locate the name of the file to replay and click the **Open** button.

By default, the data is replayed at the same rate it is collected (written to the log). If data is collected every minute, it is displayed every minute. You can change the speed at which the log is replayed by clicking **Options** -> **Play Speed**. If the data is collected every minute and the speed factor is set to 60x, then data is displayed every second.

While replaying the log, you can change your resource selections with the resource selection pane. You can also view the data in either of the views available (data or chart) in the data display window.

You can stop and resume the log at any point. However, data cannot be replayed in reverse.

To rewind the log file, select **Actions** -> **Rewind** from the main menu. Alternatively, you can use the Rewind icon on the toolbar.

Getting current data

The following must be done to obtain the current values for all counters that belong to a resource:

1. Select one or more resources in the resource hierarchy.
2. Select **Actions** -> **Get Value** from the main menu. Alternatively, right-click the resource and select **Get Value** from the pop-up menu.

The values are displayed independently of the refresh rate. The resource does not need to be running in order to get the values of the counters.

Clearing values from tables and charts

After stopping a resource, use the Clear Values operation to remove the remaining data from a table or chart. You can then begin populating the table or chart with new data. To clear the values currently displayed, do the following:

1. Select one or more resources in the resource hierarchy.
2. Select **Actions** -> **Clear Values** from the main menu. Alternatively, right-click the resource and select **Clear Values** from the pop-up menu.

Resetting counters to zero

To reset the start time for calculating aggregate data, do the following:

1. Select one or more resources in the resource hierarchy.
2. Select **Actions** -> **Reset to Zero** from the main menu. Alternatively, right-click the resource and select **Reset to Zero** from the pop-up menu.

The reset operation sets the clock used for reporting aggregate data for counters of the selected performance category. Instead of reporting data from the time the server was started, reporting now begins from the time of the reset action. Not all counters can be reset. If you use the reset operation for a group containing counters that cannot be reset, the reset action has no effect. You can select multiple performance groups and reset them simultaneously.

Viewing and modifying chart data

When selected counters are using measurement units that are not proportionally similar, the scaling factor can be set manually to allow a more meaningful display. The following sections explain how you can manually change the scaling factor for the chart view.

Scaling the chart display manually

To manually change the scale of a counter:

1. In the counter selection window, on the lower right, double-click the **Scale** column for the counter that you want to modify.
2. Type the desired scale value for the counter into the field.

The chart view will be updated immediately to reflect the change in the scaling factor.

The possible values for the Scale field range from 0 to 100 and show the following relationships:

< 1 Scaling reduces the value. For example, a scale of 0.5 means that the data points for the variable are reduced to half of their actual values.

= 1 The value is not scaled. This is the default.

> 1 Scaling increases the value. For example, a scale of 1.5 means that the data points for the variable are increased to one and one-half times their actual values.

Negative results are displayed as zero (0).

This value is reflected only in the View Chart window.

Changing display settings

This section looks at some of the Resource Analyzer display settings:

- Specifying a different refresh rate
- Changing the data view
- Changing the table size

Specifying a different refresh rate

By default, the Resource Analyzer retrieves data from the administrative server every 10 seconds. To change the rate at which data is retrieved from the server, do the following:

1. Select **Options** -> **Set Refresh Rate...** from the main menu.
2. In the Set Refresh Rate window, type a positive integer representing the number of seconds. The integer must be 1 or greater.
3. Click **OK**.

Changing the data view

The data view mode determines whether counter values represent absolute values, changes in values, or rates of change. The data view mode meanings differ slightly depending on where you are viewing data. To change the data view mode:

1. Select **Options** -> **View Data As** from the main menu.
2. Select from the following choices:
   - **Raw Value**. Displays the absolute values. If the counter represents load data (for example, the average number of connections in a database pool or the average number of live bean objects), the Resource Analyzer displays the current value followed by the average, for example, 18(avg:5).
   - **Change in Value**. Displays the change in the current value from a previous value.
   - **Rate of Change**. Displays the ratio change/(T1 - T2), where change is the change in the current value from a previous value, T1 is the time when the current value was retrieved and T2 is the time when the previous value was retrieved.

Changing the table size

By default, the View Data window displays 40 rows, corresponding to the values of the last 40 data points retrieved from the administrative server. To change the size of the table (number of rows displayed):

1. Select **Options** -> **Set Table Size...** from the main menu.

2. In the Set Table Size window, specify the number of rows to display.
3. Click **OK**.

22.2.7 Getting online help

WebSphere Resource Analyzer has HTML-based online help. To see the help pages select **Help** -> **Help Topics** from the main menu. There are also a number of Resource Analyzer articles in the WebSphere InfoCenter, available at:

`http://www.ibm.com/software/webservers/appserv/infocenter.html`

22.3 Using performance monitoring servlet

The performance servlet provides a way to use an HTTP request to query the performance metrics for an entire WebSphere Application Server administrative domain.

The performance servlet is used for simple end-to-end retrieval of performance data that can be consumed by any tool, provided by either IBM or a third-party vendor. Because the servlet provides the performance data via HTTP, it can be used with firewalls.

The performance servlet uses the Performance Monitor Interface (PMI) infrastructure to retrieve the performance information from WebSphere Application Server. PMI is used by the Resource Analyzer as well. Therefore, it is subject to the same restrictions on the availability of data as the Resource Analyzer.

The performance monitoring servlet is bundled as an EAR file that can be found at <WAS_HOME>/installableApps/perfServletApp.ear. In order to use it, simply install that EAR file into an existing application server. After starting the enterprise application containing the performance monitoring servlet, you can retrieve performance data from the entire domain.

22.3.1 How the performance servlet provides data

The performance servlet provides the performance data as an XML document, as described in the provided DTD. In the XML structure, the leaves of the structure provide the actual observations of performance data, and the paths to the leaves provide the context. There are three types of leaves within the XML structure:

- PerfNumericInfo
- PerfStatInfo

- PerfLoadInfo

Note: All three types provide values that are relative to the time when the server was started.

PerfNumericInfo type

This type represents raw counters. Raw counters record the number of times a specific event occurs during the lifetime of the server.

Meaning of attributes:

time	Specifies the time when the observation was collected (Java System.currentTimeMillis)
uid	Specifies the PMI identifier for the observation
val	Specifies the raw counter value

Example 22-1 shows the number of loaded servlets. Note that the path providing the context of the observation is not shown.

Example 22-1 Number of loaded servlets

```
<numLoadedServlets>
   <PerfNumericData time="988162913175" uid="pmi1" val="132"/>
</numLoadedServlets>
```

PerfStatInfo type

This type represents statistical data. Statistical data records the number of occurrences of a specific event (such as the PerfNumericInfo type), in addition to the squares of each observation.

Meaning of the attributes:

time	Specifies the time when the observation was collected (Java System.currentTimeMillis)
uid	Specifies the PMI identifier for this observation
num	Specifies the number of observations
sum_of_squares	Specifies the sum of the squares of the observations
total	Specifies the sum of the observations
mean	Specifies the mean (total/num) for this counter

Example 22-2 shows the response time of an object. Note that the path providing the context of the observation is not shown.

Example 22-2 Response time of an object

```
<responseTime>
   <PerfStatInfo mean="1211.5" num="5" sum_of_squares="3256265.0"
      time="9917644193057" total="2423.0" uid="pmi13"/>
</responseTime>
```

PerfLoadInfo type

When each invocation of the performance servlet retrieves the performance values from PMI, some of the values are stored as a load. Loads record values as a function of time.

Meaning of attributes:

time	Specifies the time when the observation was collected (Java System.currentTimeMillis)
uid	Specifies the PMI identifier for this observation
currentValue	Specifies the current value for this counter
integral	Specifies the time-weighted sum
timeSinceCreate	Specifies the elapsed time in milliseconds since this data was created in server
mean	Specifies time-weighted mean (integral/timeSinceCreate) for this counter

Example 22-3 shows the number of concurrent requests. Note that the path providing the context of the observation is not shown.

Example 22-3 Number of concurrent requests

```
<poolSize>
   <PerfLoadInfo currentValue="1.0" integral="534899.0"
      mean="0.9985028962051592" time="991764193057"
      timeSinceCreate="535701.0" uid="pmi5"/>
</poolSize>
```

22.3.2 Monitoring an application with the performance servlet

The first request initializes the performance servlet. During the initialization it retrieves the list of nodes and servers located within the domain in which it is deployed. Because the collection of this data is expensive, the performance servlet holds this as a cached list for performance reasons.

To show the performance data as XML do the following:

1. Make sure the Default Server is running.

2. Make sure performance monitoring is enabled and instrumentation levels are set.
3. Open your browser using the following URL to invoke the servlet. The URL depends on the environment configuration:

```
http://localhost/wasPerfTool/servlet/perfservlet
```

Figure 22-12 shows the XML page generated by the performance monitoring servlet in Microsoft Internet Explorer 5.5, for the Default Server and JVM runtime instrumentation level set to Low.

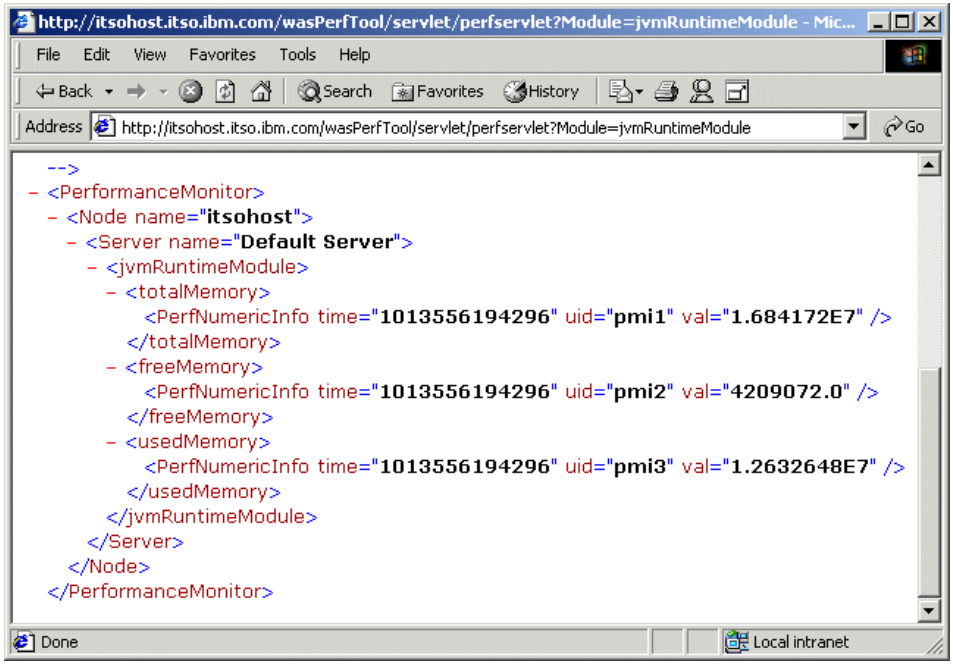


Figure 22-12 XML output from performance monitoring servlet

Refresh the configuration

When a new element, such as a new application server, is added, the servlet needs to be refreshed to show the new element. This can be done by adding the parameter `refreshConfig` to the servlet URL, as follows:

```
http://localhost/wasPerfTool/servlet/perfservlet?refreshConfig=true
```

Limit the requested information

There are additional parameters available to limit the requested performance data. These parameters are listed in Table 22-3.

Table 22-3 Performance monitoring servlet parameter

Parameter	Description
`Node=node_name`	Requests only performance data for the specified node.
`Server=server_name`	Requests only performance data for the specified application server.
`Module=module_name`	Requests only performance data for the specified module type.

The following module names can be used with the Module parameter:

- beanModule
- connectionPoolModule
- jvmpiModule (must be enabled, as described in 22.4, “Using JVMPI facility” on page 868)
- jvmRuntimeModule
- servletSessionsModule
- threadPoolModule
- transactionModule
- webAppModule

When the specified module is not a top-level module, the parent XML wrappers are generated, but only the content of the specified module is generated.

The following URLs show examples for use of these parameters:

```
http://localhost/wasPerfTool/servlet/perfservlet?Node=yourHostName
http://localhost/wasPerfTool/servlet/perfservlet?Server=Default%20Server
http://localhost/wasPerfTool/servlet/perfservlet?Module=beanModule+
jvmRuntimeModule
```

(%20 represents a space in a URL)

In our example in Figure 22-12 on page 866, we used the following URL to request performance data for only the jvmRuntimeModule module:

```
http://hostname/wasPerfTool/servlet/perfservlet?Module=jvmRuntimeModule
```

22.4 Using JVMPI facility

All the performance monitoring tools that use PMI, such as Resource Analyzer and performance monitoring servlet, are able to use the Java Virtual Machine Profiler Interface (JVMPI) to enable a more comprehensive performance analysis. This profiling tool enables the collection of information, such as garbage collection data, about the Java Virtual Machine (JVM) that runs the application server.

JVMPI is a two-way function call interface between the JVM and an in-process profiler agent. The JVM notifies the profiler agent of various events, such as heap allocations and thread starts. The profiler agent can activate or inactivate specific event notifications, based on the needs of the profiler.

The JVMPI facility is available on the Windows, AIX, and Solaris platforms.

22.4.1 Performance data provided by JVMPI

The JVM Profiler Interface provides internal runtime performance data about the following resources:

- Garbage collector
  - Number of garbage collection calls
  - Average time in milliseconds between garbage collection calls
  - Average duration in milliseconds of a garbage collection call
- Monitor
  - Number of times that a thread waits for a lock
  - Average time that a thread waits for a lock
- Object
  - Number of objects allocated
  - Number of objects freed from heap
  - Number of objects moved in heap
- Thread
  - Number of threads started
  - Number of threads died

22.4.2 Enabling JVMPI from administrative console

To enable JVMPI reporting for each individual application server, do the following on the WebSphere Administrative Console:

1. Locate the required application server in the hierarchical tree located in the left pane of the administrative console.
2. Right-click the application server and select **Properties** from the pop-up menu.
3. In the Application Server Properties window, click the **JVM Settings** tab.
4. Click the **Advanced JVM Settings** button on the bottom left of the JVM Settings sheet. The Advanced JVM Settings window is displayed.
5. In the Command line arguments field, type the following case-sensitive argument: -XrunpmiJvmpiProfiler, as shown in Figure 22-13.

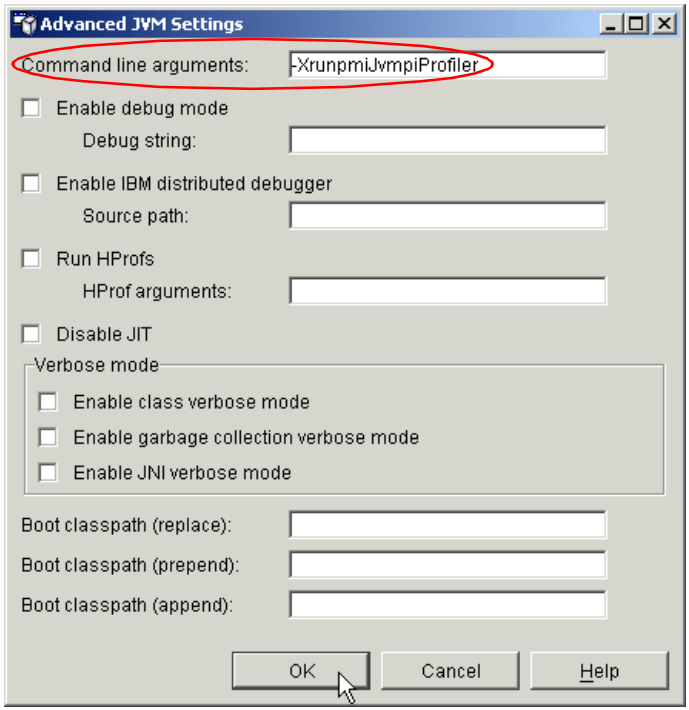


Figure 22-13 Enabling JVMPI

6. Click **OK** to close the Advanced JVM Settings window.
7. Click **OK** to close the Application Server Properties window. When using the console property pane, click **Apply** to apply the changes.
8. Start the application server, or restart the application server if it was already running.
9. Refresh your preferred performance monitoring tool (Resource Analyzer and or Performance Servlet).

22.4.3 Enabling JVMPI with the WebSphere Control Program

To enable JVMPI profiling using the WebSphere Control Program, follow the instructions:

1. Start the WebSphere Control Program.
2. Enter the following command at the prompt:

```
wscp> ApplicationServer modify \
   {/Node:m23vnx62/ApplicationServer:Default Server/} \
      -attribute {{JVMConfig \
         {{AdditionalCommandLineArgs -XrunpmiJvmpiProfiler}}}}
```

3. Start the application server, or restart the application server if it was already running.

Note: WSCP commands are case sensitive.

22.4.4 Disabling JVMPI profiling

From WebSphere Administrative Console

Follow the steps described in 22.4.2, “Enabling JVMPI from administrative console” on page 868 and remove the JVM command line argument -XrunpmiJvmpiProfiler.

From the WebSphere Control Program

Use the **wscp** command shown in 22.4.3, “Enabling JVMPI with the WebSphere Control Program” on page 870, but change the JVMConfig parameter to `{AdditionalCommandLineArgs }`.

22.5 Monitoring IBM HTTP Server

The Windows version of IBM HTTP Server includes Windows NT performance monitor hooks. This allows you to use the Windows NT performance monitor to observe the current state of an active IBM HTTP Server, along with all kinds of other system resources.

22.5.1 Configure your IBM HTTP Server

The default configuration of the IBM HTTP Server provides some performance data to the Windows performance monitoring tools. You can enable more performance data by modifying the HTTP server configuration file (httpd.conf). Add the lines shown in Example 22-4 and restart your IBM HTTP Server.

Example 22-4 Enable extended performance data in httpd.conf

```
LoadModule status_module modules/ApacheModuleStatus.dll
ExtendedStatus On
```

For more information about modifying your HTTP server configuration please refer to the IBM HTTP Server InfoCenter:

http://www.ibm.com/software/webservers/httpservers/library.html

22.5.2 Starting the Windows NT performance monitor

On Windows NT use the following steps to start the performance monitor:

- Click **Start** -> **Programs** -> **Administrative Tools (Common)** -> **Performance Monitor**.

On Windows 2000 use the following steps to start the System Monitor:

1. Click **Start** -> **Settings** -> **Control Panel** to open the Control Panel window.
2. Double-click **Administrative Tools** to open the Administrative Tools window.
3. Double-click **Performance** to open the Performance window.
4. Select the **Tree** tab on the top-left and click **System Monitor** to activate it.

22.5.3 Selecting performance data

To select the IBM HTTP Server as the source for your performance data and specify the counters do the following:

1. Click the plus icon on the taskbar to open the Add Counters window.
2. Select **IBM HTTP Server** from the Performance Object drop-down list.
3. From the Instance field choose one or more process IDs (if there is only a 0 shown, this means no instance of IBM HTTP Server is running).
4. Go to the Counters field and select the required counter. Use the **Explain** button on the right-hand side to get a description of a counter. Select multiple counters by holding down the Shift or Ctrl key.
5. Click **Add** to add your selection to your monitor.
6. Click **Done** or **Close** and the data reporting starts.

Figure 22-14 shows an example of a monitoring session.

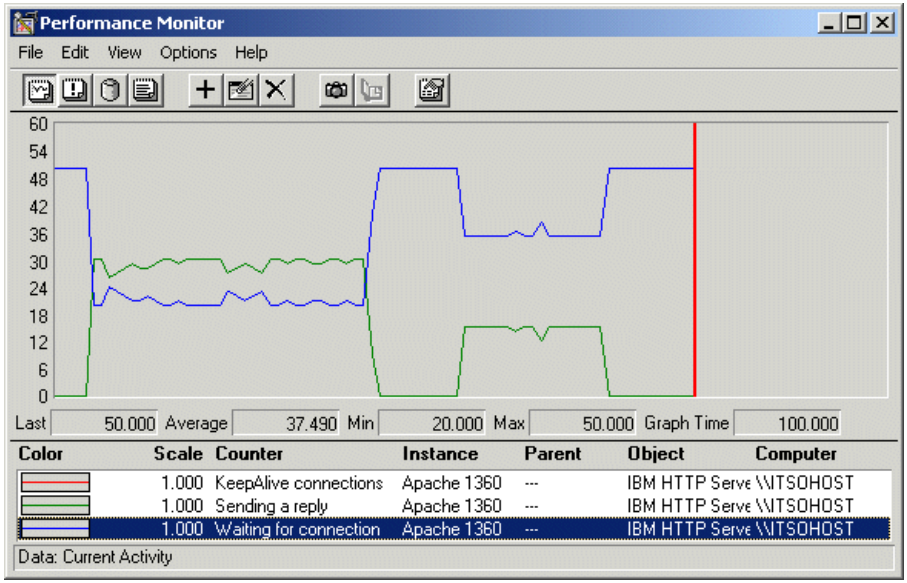


Figure 22-14 Windows NT Performance Monitor

Note: ApachePerfomanceMonitor.dll, shipped with IBM HTTP Server V1.3.19, is not compatible with the Windows 2000 performance monitor. As a work-around, we copied the Windows NT version of the performance monitor to Windows 2000, for example:

```
C:\>copy \winnt4\system32\perfmon.exe \winnt5\system32\NTperfmon.exe
```

Don't overwrite the Windows 2000 perfmon.exe, since Windows 2000 will use its cache to restore the original version.

IBM HTTP Server support for Windows 2000 performance monitor is expected to be available in June 2002.

You can also use IBM HTTP Server server-status page to view IBM HTTP Server performance data. This choice works on all platforms, not just on Windows NT/2000. Use the following steps to activate the server-status page:

1. Open the IBM HTTP Server file httpd.conf in an editor.
2. Remove the comment character "#" from the following lines:

```
#LoadModule status_module modules/ApacheModuleStatus.dll
#<Location /server-status>
#SetHandler server-status
```

```
#</Location>
```

3. Save the changes and restart the IBM HTTP Server.
4. Open the URL `http://yourhost/server-status` in a Web browser, and click **Refresh** to update the status.

   If your browser supports refresh, you can also use the URL `http://yourhost/server-status?refresh=5` to refresh every 5 seconds. As shown in Figure 22-15, you can see (along with additional information) that 12 requests are currently being processed, and there are 38 idle servers.

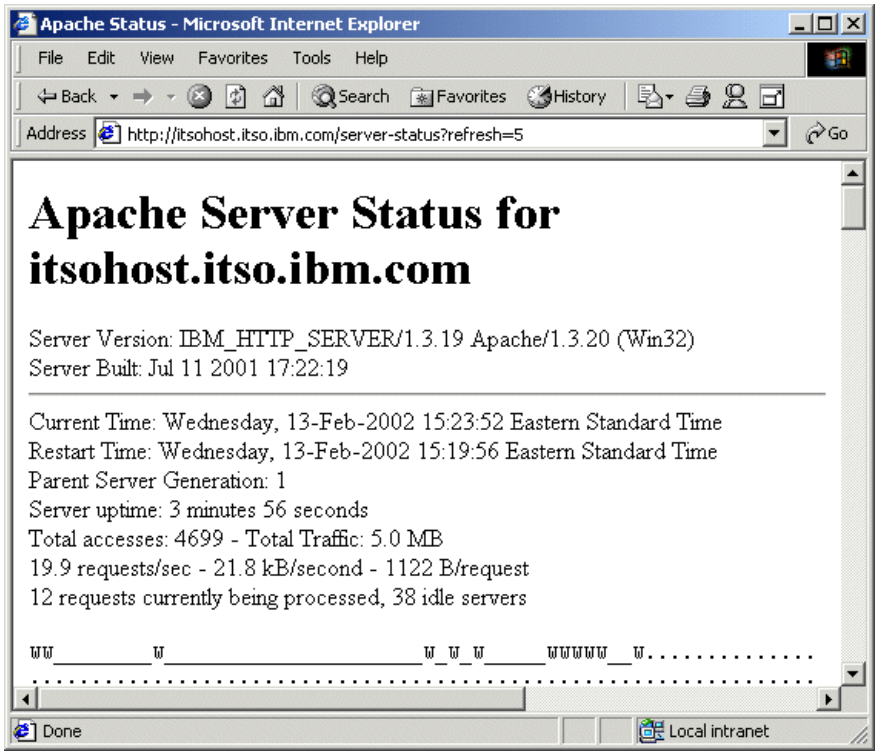


Figure 22-15 IBM HTTP Server status page

22.6 Customizing WebSphere system queues

WebSphere Application Server establishes a queuing network, which is a network of interconnected queues that represent the various components of the application serving platform. These queues contain the network, Web server, Web container, Object Request Broker, data source and possibly a connection manager to a custom back-end system.

The WebSphere queues are load-dependent resources. The average service time of a request depends on the number of concurrent clients. Figure 22-16 provides an overview of WebSphere queues, showing how a request is processed through the WebSphere environment.

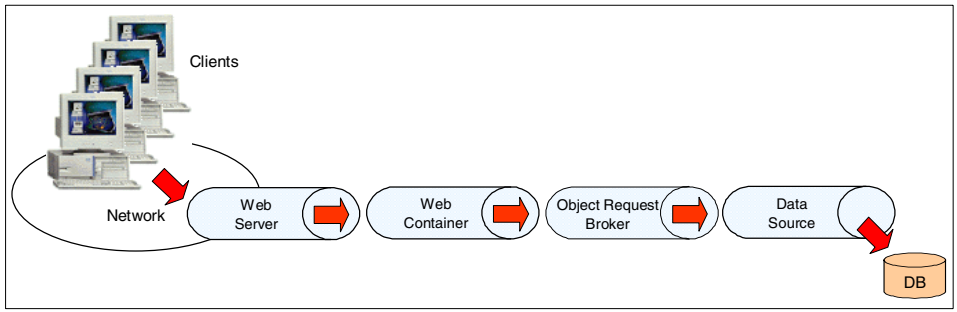


Figure 22-16 WebSphere queueing network

22.6.1 Configuring the queues

The following sections look at configurations for each of the queues.

Web server

To configure the number of maximum concurrent allowed connections, do the following:

1. Open the httpd.conf file in your preferred editor.
2. Change the value of ThreadsPerChild (Windows) or MaxClients (UNIX) to the appropriate value, as seen in Example 22-5.

Example 22-5 Parameter in the httpd.conf

```
# Number of concurrent threads at a time (set the value to more or less
# depending on the responsiveness you want and the resources you wish
# this server to consume).

ThreadsPerChild 50
```

Web container

To configure the size of your Web container queue, start the WebSphere Administrative Console and do the following:

1. Highlight your application server in the WebSphere administrative domain hierarchy.
2. Go to the property pane and click the **Services** tab.

3. Select the **Web Container Service** and click the **Edit Properties...** button to open the Web Container Service window.
4. Change the Maximum thread size to the appropriate value, as seen in Figure 22-17.

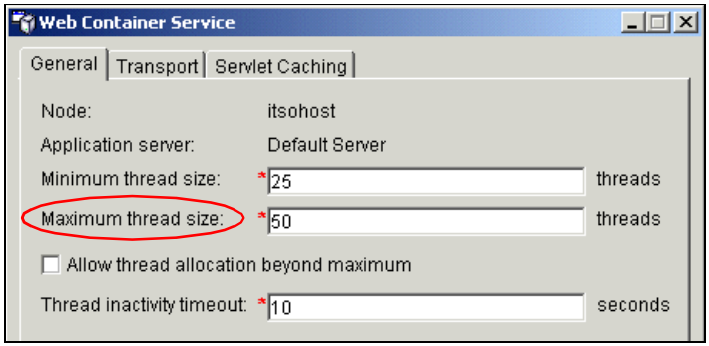


Figure 22-17 Web Container Service properties

Object request broker

To configure the thread pool size of your Object Request Broker, start the WebSphere Administrative Console and do the following:

1. Highlight your application server in the WebSphere administrative domain hierarchy.
2. Go to the property pane and click the **Services** tab.
3. Select the **Object Request Broker** service and click the **Edit Properties...** button to open the Object Request Broker window.
4. Change the Thread pool size to the appropriate value, as seen in Figure 22-18.

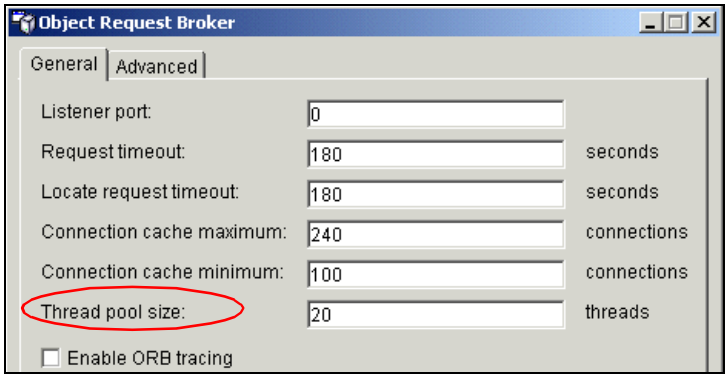


Figure 22-18 Object Request Broker properties

Data source

To configure the size of your data source connection pool, start the WebSphere Administrative Console and do the following:

1. Open the **Resources** folder in the WebSphere administrative domain hierarchy.
2. Open the **JDBC Providers** folder.
3. Open the JDBC provider that contains the required data source.
4. Select the **Data Sources** folder.
5. Right-click the required data source in the property pane and select **Properties** from the pop-up menu.
6. Select the **Connection Pooling** tab in the Data Source Properties window.
7. Change the Minimum pool size and Maximum pool size to the appropriate values, as seen in Figure 22-19.

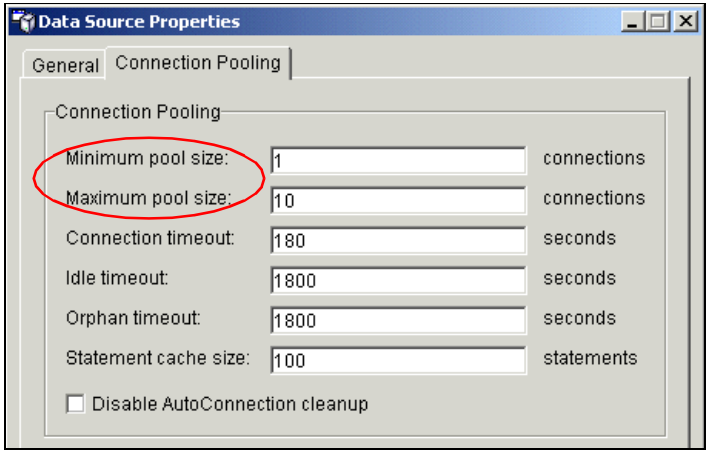


Figure 22-19 Data Source Properties

Determining the settings

The main objective is to minimize the number of requests in WebSphere queues. In general, it is better for requests to wait in the network (in front of the Web server) than it is for them to wait in the application server. In general the queues upstream (closest to the client) should be slightly larger. Queues downstream should be progressively smaller. Figure 22-20 illustrates a sample configuration that leads to client requests being queued upstream. Arriving requests are queued in the network as the number of concurrent clients increases beyond 75 concurrent users.

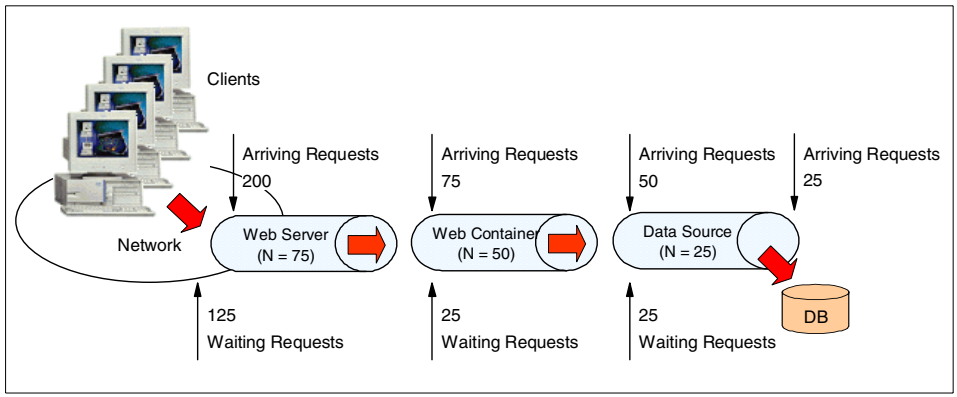


Figure 22-20 WebSphere queuing example

The example shows 200 clients arriving at the Web server. Because the Web server is set to handle 75 requests concurrently, 125 requests are pending in the network. As the 75 requests pass from the Web server to the Web container 25 remain queued in the Web server. 50 requests will be processed by the Web container. This process progresses through the data source until finally 25 users arrive at the database server. No component in this system will have to wait for work to arrive because at each point there is some work waiting to enter that component.

The bulk of requests will be waiting outside of WebSphere Application Server in the network. This will add stability to the WebSphere Application Server, because no component is overloaded. Waiting users can also be re-routed to other servers in a clustered WebSphere environment using routing software such as the IBM Network Dispatcher.

Tip: All these parameters (except the one for the HTTP server) can also be manipulated by using the Performance Tuner wizard (see 18.6, “Performance Tuner” on page 199).

22.7 Performance Tuner wizard

You can use this WebSphere Administrative Console wizard to tune common performance-related application server settings, as follows:

1. Start the Performance Tuner wizard by selecting **Console** -> **Wizards** -> **Performance Tuner** from the WebSphere Administrative Console main menu.

2. In the Application Server window, select the application server or server group that you would like to tune.
3. In the Web Container window, update the pool size within the normal range by adjusting the slider.

   Edit the numeric field to update the pool size within or outside the normal range. Think about specifying the maximum number of Web container threads less than the number of connections accepted by the Web server. Consider setting the Web container maximum threads number significantly smaller than the number of connections accepted by the Web server (MaxClients or ThreadsPerChield in httpd.conf), for a site with many static pages returned directly by the Web server.
4. In the ORB Properties window, the following Object Request Broker properties are available for tuning:
   - Pass by Reference

     This option can provide better performance. Select **Pass by Reference**, only if appropriate for your application. Selecting this option can break remote transparency, since you can modify objects passed to an EJB method. Know your application before using this option.
   - ORB threads pool size

     A thread is needed for each EJB request. Enterprise beans are typically invoked from servlets in another JVM, using RMI/IIOP and remote EJB client applications, using RMI/IIOP. The ORB thread pool size should accommodate both request sources.
5. In the Data Source window, select a data source that is associated with the selected application server and click **Next**.
   a. In the Data Source connection pool size window, base the Data source maximum connections setting for your application on the sum of the concurrent connections used by your servlets and enterprise beans.

      The servlet contribution to the connection pool size can be significantly smaller than the Web container maximum threads, if only a small percentage of servlet requests use these database connections.
   b. In the Data Source prepared statement cache size window, make your prepared statement cache large enough for all prepared statements, by setting the cache size to the product of:
      - The number of SQL prepared statements in your application
      - The maximum number of configured data source connections

      To accomplish this, you need to know very well how your application is dealing with the database.

c. In the Databases window (DB2 only), check the **Tune database** option to tune the DB2 database. When you complete the remaining windows, and click **Finish** on the Summary window, the Tuner wizard will call the DB2SmartGuide API, to tune the DB2 database associated with the data source. Stop and restart the database instance, in order for the DB2SmartGuide changes to take effect. Before tuning a database, it is recommended that you back up the database.

> **Note:** DB2SmartGuide tuning works better if the database is already populated. It is not necessary or recommended that you tune the repository database (WAS40).

6. In the JVM Heap Size window, adjust the JVM starting heap size and JVM maximum heap size.

   The Java Virtual Machine (JVM) heap size settings influence garbage collection of Java objects. If you increase the heap size, garbage collection occurs less frequently, but takes longer.

   These settings depend strongly on your application and on the amount of physical memory available. Consider:

   – Whether the JVM heap for the selected application server shares physical memory with other application server JVM Heaps on the same machine.
   – Specifying JVM heaps to reside in physical memory and prevent swapping to disk.
   – Setting the starting JVM heap size to one quarter of the maximum JVM Heap Size.
   – Setting the maximum JVM heap size to the following, if you have only one application server on the machine:
     • 128 MB, for small systems with less than 1 GB of memory
     • 256 MB, for systems with 2 GB of memory
     • 512 MB, for larger systems

   A value of 0, or blank, indicates that no starting or maximum heap size is passed when initializing the JVM.

7. In the Summary window, review your tuning properties. Click **Finish** to apply the properties, or **Back** to make further changes.

23

Command-line administration and scripting

While the WebSphere Administrative Console is a powerful tool for the day-to-day administration of your WebSphere environment, there are times when you will want to automate repetitive administration tasks. This chapter describes how you can perform common administrative tasks from the command line, using the WebSphere Control Program tool and XMLConfig.

23.1 Introducing WebSphere Control Program

This section describes the WebSphere Control Program. We will introduce the Tool command language (Tcl) and the WSCP tool, and describe the syntax and usage of both. We go on to present some sample commands and procedures, and explain how the WSCP tool can be used to administer the WebSphere Application Server.

23.1.1 Command-line administration

In WebSphere Application Server, the administrative server tracks the contents and activities of a domain by maintaining an administrative database. The administrative server allows the administration of a domain from any machine, and all information is stored in a central location. The administrative database contains information about the applications that are configured to run in the domain. For example, it contains the names of all application servers, EJB containers, Web containers, servlets and enterprise beans, and their current state (running, defined, or stopped).

All administration takes place through the manipulation of objects in the administrative database. Each resource in a domain corresponds to an object in the administrative database. For example, when you create an application server, a corresponding application server object is created in the administrative database.

The command-line WSCP interface and the administrative console are compatible. The results of actions performed with the WSCP tool are reflected in the console interface, and vice versa, although an explicit refresh is required to display the changes. Both the administrative console and the WSCP tool can be used to do the following:

- Define, configure, and manage application servers, enterprise applications and other WebSphere resources from any node in the network.
- Install enterprise applications (using wizards in the administrative console and scripts in the WSCP tool).
- Perform daily administrative operations, such as starting and stopping enterprise applications and making changes to their configuration.
- Replicate objects to improve performance or availability or to simplify administration tasks (by defining and managing server groups and clones).
- Track the occurrence of specific events by setting and enabling tracing.

What is WSCP?

The WebSphere Control Program is a command-line administrative tool for WebSphere. Using this tool, one can define, configure, and manage administrative database objects from any node, import or export configuration data, and perform diagnostic operations such as enabling a trace.

The command-line program WSCP has an interactive mode, and is particularly useful for scripting. It is based on a standard scripting language, Tcl. You can use the WSCP tool to administer the resources in a domain. It modifies the administrative database in response to your commands, and reflects any changes to the configuration and status of the domain. An administrator manipulates objects in the administrative database when executing wscp commands or scripts.

What is Tcl?

Tcl was originally developed by John Ousterhout, and is now distributed by Ajuba Solutions, formerly named Scriptics. Tcl is open source and therefore free of any licensing fees. There is also a pure-Java implementation of Tcl called JACL on which the WSCP tool is actually based. In addition to the benefits of using Java rather than platform-specific code, this enables easy invoking of Java methods in scripts using the java:: package.

The Tcl language has a simple and programmable syntax, and it can be used standalone or embedded in other applications. It is extensible, and indeed hundreds of extensions already exist. WSCP itself extends Tcl by providing a set of commands for manipulating WebSphere objects.

23.1.2 Tcl language fundamentals

Before learning about the WSCP tool, you need to have at least a basic idea about Tcl scripting. In this section, we give a very basic summary of the fundamentals of the language. If you know other programming or scripting languages, this should be enough to understand the example wscp commands presented here.

Note that this really is just a brief introduction to Tcl syntax. There are many good books available for those who wish to learn Tcl. This introduction is included for those who want a *quick start* to understanding the examples in this book.

Basic Tcl syntax

Tcl is a scripting language. Commands are separated by new lines or semicolons; words are separated by spaces. The following example uses the **expr** command, a simple command that treats the concatenation of its arguments as an arithmetic expression and returns the result as a string.

```
expr 1 + 2
```

The above command consists of four words. It will return the result of the arithmetic expression. All Tcl commands will return a result, even just an empty string.

Variables

Variables in Tcl do not need to be declared, but are created as necessary. The **set** command is used to read and write variables. The syntax is:

```
set <variablename> <value>
```

and the command will return the value that the variable has been set to. For instance:

```
set x 123
```

will set the value of the variable x to “123”, and the command will return the value “123”. You can use:

```
set <variablename>
```

or

```
$<variablename>
```

to just return the value of the variable. For instance, assume that the variable x is set to “abc”. Both of the following:

```
set y $x
set y [ set x ]
```

will set the variable y to “abc”, the value of variable x.

Variable substitution can be used as part of a word. The name of the variable is enclosed in braces to establish a single interpretation, as follows:

```
set x abc
set y ${x}def
```

This will set the variable y to “abcdef”.

Command substitution

Tcl command substitution enables the use of the result of one command as an argument for another. For example:

```
set x [expr 1 + 2]
```

This results in the variable x being set to the value 3, which is the return value of the **expr 1 + 2** command. Everything between [and the matching] is evaluated as a nested Tcl command, with the result substituted into the outer command.

Quoting

Words in Tcl are separated by spaces unless quoting is used.

Double-quote characters can be used to surround a word. When this is done, command and variable substitutions are performed inside the quotes, and the quotes themselves are not passed to the command. For instance:

```
set a 1
set b 2
set x "$a + $b == [expr $a + $b]"
```

This results in the variable x being set to the value “1 + 2 == 3”. The variables a and b are evaluated throughout the quoted section, and the command substitution inside the square brackets is evaluated. This kind of quoting is called *deferred substitution* because the unevaluated string is often evaluated later in another context (such as in a procedure).

Braces (the { } characters) can also be used to surround a word. However, no substitution is performed (the braces themselves are still not passed to the command).

```
set x {$a + $b == [expr $a + $b]}
```

This results in the variable x being set to "$a + $b == [expr $a + $b]".

Procedures

The Tcl **proc** command is used to create a Tcl procedure. The syntax is:

```
proc name {argument list} {script}
```

Here is an example procedure, called “sum”. It takes two arguments, and returns their sum:

```
proc sum {a b} {
   set return [expr $a + $b]
}
```

Basic file operations

Tcl can handle the normal stdin, stdout and stderr channels. To read a line from stdin and store it in a variable use:

```
set line [ gets stdin ]
```

To write the value of $line to stdout use:

```
puts stdout “Input was: $line“
```

To read and evalute a Tcl script, for example with some helper functions and shortcuts, use:

```
source filnename.tcl
```

If filename.tcl is not in the current working directory, you have to specify the absolute path name like this:

```
source {D:\WebSphere\AppServer\scripts\procedures.tcl}
```

Tip: Create one Tcl script that includes all the scripts you want to load before doing anything in WSCP. After you start the WSCP tool, source this file to get all your defined procedures and variables.

23.1.3 Starting the WebSphere Control Program

The WebSphere Control Program tool is started from the command line on Windows platforms by running:

```
<WAS_HOME>\bin\wscp.bat
```

On UNIX platforms use:

```
<WAS_HOME>/bin/wscp.sh
```

If the <WAS_HOME>\bin directory is in your PATH, then you can invoke the wscp command without including this directory.

Command-line options

The following command-line options are accepted by WSCP:

```
Usage: wscp [-h] [-c command] [-f file] [-p file] [-x extension] [-- options]
```

- The `-c` option evaluates the specified Tcl command.
- The `-h` option displays the usage information.
- The `-f` option specifies a file containing Tcl commands to be executed.
- The `-p` option specifies a Java properties file to be loaded instead of the default (.wscprc in the user’s home directory).
- The `-x` option specifies a Tcl extension class to be loaded.
- The `--` option is used to set Tcl argc and argv variables.

The `-c -f -p` and `-x` options may be repeated on the command line.

If no `-c` or `-f` options are specified, an interactive shell is invoked, which is terminated by the **`exit`** command. Most of the examples in this section are shown using the interactive shell.

Command syntax

The basic wscp command syntax for operating on objects is as follows:

```
<object_type> <action> [<object_name>] [ -<option> [<value>] | -attribute
<attribute_list>]
```

Where:

`object_type`	Is the name of an object type (for example, ApplicationServer)
`action`	Is the action to be performed (for example, show)
`object_name`	Is the name of the object instance
`option`	Varies by action
`value`	Applies to some options that may require values
`attribute`	Varies by object type
`attribute_list`	Specifies a list of valid attributes for this object type and action

Note: The WSCP shell is case sensitive!

Objects are referred to using the syntax:

```
/<object_type>:<object_name>/
```

For example, to refer to a node use the following syntax:

```
/Node:itsohost/
```

To refer to an application server, use the following syntax:

```
{/Node:itsohost/ApplicationServer:Default Server/}
```

Quoting in WSCP

As with Tcl, wscp commands consist of one or more words separated by spaces. Words that contains spaces must be enclosed in either braces ({ }) or double quotation marks (" "). The use of braces and double quotation marks is similar. Both braces and double quotation marks can be placed around a word that contains embedded spaces. However, using braces and double quotation marks differs in two respects. First, unlike double quotation marks, braces can be

nested. Also, no substitutions occur inside braces, as they do inside double quotation marks. All characters between braces are passed as an argument to a command or procedure, without any special processing. Substitutions can occur later if the argument is evaluated again.

Object names in the WSCP tool can contain spaces and therefore must be enclosed in either braces or double quotation marks. If object names are enclosed in braces, no evaluation takes place and everything in braces is passed to the command as an argument. If substitution must take place, for example, when the object name argument contains a variable that must be expanded to form the name, then the object name must be enclosed in double quotation marks. The example commands shown in Example 23-1 demonstrate the use of braces and double quotation marks in each case.

Example 23-1 Sample wscp commands with quoting used

```
# Braces used because object name contains spaces
# Object name is passed to the command as an argument
# and no substitution takes place

wscp> ApplicationServer create {/Node:itsohost/ApplicationServer:My Server/}

wscp> ApplicationServer list
{/Node:itsohost/ApplicationServer:Default Server/}
{/Node:itsohost/ApplicationServer:My Server/}

# Double quotation marks used because substitution must take place

wscp> set NODE /Node:itsohost/
wscp> ApplicationServer create "${NODE}ApplicationServer:Test Server/"

wscp> ApplicationServer list
{/Node:itsohost/ApplicationServer:Default Server/}
{/Node:itsohost/ApplicationServer:My Server/}
{/Node:itsohost/ApplicationServer:Test Server/}
```

Using interactive mode

If no `-c` or `-f` options are specified with the **`wscp`** command, an interactive shell is invoked. The following prompt signals that you have entered interactive mode:

```
wscp>
```

When you enter a wscp command at the interactive prompt, the WSCP tool executes the command, displays the result, and is ready to accept another command. Use the **`exit`** command to terminate a WSCP interactive session.

In an interactive WSCP session, you can break the line of a wscp command after a left brace ({) or after typing a backslash (\ ). The WSCP session displays a question mark (?) prompt, and you can continue typing the command.

Tip: With the default command interpreter (cmd.exe) on Windows platforms, you can use the history functionality (arrow keys) inside WSCP. Use F7 to get a history list of all commands from the current session.

Running scripts

To execute a WSCP script from the command line, specify the `-f` option and the name of the executable file that contains the script. For example, the following command executes the script in the myScript.tcl file:

```
wscp -f myScript.tcl
```

To execute a script from within an interactive WSCP session, use the Tcl **`source`** command to source the script. The following command runs the script in the myScript.tcl file:

```
wscp> source myScript.tcl
```

The WSCP properties file

On startup, the WSCP tool reads a Java properties file as specified by the `-p` option on the command line. If no properties file is specified, the .wscprc file in the user's home directory is loaded. This may be "C:\Documents and Settings\wasadmin\.wscprc" on Windows 2000, for example.

At a minimum, the wscp.hostName property should be specified in the properties file to point to the administrative server host. The default is set to localhost, which should work. However, it is generally preferable to explicitly specify this value. Any WebSphere property values can be added to this file, such as tracing, CORBA properties, and so on.

Refer to "Invoking and terminating wscp" in the InfoCenter for more information.

Connecting to local or remote nodes

By default, WSCP connects to the administrative server running on the local machine. Use the wscp.hostName property to specify a different host, and the wscp.hostPort property to specify a port other than the default port 900.

For example, if your administrative server is running on machine was02.yourdomain.com and on port 900, add the entries shown in Example 23-2 to your properties file.

Example 23-2 WSCP properties to connect to a remote node

```
wscp.hostName=was02.yourdomain.com
wscp.hostPort=900
```

Figure 23-1 provides an overview of how the WSCP tool interacts with local and remote nodes.

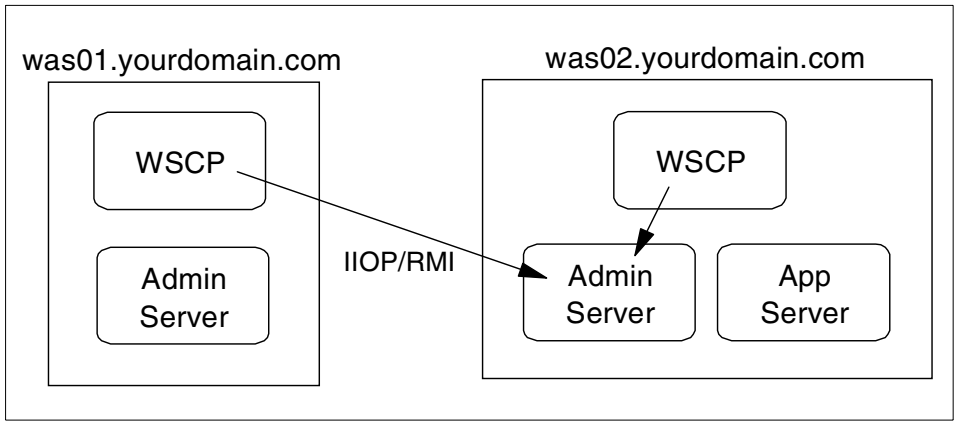


Figure 23-1 Connecting to local or remote nodes

Connecting to a secure administrative server

If security is enabled on the administrative server (as should always be the case in a production system), you have to authenticate in order to use WSCP with that server.

By default, the WSCP tool will authenticate to the administrative server using the properties specified in the <WAS_HOME>\properties\sas.client.props file. This will require that you interactively enter the user ID and password each time you start the WebSphere Control Program. To configure WSCP to authenticate with the administrative server automatically, use the following steps:

1. Copy of the <WAS_HOME>\properties\sas.client.props file from the node you want to connect to, for example:

   ```
   copy D:\WebSphere\AppServer\properties\sas.client.props sas.wscp.props
   ```

2. Edit the new sas.wscp.props file and update the following properties:

   ```
   com.ibm.CORBA.loginSource=properties
   com.ibm.CORBA.loginUserid=<your_admin_userid>
   com.ibm.CORBA.loginPassword=<your_admin_password>
   ```

3. You can use either of the following methods to specify that WSCP use the new sas.wscp.props file:

a. Put the following ConfigURL entry in your .wscprc properties file, or in a properties file specified by the `-p` command-line option (Note that it must be one line):

   For Windows:

   ```
   com.ibm.CORBA.ConfigURL=file:/D:/Websphere/AppServer/properties/sas.wscp
   .props
   ```

   For UNIX:

   ```
   com.ibm.CORBA.ConfigURL=file:///usr/WebSphere/AppServer/properties/sas.w
   scp.props
   ```

b. In the setupCmdLine.bat (Windows) or setupCmdLine.sh (Unix) file, set the value of the WSCPCLIENTSAS variable to the location of the copied file. For example on Windows:

   ```
   WSCPCLIENTSAS=-Dcom.ibm.CORBA.ConfigURL=file:/D:/Websphere/AppServer/pro
   perties/sas.wscp.props
   ```

Note: Do not try to include a subset of the properties from the sas.client.props into your .wscprc as was done in WebSphere V3.5. This will not work in WebSphere V4.0.

23.1.4 Getting online help

A **`Help`** command is built into the WSCP tool. It provides general help on the objects available within the tool. Each object also supports a `help` action, which can also give verbose action information with the `-verbose` option.

The Help command

To display the WSCP main help information type in the **`Help`** command, as shown in Example 23-3.

Example 23-3 WSCP main help information

```
wscp> Help

The general format of all wscp actions is:
        <object-type> <action> [name] [options]

The following is a list of the supported objects:

        ApplicationServer
        Context
        DataSource
        Domain
        DrAdmin
        EnterpriseApp
```

```
        GenericServer
        Help
        J2CConnectionFactory
        J2CResourceAdapter
        JDBCDriver
        JMSConnectionFactory
        JMSDestination
        JMSProvider
        MailSession
        Module
        Node
        PmiService
        Remote
        SecurityConfig
        SecurityRoleAssignment
        ServerGroup
        URL
        URLProvider
        VirtualHost
        XMLConfig

To list all actions an object supports: <object> help
To list all the options for an action:  <object> help <action>
For verbose information on an action:   <object> help <action> -verbose
```

Note: Because the **`Help`** command in the WSCP tool is implemented as a Tcl object, and because WSCP is case sensitive, it must always be referred to as Help, not help. This can be confusing at first, when typing help returns the phrase invalid command name “help”. It is possible to change the name of the **`Help`** command to help by using the Tcl **`rename`** command:

```
rename Help help
```

or by defining a procedure so that either form can be used:

```
proc help { args } { Help $args }
```

Object help

Each object supports a `help` action, which gives information about the actions available for that object. A short description is also given for each action.

Also, more detailed information on any of the actions possible on a given object can be gained by specifying the name of the action after the `help` action.

Finally, there is a `-verbose` option that will give the highest level of detail available for any given action. This will also list all of the options available for that action.

To get the online help for the ApplicationServer object type, use the command shown in Example 23-4. It also contains the output of the command.

Example 23-4 Application server help page

```
wscp> ApplicationServer help
WSCP0000I: The following actions are available for ApplicationServer

attributes          Display the attributes of the object
containment         Display the containment hierarchy for the object
create              Create the specified object
defaults            Display attribute defaults
help                Display this help message
list                Display all the instances of this type
modify              Modify the attributes of the specified object
operations          List all the actions available on the object type
remove              Remove the specified object
show                Display the attributes of specified object
showall             Display all attributes of specified object
start               Start the specified object
stop                Stop the specified object
```

23.1.5 Listing objects

To display instances of a particular object type use the following syntax:

```
<object_type> list [-constraint <a_list>] [-recursive]
```

Where:

`-constraint <a_list>`	Specifies a Tcl list of attributes to use as a constraint. Only objects with the specified attribute-value pairs are listed.
`-recursive`	Lists all instances of any object type that belongs to the containment hierarchy of the specified type.

For example, to show all application servers use the following command:

```
wscp> ApplicationServer list
{/Node:itsohost/ApplicationServer:Default Server/}
/Node:itsohost/ApplicationServer:newas01/
```

The `-recursive` option lists all instances of any object type that belongs to the containment hierarchy of the specified type. For instance, for a JDBCDriver object type, the `-recursive` option lists all instances of DataSource as shown here:

```
wscp> JDBCDriver list -recursive
```

```
{/JDBCDriver:Sample DB Driver/} {/JDBCDriver:Sample DB
Driver/DataSource:SampleDataSource/} {/JDBCDriver:Sample DB
Driver/DataSource:sample/}
```

To list only those DataSources whose DatabaseName attribute is `SAMPLE`, do the following:

```
wscp> DataSource list -constraint {{DatabaseName SAMPLE}} -recursive
{/JDBCDriver:Sample DB Driver/DataSource:sample/} {/JDBCDriver:Sample DB
Driver/DataSource:newds01/}
```

23.1.6 Displaying objects and their attributes

To display all attributes and their values or a subset of attributes of a specific object instance use the following command syntax:

```
<object_type> show <object_name> [-all] [-attribute <a_list>]
```

Where:

`object_type`	Specifies the object type of the instance.
`object_name`	Specifies the object instance whose attributes are to be displayed.
`-all`	Displays the values of all attributes (those that are set as well as those that are not set).
`-attribute <a_list>`	Specifies a Tcl list of attributes to display.

For instance to show the attributes related to a node and their values use the following command.

```
wscp> Node show /Node:itsohost/
{Name itsohost} {FullName /Node:itsohost/} {CurrentState Running}
{DesiredState Running} {StartTime 995750284613} {OsName {Windows NT}}
{HostName itsohost} {HostSystemType x86} {ProcessId 205} {InstallRoot
{D:\WebSphere\AppServer}}
```

Application server attributes

By default, the ApplicationServer `show` action displays all application server attributes except the JVMConfig, ORBConfig, and WebContainerConfig attributes, which have very long multi-part values.

To see all attributes do one of the following:

- Use the `showall` option:

  ```
  wscp> ApplicationServer showall /Node:itsohost/ApplicationServer:newas01/
  ```

- Specify the attributes your are interested as shown here:

```
wscp> ApplicationServer show \
/Node:itsohost/ApplicationServer:newasO1/ \
-attribute {JVMConfig}
{JVMConfig {{AdditionalCommandLineArgs {}} {BootClasspathAppend {}}
{BootClasspathPrepend {}} {BootClasspathReplace {}} {Classpaths {}}
{DebugMode false} {DebugString {}} {DisableJIT false} {HProfArgs {}}
{InitialHeapSize 64} {MaxHeapSize 256} {RunHProf false} {SystemProperties
{}} {VerboseClassLoading false} {VerboseGC false} {VerboseJNI false}
{GeneratedCommandLineArgs -Xms64m -Xmx256m }}}
```

Server group attributes

In addition to the `show` action, the object type ServerGroup has the `showAttrs` action.

`show`	Displays the attributes associated with the server group. For example, the CloneInterfaceClass, IfStarted, and StartTime attributes.
`showAttrs`	Displays the clone attributes associated with the server group. (These attributes match the properties for the resource type of the server group, such as a application server.)

For example, try the following commands:

```
wscp> ServerGroup show /ServerGroup:sgO1/
wscp> ServerGroup showAttrs /ServerGroup:sgO1/
```

The output is too long to include here.

23.1.7 Creating resources

To create a new object use the following syntax:

```
<object_type> create <object_name> [-attribute <a_list>]
```

The arguments are as follows:

`object_type`	Specifies the object type of the instance.
`object_name`	Specifies the object instance to be created.
`-attribute <a_list>`	Specifies a Tcl list of attribute-value pairs to set.

For instance, to create a new application server, use the following command:

```
wscp> ApplicationServer create /Node:itsohost/ApplicationServer:newasO1/
```

Some object attributes are required to create an object. However, others are optional. To figure out which attributes are required and which are optional for an application server use the following command:

```
wscp> ApplicationServer attributes -required
Name
```

As you will see only the Name is required. However, some objects have more than the name as required attributes, for instance a JDBCDriver:

```
wscp> JDBCDriver attributes -required
Name ImplClass
```

Therefore you need to specify those attributes for creation of such an object. The following command creates an JDBCDriver for a DB2 database:

```
wscp> JDBCDriver create /JDBCDriver:newDB2Driver/ -attribute \
{{ImplClass COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource}}
```

Note that this only creates a JDBCDriver object. You need to install the driver for that object on each node in your WebSphere administrative domain using the `install` action.

For more examples on how to use the WSCP tool to set up an environment, please see 23.2, “Creating your runtime environment using WSCP” on page 911.

23.1.8 Updating resources

To modify one or more attributes use the following syntax:

```
<object_type> modify <object_name> -attribute <a_list>
```

Where:

`object_type`	Specifies the object type of the instance.
`object_name`	Specifies the object instance whose attributes are to be modified.
`-attribute <a_list>`	Specifies a Tcl list of attribute-value pairs to set.

The following example shows how to change the JVMConfig heap attributes of an application server:

```
wscp> ApplicationServer modify /Node:itsohost/ApplicationServer:newas01/
-attribute {{JVMConfig {{InitialHeapSize 128} {MaxHeapSize 256}}}}
```

23.1.9 Starting and stopping resources

The WSCP tool can be used to start and stop resources. Valid resources for these actions are:

- Nodes
- Application servers

- Generic servers
- Enterprise applications
- Modules
- Server groups

The syntax for the `start` and `stop` actions is:

```
<object_type> start <object_name> [-wait <seconds>]
<object_type> stop <object_name> [-force] [-wait <seconds>]
```

Where:

object_type	Specifies the object type of the instance.
object_name	Specifies the object instance whose attributes are to be modified.
-wait <seconds>	Specifies the wait time in seconds before the action starts.
-force	Forces the object to stop.

To start and stop the previously created application server from within WSCP, use the following command:

```
wscp> ApplicationServer start /Node:itsohost/ApplicationServer:newas01/
wscp> ApplicationServer stop /Node:itsohost/ApplicationServer:newas01/
```

23.1.10 Removing resources

The `remove` action removes the object (and all references to the object) from the domain. The object must be stopped before being removed. The syntax of the `remove` action is as follows:

```
<object_type> remove <object_name> [-recursive]
```

If the object to be removed contains other objects, removal fails unless you use `-recursive` option. This option removes all instances of any resource type that belongs to the containment hierarchy of the resource instance being removed. For example, if a node is removed recursively all resources belonging to the containment hierarchy of the node are also removed. The recursion takes place in a downward direction.

To remove the previously created JDBC driver and the contained data source do the following:

```
wscp> JDBCDriver remove /JDBCDriver:newDB2Driver/ -recursive
```

23.1.11 Configuring security

The WSCP SecurityConfig actions can be used to do the following:

- Enable and disable WebSphere security on a global basis.
- Specify the default authentication mechanism.
- Configure secure sockets layer (SSL) communication.

They cannot be used to configure security for individual applications or components. With WebSphere V4.0, application security is specified in J2EE enterprise application deployment descriptors.

The following example command checks whether security is enabled:

```
wscp> SecurityConfig isSecurityEnabled
```

Return values are:

- 1 (true) if security is enabled.
- 0 (false) if security is disabled.

The following example command enables security for all applications:

```
wscp> SecurityConfig enableSecurity
```

The following example command disables security for all applications:

```
wscp> SecurityConfig disableSecurity
```

The following example command returns the current authentication mechanism for security:

```
wscp> SecurityConfig getAuthenticationMechanism
```

Possible return values are:

- LOCALOS

  The underlying operating system's authentication mechanism. The local operating system supports basic authentication such as checking a user ID and password.

- LTPA

  Lightweight Third Party Authentication (LTPA). LTPA authenticates users with a Lightweight Directory Access Protocol (LDAP) directory service and supports certificate-based authentication. LTPA can also use a custom user registry.

Note: LTPA cannot be directly configured from the wscp command line because the configuration settings are too complex. However, you can use the WSCP XMLConfig object to import LTPA configurations that have been stored in XML files. See 23.1.15, “Exporting and importing a configuration using XMLConfig” on page 902 for instructions on how to use this command.

The following example command returns the administrative server user ID:

```
wscp> SecurityConfig getUserid
wasadmin
```

The following example command sets the authentication method to local operating system with user ID wasadmin and password wasadmin:

```
wscp> SecurityConfig setAuthenticationMechanism LOCALOS -userid {wasadmin
wasadmin}
```

WebSphere will now use the operating system's security repository to authenticate users. The administrative server must be restarted for the change to take effect.

The following example command displays information about how SSL is configured in WebSphere Application Server:

```
wscp> SecurityConfig getSSLConfig
{{KeyFileName ${WAS_HOME}/etc/DummyServerKeyFile.jks} {KeyFilePassword
WebAS} {KeyFileFormat 0} {TrustFileName
${WAS_HOME}/etc/DummyServerTrustFile.jks} {TrustFilePassword WebAS}
{TrustFileFormat 0} {ClientAuthentication false} {UseGlobalDe
faults true} {SecurityLevel 0} {CryptoHardwareEnabled false}
{CryptoTokenType {}} {CryptoLibraryFile {}} {CryptoPassword {}}
{SSLProperties {}}}
```

23.1.12 Managing security roles

The WSCP SecurityRoleAssignment actions allow you to manage security roles for J2EE applications.

Role-based security enables declarative, customized authentication for applications. When a J2EE application is assembled, permission to execute methods is granted to one or more roles, which represent abstract groups of users. When the application is deployed, actual users or groups of users are assigned to these roles. When the application is run, WebSphere Application Server authorizes client requests based on the user's identification information and what roles the user is assigned to. For a more detailed description about role-based authentication, see Chapter 7, “Security” on page 103.

The WSCP SecurityRoleAssignment actions can be used to perform the following tasks:

- List the roles that are defined for an enterprise application.
- List the users and groups that are assigned to each role.
- Add users and groups to a role.
- Delete users and groups from a role.
- Specify the identity under which enterprise bean methods are executed. For more information about this feature, see Chapter 21, “Configuring security” on page 739.

In addition to individual users and groups, the special groups, all users and all authenticated users can be assigned to security roles.

Note that the SecurityRoleAssignment actions only work with existing security roles; they cannot be used to define new roles. However, you can assign security roles and users when you install an enterprise application with the WSCP **`EnterpriseApp install`** command or install a module with the WSCP **`Module install`** command. See 23.2.6, “Creating the enterprise application” on page 917 for more information.

To map roles to users or groups there are three different types of mappings:

- For individual users use the following actions:

```
getUserRoleMapping
addUserRoleMapping
deleteUserRoleMapping
```

- For groups use the following actions:

```
getGroupRoleMapping
addGroupRoleMapping
deleteGroupRoleMapping
```

- For special predefined groups (Everyone, AllAuthenticatedUsers) use the following commands:

```
getSpecialRoleMapping
addSpecialRoleMapping
deleteSpecialRoleMapping
```

List the defined roles

The following example command lists the roles defined for the Webbank enterprise application:

```
wscp> SecurityRoleAssignment listRoles /EnterpriseApp:Webbank/
AllAuthenticated Everyone Employee Manager
```

Get the role mapping

The following example command lists the roles defined for the Banking application and the users assigned to each role:

```
wscp> SecurityRoleAssignment getUserRoleMapping /EnterpriseApp:Webbank/
```

This operation returns nothing when no role-to-individual-user mappings have been made; otherwise it would show the current mappings.

Mapping a role to a user

The following command adds the mapping from Manager role to wasadmin user:

```
wscp> SecurityRoleAssignment addUserRoleMapping /EnterpriseApp:Webbank/
-userroles {{Manager wasadmin}}
```

Deleting a mapping

The following command deletes the mapping from Manager role to wasadmin user, without affecting other user mappings for this role:

```
wscp> SecurityRoleAssignment deleteUserRoleMapping /EnterpriseApp:Webbank/
-userroles {Manager wasadmin}
```

23.1.13 Procedures and variables can make your life easier

To avoid typing the long object names like:

```
/Node:itsohost/ApplicationServer:Default Server/
```

It can be easier to create variables instead:

```
set node /Node:itsohost/
set server "${node}ApplicationServer:Default Server/"
```

The quotation mark " is necessary because the application server name contains a space. The {} can't be used because the $node must be evaluated.

Together with the abbreviation for application server (see "Using abbreviations" on page 905), you can use the `show` action as follows:

```
wscp> App show ${server}
```

As seen in 23.1.4, "Getting online help" on page 891, you can use a procedure if you want to get the main help page using all lowercase "help" instead of "Help":

```
wscp> proc help { args } { Help $args }
```

See also 23.1.18, "An example WSCP procedure to show attribute listings" on page 907.

23.1.14 Handling status and error information

If a wscp command returns success, a result string of the command is returned. This result string may contain a single value, a list of values, or be empty. The interactive WSCP shell will display non-empty results.

If a wscp command fails, three things happen:

- A TclException is raised. This may be caught using the Tcl **catch** command. If the exception is not caught, execution of the current procedure or command substitution is terminated.
- The Tcl variable errorCode is set. This will contain status values and statusToString provided by the WscpStatus class. This variable is reset prior to each WSCP-specific command.
- The Tcl variable errorInfo is set if WSCP caught an exception. This will contain one or more stack traces of the exception(s). The variable contents persist until explicitly set by the next exception. They can be displayed in the same way as variables that you set yourself; set errorInfo will display the contents of the errorInfo variable, for instance.

As an example, you may want to create an application server and leave out the node name by mistake:

```
wscp> ApplicationServer create /ApplicationServer:news01/
WSCP0045E: Invalid object name : /ApplicationServer:newas01/
```

Obviously there is something wrong. Check the errorCode as follows:

```
wscp> puts $errorCode
10
```

The errorCode is 10. If errorCode is not set or different from 0 then check the errorInfo variable to see more details about the error. Use this command:

```
wscp> puts $errorInfo
WSCP0045E: Invalid object name : /ApplicationServer:news01/
    while executing
"ApplicationServer create /ApplicationServer:news01/"
```

23.1.15 Exporting and importing a configuration using XMLConfig

The XML Configuration Management tool (XMLConfig) can be invoked from within the WSCP tool. The syntax is as follows:

```
XMLConfig export <file_name> [-partial <file_name>]
XMLConfig import <file_name> [-substitute <list>]
```

Where:

`<file_name>`	Specifies the name of the XML file to read from, or write into.
`-partial <file_name>`	Specifies the name of the XML file that contains XML statements specifying what to export.
`-substitute <list>`	Specifies a list of variable-value pairs to substitute in the XML import stream. Syntax is: {{variable value}...}

To export the whole configuration use the following command:

```
wscp> XMLConfig export wasconfig.xml
```

To import an XML configuration file that contains the variable `$node$` for node name and `$server$` for the application server name use the following command:

```
wscp> XMLConfig import wasconfig.xml \
-substitute {{node itsohost} {server newas01}}
```

23.1.16 Working with WSCP scripts

WSCP scripts can be used in different ways. You can write your own procedures to make common tasks easier to handle, or you can write a WSCP script that creates a whole environment and installs enterprise applications. This section only shows the basics. For a complete example please refer to 23.2, “Creating your runtime environment using WSCP” on page 911.

Creating a WSCP script

To create a WSCP script to show the current status of all application servers and clones, follow these steps:

1. Open your favorite text editor.
2. Type in the WSCP script shown in Example 23-5. This script provides a procedure that goes through the list of all application servers and displays the Name and CurrentState of each.
3. Save the file as showServerStatus.tcl.

See Appendix D, “Additional material” on page 1083 to obtain this script.

Example 23-5 Procedure to display status of all servers

```
#
# Procedure for displaying selected attributes of application servers
#

proc showServerStatus {} {

   puts "\nStatus of servers in the domain:\n"
```

```
    foreach ejbserver [ApplicationServer list] {
    puts [ApplicationServer show $ejbserver -attribute {Name CurrentState}]
    }
}
```

Run a WSCP script

To run this script do the following steps:

1. Start the WebSphere Control Program.
2. Type the following command:

```
wscp> source showServerStatus.tcl
```

3. Use the following command to execute the procedure:

```
wscp> showServerStatus

Status of servers in the domain:

{Name {Default Server}} {CurrentState Stopped}
{Name newas01} {CurrentState Running}
{Name clone01} {CurrentState Stopped}
{Name clone02} {CurrentState Stopped}
{Name newsa01} {CurrentState Stopped}
```

After exiting the WSCP shell and restarting it you need to source the WSCP script again because the **source** command is not permanent. If you want to load your favorite procedures at WSCP startup, do the following:

1. Create a WSCP script (for instance initialWSCP.tcl) that includes all your procedures or sources other scripts. This script may look like this, for example:

```
procedure myFirstProcedure { <tcl commands> }
procedure mySecondProcedure { <tcl commands> }
source firstScript.tcl
source secondScript.tcl
```

2. Start your WSCP shell.
3. Source your initial script (initialWSCP.tcl):

```
wscp> source initialWSCP.tcl
```

Note: If you use the script file name without the path specification, it must be in the current working directory. To specify the absolute path to the file, use surrounding braces {}.

23.1.17 Advanced usage of WSCP

In this section we look at advanced usage of WSCP, including:

- System commands
- Using abbreviations
- Specifying lists
- Using qualified home names

System commands from within the WSCP tool

All UNIX and Windows commands can be used from within a WSCP session. In interactive mode, Tcl executes operating system commands with an explicit **exec** command. The **exec** command creates one or more subprocesses and waits until they complete before returning. For a directory listing on UNIX, for example, type the following command in interactive mode:

```
wscp> exec ls -l
```

If you wish to disable this behavior and instead have Tcl search your UNIX or Windows PATH environment variable for command names before checking whether they are abbreviations, enter the following command in interactive mode:

```
wscp> unset auto_noexec
```

To check the definition of the variable and to reset it to true, enter the following command in interactive mode:

```
wscp> set auto_noexec
wscp> set auto_noexec true
```

To get a list of all global variables currently defined, type in the following command:

```
wscp> info globals
```

Using abbreviations

In interactive mode, WSCP allows abbreviations of the first word of a command only (that is, for the object type). All the options must be fully specified. If an abbreviation is ambiguous, WSCP returns an error that includes a list of all matches.

For example, try typing in the following:

```
wscp> JMS
```

You will receive the following message:

```
ambiguous command name "JMS": JMSConnectionFactory JMSDestination
JMSProvider
```

To resolve the ambiguity, you must provide as many characters as needed to uniquely identify the desired object type, for example `JMSD`.

Specifying lists in wscp commands

Some WSCP options take Tcl lists as values (notably the `-attribute` option of create and modify actions). A Tcl list is an ordered collection of elements, where each element can be a string, a number, or another list. Lists can be delimited by either double quotation marks (" ") or braces ({ }). The following example sets the variable x to a three-element list whose first element is the string JVMConfig, whose second element is itself a list containing the two strings InitialHeapSize and 64, so the third element contains the two strings MaxHeapSize and 128.

```
set attr {JVMConfig {InitialHeapSize 64} {MaxHeapSize 128}}
```

In WSCP, an attribute list is a list of attribute-value pairs (a Tcl list of lists). The following command modifies the `PingInterval` and `PingTimeout` attributes of an application server. The argument to the `-attribute` option is a list containing two elements. Both elements are themselves lists.

```
wscp> ApplicationServer modify /Node:itsohost/ApplicationServer:myServer/ \
-attribute {{PingInterval 120} {PingTimeout 240}}
```

An attribute list containing only one attribute-value pair must also be enclosed in a list. For example, in the following command, the attribute-value pair {PingTimeout 240} must be nested within braces:

```
wscp> ApplicationServer modify /Node:itsohost/ApplicationServer:myServer/ \
-attribute {{PingTimeout 240}}
```

Note: In addition to the `modify` action, the `show` action has an `-attribute` option that expects a Tcl list as its argument. If you specify only one attribute to be displayed, you can use either a string or a Tcl list. Tcl interprets both the string and the Tcl list of one element as equivalent. For example, the arguments JarFile and {JarFile} are equivalent.

To let Tcl create the right order and count of braces, use the Tcl list command as shown in the following code lines:

```
# all the braces written by hand
ApplicationServer create /Node:itsohost/ApplicationServer:myServer/
-attribute {{JVMConfig {{InitialHeapSize 64} {MaxHeapSize 128}}}}

# let Tcl produce the braces
set initheapattr [list InitialHeapSize 64]
set maxheapattr [list MaxHeapSize 128]
```

```
set heapattr [list $initheapattr $maxheapattr]
set jvmconfigattr [list JVMConfig $heapattr]
set attributelist [list $jvmconfigattr]
ApplicationServer create /Node:itsohost/ApplicationServer:myServer/
-attribute $attributelist
```

Using qualified home names in the administrative server

The following property is used by the administrative server to set up the Java Naming and Directory Interface (JNDI) namespace. It specifies whether enterprise bean home names are looked up in the initial context or in a specified subcontext. If this property is set to true, bean home names are looked up in the specified subcontext.

```
com.ibm.ejs.sm.adminServer.qualifyHomeName
```

Both the administrative server and WSCP, by default, have this property set to true. If the administrative server is running with qualified home names, WSCP must also run with qualified home names. The following property controls the use of qualified home names in WSCP:

```
wscp.qualifyHomeName
```

Note: This property is not related to the fully qualified names used for object instances.

To disable the use of qualified home names, do the following:

1. In admin.config set the following property:

   ```
   com.ibm.ejs.sm.adminServer.qualifyHomeName=false
   ```

2. Set the corresponding property for WSCP in the .wscprc:

   ```
   wscp.qualifyHomeName=false
   ```

3. Restart the administrative server.

23.1.18 An example WSCP procedure to show attribute listings

The output of the `show` action on an object can be difficult to read. The procedure in Example 23-6 can be used to show object attributes in a way thay is more readable. You can either type in the procedure in the interactive WSCP shell or create a Tcl script. Follow the steps to create the new **`display`** command:

1. Open your favorite text editor.
2. Type in the Tcl procedure shown in Example 23-6.

This procedure takes two parameters, the object type and the name, and executes the wscp command **show**. The result is a list of the attributes of the supplied object. Each first level attribute (attributes can be nested) starts on a new line. You can modify the procedure to get the full range of attributes by adding the `-all` option after `$name`.

3. Save the script as disp.tcl.

Example 23-6 Tcl procedure display

```
#
# Procedure for displaying attributes in a more readable format
#
proc display {type name} {
   set attrs [$type show $name]
   foreach attr $attrs {
      puts $attr
   }
}
```

You can make this procedure available inside the WSCP tool and test it using the following steps:

1. Start the WebSphere Control Program.
2. Make sure your are in the correct folder and type the following:

   ```
   source disp.tcl
   ```

3. Try the new command:

   ```
   display ApplicationServer {/Node:itsohost/ApplicationServer:Default
   Server/}
   ```

For more examples, including how to set up a whole new environment and install enterprise applications using WSCP, refer to 23.2, “Creating your runtime environment using WSCP” on page 911.

23.1.19 Migrating WSCP scripts from V3.5.x to V4.0

This section summarizes some of the changes in WSCP for WebSphere V4.0 that can affect existing WSCP Tcl scripts. It is intended as a guide for determining whether these scripts need to be updated and is not an exhaustive description of WSCP changes.

You need to retest all scripts created under earlier versions of WSCP before migrating them to a production environment that runs the latest version of the software.

Discontinued objects

The following objects are no longer supported in WSCP:

- EJBContainer
- EnterpriseBean
- Servlet
- ServletEngine
- ServletRedirector
- SessionManager
- WebApplication
- WebResource
- UserProfile

Although they can no longer be administered on an individual basis, enterprise beans, servlets, and Web applications that have been packaged into enterprise applications or modules can be installed, started, and stopped with the EnterpriseApp and Module objects. See 23.2, “Creating your runtime environment using WSCP” on page 911.

Renamed objects and functional or syntax changes

The objects listed in Table 23-1 have been renamed or their functionality and syntax have changed. See the listed articles for more information on their new syntax and options.

Table 23-1 Changed WSCP objects in WebSphere V4.0

Old name	New name	Functional changes
EnterpriseApplication	EnterpriseApp	Application components are now packaged in modules. You no longer have to explicitly install all components of an enterprise application. See Chapter 19, “Deploying an application” on page 687.
Model	ServerGroup	Can only create server groups and clones of application servers. See Chapter 17, “Server groups and workload management” on page 605.
DataSource	DataSource	Syntax changes. See 16.1, “JDBC providers” on page 564.

New objects and new functionality

The objects listed in Table 23-2 are new for WebSphere V4.0 and represent areas of functionality that can now be handled in scripts. See the listed articles for more information on these objects and services.

Table 23-2 New WSCP objects in WebSphere V4.0

New objects and services	New functional areas	Documentation
Module	Java 2 Enterprise (J2EE) modules	Chapter 18, “Packaging an application” on page 639
J2CConnectionFactory	Enterprise Information Systems (EIS) connectivity	Chapter 16, “Configuring WebSphere resources” on page 563
J2CResourceAdapter		
JMSPRovider	Java Message Service (JMS)	Chapter 16, “Configuring WebSphere resources” on page 563
JMSConnectionFactory		
JMSDestination		
URL	URL	Chapter 16, “Configuring WebSphere resources” on page 563
URL Provider		
PmiService	Performance data	Chapter 22, “Monitoring and tuning your runtime environment” on page 839
SecurityConfig	Global Security settings	Chapter 21, “Configuring security” on page 739
SecurityRoleAssignment	J2EE security roles	Chapter 21, “Configuring security” on page 739

23.1.20 Additional resources

- Tcl Web site:

  http://www.ajubasolutions.com/
- Tcl Developers Exchange:

  http://www.scriptics.com/
- Tcl manual pages:

  http://tcl.activestate.com/man/
- Tcl Java integration (JACL):

  http://www.scriptics.com/software/java/

23.2 Creating your runtime environment using WSCP

In this section we explain how to write a WSCP script to create, configure and install all the bits and pieces needed to run the Webbank sample application (see Appendix D, “Additional material” on page 1083 to obtain webbank.ear). The WSCP scripts used here are createWebbankEnv.tcl and deleteWebbankEnv.tcl, also included in the additional material.

Notes:

1. In some of the code examples, the longer command lines are wrapped because they are too wide for the page. wscp commands must always be on one line when using scripts.
2. WSCP does not validate all the values you provide, as does the administrative console! You should double-check your settings, test your script and all the created and installed objects.

23.2.1 Assumptions

We make the following assumptions in this example:

- Your Web server is configured for serving the Webbank application on `www.webbank.itso.ibm.com` port 90 and 443 (HTTPS). See Chapter 19, “Deploying an application” on page 687 for how to set up your HTTP server.
- The WebSphere administrative server is in running state. See 13.2.1, “Starting and stopping a node” on page 459 for how to start the node.
- The IBM DB2 instance is up and running.
- DB2 administrative user is db2admin and password is db2admin.
- Database WEBBANK (for application data) is already created as described in 19.5, “Creating and populating the Webbank database” on page 705.
- Database SESSIONS (for session persistence) is already created in the same way as the WEBBANK database.
- The <WAS_HOME> is D:\WebSphere\AppServer.
- The Webbank application server working/logs directories exist, created as follows:

  ```
  mkdir D:\webbank\WebbankServer01\logs
  mkdir D:\webbank\WebbankServer02\logs
  ```

- Path to EAR file <WAS_HOME>\installableApps. The Webbank enterprise application archive (EAR) is webbank_deployed.ear. Copy the EAR from the additional material, as follows:

```
copy \Webbank\webbank.ear <WAS_HOME>\installableApps\webbank_deployed.ear
```

- For security configuration, we assume that a user wasadmin with the password wasadmin exists.
- To avoid naming conflicts, we assume that only the default installation has been performed (Default Server and sampleApp).
- Target platform for the provided examples and scripts is Windows. For UNIX, you only need to change the path specifications.

The WSCP (Tcl) scripts provided here use variables to set all the names, ports, paths, and so on. So it is very simple to modify the scripts to match your environment. The following sections show each step without using the variables.

23.2.2 Creating the virtual host

To create the new virtual host, first check the required attributes, as follows:

```
wscp> VirtualHost attributes -required
Name AliasList
```

Example 23-7 shows the commands to construct the attribute list and create the virtual host.

Example 23-7 Creating the virtual host

```
set aliaslist [list www.webbank.itso.ibm.com:9090 www.webbank.itso.ibm.com:90
www.webbank.itso.ibm.com:443]
set aliasattr [list AliasList $aliaslist]
set attributelist [list $aliasattr]

VirtualHost create /VirtualHost:webbank_vhost/ -attribute $attributelist
```

With these settings, the application server will accept URLs connecting to host name `www.webbank.itso.ibm.com` on port 90, 9090 and 443 (for HTTPS). You do not need to specify the MIME table, since it will be created by default.

23.2.3 Creating the JDBC driver

To create the new JDBC driver, first check the required attributes, as follows:

```
wscp> JDBC attributes -required
Name ImplClass
```

Example 23-8 shows the commands to construct the attribute list and create the JDBC driver for a DB2 database.

Example 23-8 Creating the JDBC driver

```
set implclattr [list ImplClass COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource]
```

```
set descattr [list Description "Webbank DB2 JDBC Driver build by wscp"]
set attributelist [list $implclattr $descattr]

JDBCDriver create /JDBCDriver:WebbankDB2Driver/ -attribute $attributelist
```

After creating the driver, you need to install the Java archive containing the driver to the node.

Installing the driver

To install the JDBC driver to a node, first check the required attributes, as follows:

```
wscp> JDBCDriver help install
JDBCDriver install <name> -node <name> -jarFile <filename>
```

To install the driver (Java archive) to a particular node, use the commands shown in Example 23-9.

Example 23-9 Installing JDBC driver on node

```
JDBCDriver install /JDBCDriver:WebbankDB2Driver/ -node /Node:itsohost/ -jarFile
"D:/SQLLIB/java/db2java.zip"
```

After installing the driver onto a node, the driver can be used at runtime. Normally applications do not use the driver directly, they using data source with connection pooling instead. You create a data source next.

23.2.4 Creating data sources

To create a data source for the Container Managed Persistence (CMP) enterprise beans, first check the attributes of the object:

```
wscp> DataSource attributes -required
Name

wscp> DataSource attributes
Name FullName ConfigProperties ConnTimeout DatabaseName DefaultPassword
DefaultUser Description DisableAutoConnectionCleanup IdleTimeout JNDIName
MaxPoolSize MinPoolSize OrphanTimeout StatementCacheSize
```

You should specify JNDI name for the data source. It must be the same as defined during the packaging of the enterprise application. Specify the database user ID and password that are used to make database connections.

In our example we are using two data sources, one for the CMP beans and one for session persistence. We explain how to do this next.

Data source for CMP enterprise beans

To construct the attribute list and create the CMP beans data source, use the commands shown in Example 23-10.

Example 23-10 Creating the data source for CMP beans

```
set dbnameattr [list DatabaseName WEBBANK]
set dbuserattr [list DefaultUser db2admin]
set dbpasswdattr [list DefaultPassword db2admin]
set jndinameattr [list JNDIName jdbc/webbank]
set dsdescattr [list Description "Webbank DataSource created by wscp"]
set attributelist [list $dbnameattr $dbuserattr $dbpasswdattr $jndinameattr
$dsdescattr]

DataSource create /JDBCDriver:WebbankDB2Driver/DataSource:WebbankDS/ -attribute
$attributelist
```

Important: Make sure you have created your application database, as advised in 23.2.1, “Assumptions” on page 911. For the Webbank application, see 19.5, “Creating and populating the Webbank database” on page 705.

Data source for session persistence

To construct the attribute list and create the session persistence data source, use the commands shown in Example 23-11.

Example 23-11 Creating data source for session persistence

```
set dbnameattr [list DatabaseName SESSIONS]
set dbuserattr [list DefaultUser db2admin]
set dbpasswdattr [list DefaultPassword db2admin]
set jndinameattr [list JNDIName jdbc/webbanksessions]
set dsdescattr [list Description "DataSource for session persistence"]
set attributelist [list $dbnameattr $dbuserattr $dbpasswdattr $jndinameattr
$dsdescattr]

DataSource create /JDBCDriver:WebbankDB2Driver/DataSource:WebbankSessions/
-attribute $attributelist
```

23.2.5 Creating the application server

To create the application server for our enterprise application, first check the required attributes for application server object, as follows:

```
wscp> ApplicationServer attributes -required
Name
```

To construct the attribute list with specific values for the JVM heap size and create the application server, use the commands shown in Example 23-12.

Example 23-12 Creating the application server

```
set minheapattr [list InitialHeapSize 64]
set maxheapattr [list MaxHeapSize 128]
set heapattr [list $minheapattr $maxheapattr]
set jvmconfigattr [list JVMConfig $heapattr]
set attributelist [list $jvmconfigattr]

ApplicationServer create /Node:itsohost/ApplicationServer:WebbankServer01/
-attribute $attributelist
```

The following sections show how to modify the new application server. Normally all of the following attributes are supplied with the `create` action. We do it this way for demonstration purposes.

Changing the HTTP port for Web server plug-in

To construct the attribute list and modify the application server to use another Web container port (9090) rather than the default (9080) use the commands shown in Example 23-13.

Example 23-13 Modifying the application server plug-in port

```
set portattr [list Port 9090]
set httptransattr [list $portattr]
set transportattr [list Transports $httptransattr]
set webcontainerattr [list WebContainerConfig $transportattr]
set attributelist [list $webcontainerattr]

ApplicationServer modify /Node:itsohost/ApplicationServer:WebbankServer01/
-attribute $attributelist
```

Changing module visibility

To construct the attribute list and modify the module visibility to Application for the application server (the default is Module), use the commands shown in Example 23-14.

Example 23-14 Modifying the application server module visibility to application

```
# Setting Module visibility to Application
# number means:
#   0 Compatibility
#   1 Application
#   2 Server
#   3 Module
#
```

```
set moduleattr [list ModuleVisibility 1]
set attributelist [list $moduleattr]

ApplicationServer modify /Node:itsohost/ApplicationServer:WebbankServer01/
-attribute $attributelist
```

For further information about module visibility and classloaders, please see Chapter 19, “Deploying an application” on page 687.

Configure session manager

To construct the attribute list and modify the application server session manager for session persistence, use the commands in Example 23-15. This will enable session persistence so you can cluster your environment. The tuning settings used are equivalent to selecting the **Very high** performance option in the administrative console.

Example 23-15 Modifying the application server for session persistence

```
set perenat [list EnablePersistentSessions true]
set perdsat [list PersistentDatasourceName WebbankSessions]
set siat [list TuningScheduleInvalidation true]
set titoat [list TuningInvalidationTimeout 30]
set wcontat [list TuningWriteContents 0]
set wfreqat [list TuningWriteFrequency 2]
set wintat [list TuningWriteInterval 300]
set sesslist [list $perenat $perdsat $siat $titoat $wcontat $wfreqat $wintat]
set sessmanattr [list SessionManagerConfig $sesslist]
set webcontainerattr [list WebContainerConfig $sessmanattr]
set attributelist [list $webcontainerattr]

ApplicationServer modify /Node:itsohost/ApplicationServer:WebbankServer01/
-attribute $attributelist
```

Notice that we used the previously created data source for the PersistentDatasourceName attribute.

Note: With form login, set EnablePersistentSessions to true only if you are using LTPA authentication mechanism. Persistent sessions with form login will not work with local operating system authentication mechanism.

23.2.6 Creating the enterprise application

You can use the WebSphere Control Program to deploy a complete enterprise application or to deploy application modules. In this section we use the EnterpriseApp `install` action to install our Webbank enterprise application. We also provide a brief example of how you can use the Module `install` action to deploy a Web, EJB, or client application module.

Installing an enterprise application

To construct the attribute list, create, and install the enterprise application, use the commands shown in Example 23-16. If the deployment code is not already generated, it will be generated now (use the `-redeploy` option to force). At deployment (installation) time you can overwrite bindings that were set at application assembly time, as shown here.

See “Creating an enterprise application” in the InfoCenter for a full description of the EnterpriseApp install options.

Example 23-16 Installing the enterprise application overwriting assembly bindings

```
# first, construct -moduleappservers option
#     module installation destination
#     it is possible to spread modules over multiple servers
#     for instance to separate web modules and ejb modules
set nodeattr /Node:itsohost/ApplicationServer:WebbankServer01/
set warmodserv1 [list webbankWeb.war $nodeattr]
set ejbmodserv [list webbankEJBs.jar $nodeattr]
set modappservers [list $ejbmodserv $warmodserv1]

# second, construct -ejbreferences option
#     this can also be defined in the deployment descriptor using the AAT
set ejbref1 [list webbankEJBs.jar::Transfer::ejb/BranchAccount
webbank/BranchAccount]
set ejbref2 [list webbankEJBs.jar::Transfer::ejb/CustomerAccount
webbank/CustomerAccount]
set ejbref3 [list webbankEJBs.jar::Consultation::ejb/BranchAccount
webbank/BranchAccount]
set ejbref4 [list webbankEJBs.jar::Consultation::ejb/CustomerAccount
webbank/CustomerAccount]
set ejbref5 [list webbankWeb.war::ejb/Transfer webbank/Transfer]
set ejbrefs [list $ejbref1 $ejbref2 $ejbref3 $ejbref4 $ejbref5]

# third, construct -ejbnames option
#     this can be also defined in the deployment descriptor using the AAT
set ejbname1 [list webbankEJBs.jar::BranchAccount webbank/BranchAccount]
set ejbname2 [list webbankEJBs.jar::CustomerAccount webbank/CustomerAccount]
set ejbname3 [list webbankEJBs.jar::Transfer webbank/Transfer]
set ejbname4 [list webbankEJBs.jar::Consultation webbank/Consultation]
```

```
set ejbnames [list $ejbname1 $ejbname2 $ejbname3 $ejbname4]

# fourth, construct -cmpdatasources option
#     this can also be defined in the deployment descriptor using the AAT
set cmpdatasource1 [list webbankEJBs.jar::BranchAccount jdbc/webbank]
set cmpdatasource2 [list webbankEJBs.jar::CustomerAccount jdbc/webbank]
set cmpdatasources [list $cmpdatasource1 $cmpdatasource2]

# fifth, construct -modvirtualhosts option
set modvirthost1 [list webbankWeb.war webbank_vhost]
set modvirthosts [list $modvirthost1]

# installing the EnterpriseApp
set defappserver /Node:itsohost/ApplicationServer:WebbankServer01/

EnterpriseApp install /Node:itsohost/
D:/WebSphere/AppServer/installableApps/webbank_deployed.ear -appname Webbank
-modvirtualhosts $modvirthosts -ejbreferences $ejbrefs -ejbnames $ejbnames
-cmpdatasources $cmpdatasources -moduleappservers $modappservers
```

In Example 23-16 the data source bindings were made per bean. However, you can also make the binding for the whole module as shown in Example 23-17 using the `-ejbdatasources` option of the `install` action.

Example 23-17 Data source binding to the module

```
...
set ejbdatasource1 [list webbankEJBs.jar jdbc/webbank]
set ejbdatasources [list $ejbdatasource1]

...
EnterpriseApp install /Node:itsohost/
D:/WebSphere/AppServer/installableApps/webbank_deployed.ear -appname Webbank
-modvirtualhosts $modvirthosts -ejbreferences $ejbrefs -ejbnames $ejbnames
-moduleappservers $modappservers -ejbdatasources $ejbdatasources
```

If there is no need to overwrite some or all the bindings, then use the `install` action without these options. Example 23-18 shows how to install with the minimum set of options, using the bindings set at assembly time.

Example 23-18 Installing the enterprise application using assembly bindings

```
# ... the settings are left out ...
EnterpriseApp install /Node:itsohost/
D:/WebSphere/AppServer/installableApps/webbank_deployed.ear -appname Webbank
-defappserver $defappserver
```

Installing a module

The WSCP Module actions allow you to deploy modules independently from enterprise applications. Application client and EJB modules are installed from Java archive (JAR) files, and Web modules are installed from Web archive (WAR) files. When you install a module, WSCP actually generates a simple EAR file with the specified JAR or WAR file as its only module. It then installs the EAR file. You can optionally assign such things as security roles, specify JNDI mappings, and specify data sources for enterprise beans.

Example 23-19 shows how to install a Web module. The WAR file containing the module is D:/storefront.war, and the context root is /WebApp.

Example 23-19 Installing the module using assembly bindings

```
Module install /Node:itsohost/ D:/storefront.war -contextroot /WebApp
```

See “Creating an enterprise application” in the InfoCenter for a full description of the Module install options.

23.2.7 Regenerate Web server plug-in

After creating new objects and modifying settings that are related to the Web server interface (host names, ports, URIs, URL), it is mandatory to regenerate the Web server plug-in configuration. See Chapter 14, “Configuring the Web server interface” on page 481 for details.

Use the following command either in interactive mode or as part of the script:

```
wscp> Node regenPluginCfg /Node:itsohost/
```

23.2.8 Starting and stopping the enterprise application

To start and test the newly installed enterprise application, do the following:

1. Start the enterprise application from within WSCP:

   ```
   wscp> EnterpriseApp start /EnterpriseApp:Webbank/
   ```

2. Use the following URL in your browser:

   ```
   http://www.webbank.itso.ibm.com:90/webbank/
   ```

To stop the enterprise application from within WSCP, use the following command:

```
wscp> EnterpriseApp stop /EnterpriseApp:Webbank/
```

23.2.9 Creating the server group and the second clone

To create a server group based on the application server created in 23.2.5, “Creating the application server” on page 914, use the commands shown in Example 23-20. The `-baseInstance` specifies the application server that is used as a template for the server group. In addition, this application server becomes the first clone in the new server group.

Example 23-20 Creating a server group

```
# creating ServerGroup from the existing application server
ServerGroup create /ServerGroup:WebbankSG/ -baseInstance
/Node:itsohost/ApplicationServer:WebbankServer01/

# creating a second clone
set clonenameattr [list Name WebbankServer02]
set attributelist [list $clonenameattr]

ServerGroup clone /ServerGroup:WebbankSG/ -node /Node:itsohost/ -cloneAttrs
$attributelist
```

The second clone is automatically assigned an HTTP port for the Web server plug-in.

23.2.10 Starting and stopping the server group

Start the server group to bring up both clones simultaneously by using the following command:

```
wscp> ServerGroup start /ServerGroup:WebbankSG/
```

Stop the server group to bring down both clones at the same time by using the following command:

```
wscp> ServerGroup start /ServerGroup:WebbankSG/
```

Use the ApplicationServer `start` or `stop` actions to start or stop the clones separately.

23.2.11 Enabling security

To secure your environment, you need to enable global security. We will use the local operating system (LOCALOS) as the authentication mechanism. You also need use the `-userid` option to specify an administrative user ID and password. Use the following commands to enable and configure security:

```
wscp> SecurityConfig setAuthenticationMechanism LOCALOS -userid {wasadmin
wasadmin}
wscp> SecurityConfig enableSecurity
```

After running these commands, you need to restart the administrative server in order for the changes to take effect.

Stop the node (administrative server) using the command:

```
wscp> Node stop /Node:itsohost/
```

Start the node as explained in 13.2.1, “Starting and stopping a node” on page 459. When starting WSCP after the administrative server has started, a user ID and password challenge window pops up. You have to supply the administrative user ID and password to log on to the administrative server (for instance, wasadmin and wasadmin).

23.2.12 Configuring security for the enterprise application

To map individual users, groups, or special groups to roles defined during development and application assembly time, use the commands shown in Example 23-21. To map individual users and groups to roles, you need to define the needed users and groups in your user repository beforehand.

Our example maps all roles to the special subject AllAuthenticatedUsers. This allows everyone who has a valid user ID and password in the user repository to use the application. To make the role mapping, use the command syntax:

```
SecurityRoleAssignment <action> <entappname> {<role> <user | group>}
```

Example 23-21 Mapping roles to special group

```
set specialrole1 [list Manager AllAuthenticatedUsers]
set specialrole2 [list Employee AllAuthenticatedUsers]
set specialrole3 [list AllAuthenticated AllAuthenticatedUsers]
set specialrole4 [list Everyone AllAuthenticatedUsers]
set specialroles [list $specialrole1 $specialrole2 $specialrole3 $specialrole4]
set specialrolesattr [list $specialroles]

SecurityRoleAssignment addSpecialRoleMapping /EnterpriseApp:Webbank/
-specialroles $specialroles}
```

23.2.13 Creating the whole environment in one shot

Use the script createWebbankEnv.tcl (see Appendix D, “Additional material” on page 1083) to create the whole Webbank environment in one shot. Modify the values at the top of the file to match your environment, then use the following command to apply the configuration:

```
wscp -f createWebbankEnv.tcl
```

You should now be able to start the Webbank enterprise application as described in 23.2.8, “Starting and stopping the enterprise application” on page 919.

23.2.14 Deleting the whole environment in one shot

You may want a script to delete an application environment so you can back out a failed installation, or to prepare for the installation of a new application release.

To delete the whole Webbank environment except for the databases, use the WSCP script deleteWebbankEnv.tcl (see Appendix D, “Additional material” on page 1083). Make sure that the server group is stopped, as described in 23.2.10, “Starting and stopping the server group” on page 920, then invoke the script as follows:

```
wscp -f deleteWebbankEnv.tcl
```

23.3 Ripple mode using WSCP

In an environment with multiple server groups and/or multiple application servers per server group (designed for 24x7), you need a mechanism to make changes to your configuration without having a service outage. One way to do this automatically is to use a WSCP script that implements a ripple mode to sequentially stop and restart each application server defined in the server group.

The script in Example 23-22 can be used to do this. The script does not check the current state of the application server. It tries to stop the clone regardless of its state. As a result, an exception is thrown when the current state of the clone is stopped. To avoid exiting the script as a result, a catch statement can be used to catch this exception. You may wish to improve this script by checking the current state of a clone and restart (stop and start) it only if the clone is in a running state.

Example 23-22 Ripple mode WSCP script

```
set servergroup WebbankSG

puts "Start searching for clones in $servergroup..."
foreach server [ServerGroup listClones /ServerGroup:$servergroup/] {
   puts "Found clone $server"
   puts "Stopping clone $server ..."
   catch {ApplicationServer stop $server}

   puts "Starting clone $server ..."
   catch {ApplicationServer start $server}

   puts "Clone $server ready.\n"
}
```

Use the following command to start the script:

```
wscp -f ripplemode.tcl
```

The output of ripplemode.tcl for our Webbank environment is:

```
Start searching for clones in WebbankSG...
Found clone /Node:itsohost/ApplicationServer:WebbankServer02/
Stopping clone /Node:itsohost/ApplicationServer:WebbankServer02/ ...
Starting clone /Node:itsohost/ApplicationServer:WebbankServer02/ ...
Clone /Node:itsohost/ApplicationServer:WebbankServer02/ ready.

Found clone /Node:itsohost/ApplicationServer:WebbankServer01/
Stopping clone /Node:itsohost/ApplicationServer:WebbankServer01/ ...
Starting clone /Node:itsohost/ApplicationServer:WebbankServer01/ ...
Clone /Node:itsohost/ApplicationServer:WebbankServer01/ ready.
```

See Appendix D, “Additional material” on page 1083 to obtain the ripplemode.tcl script.

Note: Support for ripple mode of a server group via the administrative console is being considered for a future release of WebSphere V4.0.

23.4 Introducing XMLConfig

The XMLConfig tool allows you to interact with the WebSphere administrative repository, so you can modify or extract configuration data. XMLConfig makes full use of the eXtensible Markup Language (XML), adhering to grammar and hierarchical conventions.

With XML documents coded to the WebSphere Application Server Configuration Markup Language syntax (WASCML), it is possible to invoke an action on a specified resource. Starting or stopping an application server is a common example.

23.4.1 XML and WebSphere

Those not familiar with XML should recognize that it considerably contrasts with other markup languages, such as HTML. XML offers the user the scope to expand or enhance the actual markup tags of a document. Unlike HTML, where a fixed set of standardized function tags are employed, this enhancement or design freedom does not lie with the simple creation of an XML document.

Rather, it requires the design and construction of a Document Type Definition (DTD) file to actually specify the underlying XML syntax and structure. Subsequent XML documents can then be coded, adhering to the constraints and validity set forth in the DTD specification.

XML has become a standard Web-based technology for transferring data prior to presentation at the client, and WebSphere now ships with over 100 DTD catalogs. However, with XMLConfig we are only concerned with the DTD xmlconfig.dtd found in the <WAS_HOME>/bin directory. It is this file that maintains the XML specifications, governing structure and syntax when invoking XMLConfig functions on the WebSphere administrative database.

XML in part is a hierarchical-based metalanguage, embracing elements, entities and attributes in a logical fashion. This is also true of WebSphere, which utilizes the concept of an object for each resource found under the administrative domain. Visually, the WebSphere Administrative Console best demonstrates this concept, with the WebSphere administrative domain being at the top of the hierarchy with nodes, application servers, and Web modules successively lower in the hierarchy.

Using the WebSphere Administrative Console topology view quickly enables the administrator to recognize the differing elements and attributes that constitute the component parts of WebSphere. Familiarization with this structure or hierarchy is paramount when editing XML documents, as it enables the WebSphere administrator to quickly identify an object and understand the corresponding attributes.

In conclusion, XMLConfig offers an alternative method for configuring and administrating WebSphere. However, it does not directly replace the administrative console or the WebSphere Control Program.

23.4.2 When to use XMLConfig

You can use XMLConfig tool for the following tasks:

- Export and import your WebSphere configuration
- Make multiple changes to your WebSphere configuration without the console
- Start and stop resources

XMLConfig is mainly designed to support automated tasks, such as changing multiple resources in the WebSphere configuration. It is very useful and powerful for creating a whole environment and for installing multiple enterprise applications in one action.

Although you can start and stop resources with XMLConfig, other tools such as WSCP are more suitable for this job. The main reason for not using XMLConfig for such simple tasks as starting and stopping resources is that you have to write an XML file to describe the action you want to do. For example, to start an application server you need the XML file shown in Example 23-23.

Example 23-23 startAppServer.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <node action="locate" name="itsohost">
      <application-server action="start" name="Default Server">
      </application-server>
   </node>
</websphere-sa-config>
```

And you need the following command to import the XML file:

```
XMLConfig -import startAppServer.xml -adminNodeName itsohost
```

The appropriate wscp command for this task is as follows:

```
wscp -c "ApplicationServer start {/Node:itsohost/ApplicationServer:Default
Server/}"
```

23.4.3 Invoking XMLConfig

XMLConfig can be started from the command line on Windows platforms by running:

```
<WAS_HOME>\bin\XMLConfig.bat
```

On UNIX platforms, use:

```
<WAS_HOME>/bin/XMLConfig.sh
```

If the <WAS_HOME>\bin directory is in your PATH, then you can invoke XMLConfig without including this directory.

Command-line options

The following command-line options are accepted by XMLConfig:

```
XMLConfig
        {[( -import <xml data file> ) ||
        ( -export <xml output file> [-partial <xml data file>] )]
        -adminNodeName <primary node name>
        [ -nameServiceHost <host name> [ -nameServicePort <port number> ]]
        [-traceString <trace spec> [-traceFile <file name>]]
```

```
[-generatePluginCfg <true || false>.]
[-substitute <"key1=value1[;key2=value2;[...]]">]}
```

Where:

`-adminNodeName`	Required argument that specifies the node containing the administrative server. This must match the node name of your configuration.
`-export file`	Required argument that specifies the export operation to perform. Unless you also specify the parameter `-partial`, the export is treated as a full export.
`-partial file`	Takes the XML file as argument that specifies which parts should be exported.
`-import file`	Required argument that specifies the export operation to perform.
`-generatePluginCfg`	If set to true, it generates the plug-in configuration if necessary.
`-nameServiceHost, -nameServicePort`	Optional arguments that specify the host name of the machine that contains the naming service, and the port through which to communicate with the naming service. The default value of `-nameServicePort` is 900.
`-substitute list`	Optional argument that specifies the variable-key pairs to be substituted. In input XML file, the key(s) should appear as key for substitution. Multiple pairs are separated by `;` (semicolon).
`-traceString string`	Optional argument that specifies the internal code to trace. For further information see Chapter 24, “Troubleshooting” on page 949.
`-traceFile file`	Specifies the file where the trace information goes.

Variable substitution

You can create generic XML files to handle your WebSphere configuration using the substitute operation of XMLConfig. This allows you to specify place holders (variables) in your XML files instead of the real names.

For instance, if you want to change the JVM heap size for an application server, your XML file should look like Example 23-24.

Example 23-24 appServerSetJVMHeap.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <node action="update" name="$node$">
      <application-server action="update" name="$server$">
         <jvm-config>
            <initial-heap-size>$initheap$</initial-heap-size>
            <max-heap-size>$maxheap$</max-heap-size>
         </jvm-config>
      </application-server>
   </node>
</websphere-sa-config>
```

Use the following command to import new values (128M and 256M) for the JVM heap into the Default Server on node itsohost:

```
XMLConfig -import appServerSetJVMHeap.xml -adminNodeName itsohost
-substitute "node=itsohost;server=Default Server;initheap=128;maxheap=256"
```

You can use the following variables provided in XMLConfig to create platform-neutral XML documents:

- `$server_root$`

  Replaced with the product installation directory, such as D:\WebSphere\AppServer on Windows (not available on iSeries).

- `$psep$`

  Replaced with the path separator as specified in the operating system JDK. On Windows it is ; (semicolon), on UNIX it is : (colon).

- `$dsep$`

  Replaced with the directory separator as specified in the operating system JDK. On Windows it is \ (backward slash), on UNIX it is / (forward slash).

Partial export

To export one or more parts of the WebSphere configuration, rather than the full configuration, use the `-partial` option. You need to provide an XML file that includes all the parts to extract. An example is shown in Example 23-25. It exports the application server and enterprise applications but not the security configuration.

Example 23-25 expAppServEntApp.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <node action="locate" name="itsohost">
      <application-server action="export" name="Default Server">
      </application-server>
   </node>
   <enterprise-application action="export" name="itsohost_sampleApp">
   </enterprise-application>
</websphere-sa-config>
```

Use the following command to make the partial export:

```
XMLConfig -export partialExport.xml -adminNodeName itsohost -partial
expAppServEntApp.xml
```

What happens if security is enabled?

With global security enabled in your WebSphere administrative domain, you have to authenticate to your administrative server before performing operations. There are two ways to authenticate against the administrative server:

- Interactively enter the user ID and password each time you start XMLConfig.
- Set login properties in the sas.client.props file as follows:
  a. Open <WAS_HOME>\properties\sas.client.props for editing.
  b. Set the following properties:

     ```
     com.ibm.CORBA.loginSource=properties
     com.ibm.CORBA.loginUserid=<your_admin_userid>
     com.ibm.CORBA.loginPassword=<your_admin_password>
     ```

  c. Save your changes.

Note: The <WAS_HOME>\properties\sas.client.props file is shared with other local clients by default. This means that login properties specified here will be used by other clients, such as the administrative console and J2EE client applications.

23.4.4 Working with XML files for XMLConfig

You can use any text editor to create or modify XML files for XMLConfig. However, it is difficult and error prone to create the required XML code from scratch. A better way is to make a full export of an existing configuration and modify it in the way you need.

To get a full export of your WebSphere administrative domain configuration, use the following command:

```
XMLConfig -export wasconfig.xml -adminNodeName itsohost
```

XML schema for XMLConfig

Common to all XML is the schema or syntax that adheres to the specifications defined in the DTD. WebSphere XML is no exception to this rule, being valid only if it adheres to the constraints stipulated in the xmlconfig.dtd file.

At the simplest level, XML elements are constructed in a hierarchical or tree-like structure. Indeed, each element is defined by a tag and associated end tag. This concept is further extended since elements can be nested, forming the various branches of a tree. Each element can also have an associated attribute, which in turn further describes the element. For example:

```
<websphere-sa-config>
   <parent name="pname" action="valid-action">
      <child name="cname" action="valid-action"/>
   </parent>
</websphere-sa-config>
```

It is not surprising that the XML representation of the WebSphere topology tree is similar to that used to navigate the administrative console topology tree. This is evident in Example 23-26, which performs a `create` action on the application server “newas01”, but only after first locating the respective node.

Example 23-26 XML hierarchy to create application server

```
<websphere-sa-config>
   <node name="itsohost" action="locate">
      <application-server name="newas01" action="create">
         ...
         ...
      </application-server>
   </node>
</websphere-sa-config>
```

The `locate` action is used to select the specified element and effectively serves to guide us to the next nested object. Once the node itsohost is located, we can create an application server called newas01. The actual creation details have been omitted to save space.

XMLConfig elements

Throughout the examples provided here, you will see some common elements that never change:

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   ...
</websphere-sa-config>
```

Here, the first line specifies the XML version and the second line the Document Type Definition (DTD) file that governs the XML syntax and structural validity of any following XML markup. Since WebSphere is a cross-platform product, variable substitution will automatically supplement the correct values for the variables. See “Variable substitution” on page 926 for details.

XML elements are defined using tags. Each element has a start and end tag, with the end tag prefixed with a “/” forward slash. Example 23-27 highlights the validity of nested tags, only modifying the jvm-config once the node and application-server are correctly located. The actual jvm-config specifics have been omitted in this case.

Example 23-27 Nested XML elements

```
<websphere-sa-config>
   <node action="locate" name="itsohost">
      <application-server action="update" name="Default Server">
         <jvm-config>
            ...
         </jvm-config>
      </application-server>
   </node>
</websphere-sa-config>
```

Each element requires a unique name to distinguish itself from any peers at the same level in the topology tree. However, since each branch of the topology tree is independent, unique names are not so critical towards the ends of the branches.

XMLConfig actions

Example 23-27 on page 930 also illustrates the concept of element actions, with locate and update briefly introduced to configure the application server jvm-config, shown in bold type. Valid element actions are listed in Table 23-3.

Table 23-3 XML actions

Action	What it does
`create`	Adds the specified resource to the administrative domain. If the resource already exists, the `create` action is treated as an update action. When using this action, you must specify all required attributes for the object type.

Action	What it does
`update`	Updates the properties of a specified resource. You need only specify the properties that you want to update. However, if the resource does not exist, the `update` action becomes a create. In such a case, if some of the properties are not specified, an error occurs.
`delete`	Removes the specified resource and its children (recursive delete).
`locate`	Locates the specified resource. Use this action to provide the containment path to specific child resources. Applies to the partial export operation as well as the import operation.
`export`	Exports the configuration of the specified resource and its children to the output XML document. Applies only to the partial export operation.
`start`	Starts the specified resource (if applicable).
`stop`	Stops the specified resource (if applicable).
`stopforrestart`	Stops the specified resource, and restarts it (if applicable). Node only.
`restart`	Stops the specified resource and starts it again (if applicable).

Not all actions are valid for all types of elements. There are two possible ways to determine which attributes are valid for each respective element or what elements are valid at the various levels in the XML hierarchy:

- Consult the xmlconfig.dtd DTD found in the <WAS_HOME>/bin directory for the definitive answer.
- If a similar element or object in WebSphere terminology already exists in your WebSphere configuration, an XMLConfig export can be consulted.

Another very good reference is the com.ibm.websphere.xmlconfig package under Product Javadoc in the InfoCenter.

Note: All element names in XML are both case-sensitive and white-space-sensitive; for example, “Default Server” is not the same as “default server”, nor is “Default Server” the same as “DefaultServer”.

In our XMLConfig examples, new lines start with a < and terminate with a carriage return only after a > or />.

Now let's look at the main tasks that can be performed with XML files and the XMLConfig tool. For a complete example of how to create a new environment and install an enterprise application, please see 23.5, "Creating your runtime environment using XMLConfig" on page 935.

23.4.5 Creating resources

To create a new resource in your WebSphere configuration, use the `create` action. For instance, Example 23-28 shows how to create a new application server with an initial JVM heap size of 128 MB and maximum JVM heap size of 256 MB.

Example 23-28 createAppServer.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <node action="locate" name="itsohost">
      <application-server action="create" name="newas01">
         <jvm-config>
            <initial-heap-size>128</initial-heap-size>
            <max-heap-size>256</max-heap-size>
         </jvm-config>
      </application-server>
   </node>
</websphere-sa-config>
```

Use the following command to create the application server:

```
XMLConfig -import createAppServer.xml -adminNodeName itsohost
```

23.4.6 Updating resources

To modify resources in your WebSphere configuration use the `update` action. For instance, Example 23-29 shows how to update an existing data source with new timeout values.

Example 23-29 updateDataSource.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <jdbc-driver action="locate" name="Sample DB Driver">
      <data-source action="update" name="sample">
         <connection-timeout>10</connection-timeout>
         <idle-timeout>600</idle-timeout>
         <orphan-timeout>900</orphan-timeout>
```

```
      </data-source>
   </jdbc-driver>
</websphere-sa-config>
```

Use the following command to update the data source:

```
XMLConfig -import updateDataSource.xml -adminNodeName itsohost
```

23.4.7 Deleting resources

To delete a resource from your WebSphere configuration use the `delete` action. For instance, Example 23-30 shows how to remove an application server.

Example 23-30 deleteAppServer.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <node action="locate" name="$node$">
      <application-server action="delete" name="$server$">
      </application-server>
   </node>
</websphere-sa-config>
```

Use the following command to delete the application server:

```
XMLConfig -import deleteAppServer.xml -adminNodeName itsohost -substitute
"node=itsohost;server=newas01"
```

This example makes use of variable substitution, as described in “Variable substitution” on page 926.

23.4.8 Installing an enterprise application

A J2EE enterprise application needs to be packaged as an enterprise application archive (EAR). Once you have created your enterprise application archive, you can install (or deploy) it into your environment. Application packaging is explained in Chapter 18, “Packaging an application” on page 639.

Example 23-31 shows how to install the performance monitoring servlet application, which is shipped with WebSphere V4.0 but not installed by default (some of the long lines are wrapped). Note that we use variable subsitution in the ear-file-name and ear-install-directory. See “Variable substitution” on page 926 for details.

Example 23-31 createEntApp.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <enterprise-application action="create" name="perfServletApp">
      <source-node>itsohost</source-node>

<ear-file-name>$server_root$$dsep$installableApps$dsep$perfServletApp.ear</ear-
file-name>
      <enterprise-app-install-info>
         <node-name>itsohost</node-name>

<ear-install-directory>$server_root$$dsep$installedApps$dsep$perfServletApp.ear
</ear-install-directory>
      </enterprise-app-install-info>
      <web-module name="perfservlet">
         <war-file>perfServletApp.war</war-file>
         <context-root>/wasPerfTool</context-root>
         <module-install-info>

<application-server-full-name>/NodeHome:itsohost/EJBServerHome:Default
Server/</application-server-full-name>
         </module-install-info>
         <web-module-binding>
            <virtual-host-name>default_host</virtual-host-name>
         </web-module-binding>
      </web-module>
   </enterprise-application>
</websphere-sa-config>
```

Make sure that all the needed resources (such as the application server, virtual host, and referenced directories) are created before you install an enterprise application.

To install the enterprise application using XMLConfig, use the following command:

```
XMLConfig -import createEntApp.xml -adminNodeName itsohost
```

23.4.9 Starting and stopping resources

To start or stop a resource in your WebSphere administrative domain, use the start and stop actions. For instance, you can start and stop an application server using Example 23-32. This example also makes use of variable substitution, as explained in “Variable substitution” on page 926. Only one XML file is needed to start, stop and even create an application server.

Example 23-32 actionAppServer.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <node action="locate" name="$node$">
      <application-server action="$action$" name="$server$">
      </application-server>
   </node>
</websphere-sa-config>
```

Start the application server with the following command:

```
XMLConfig -import actionAppServer.xml -adminNodeName itsohost -substitute
"node=itsohost;server=Default Server;action=start"
```

Stop the application server with the following command:

```
XMLConfig -import actionAppServer.xml -adminNodeName itsohost -substitute
"node=itsohost;server=Default Server;action=stop"
```

For more examples, please see 23.5, “Creating your runtime environment using XMLConfig” on page 935.

23.5 Creating your runtime environment using XMLConfig

In this section we explain how to use XMLConfig to create, configure, and install all the bits and pieces needed to run the Webbank sample application (see Appendix D, “Additional material” on page 1083 to obtain webbank.ear). The XMLConfig documents and scripts used here are also included in the additional material.

Note: In some of the examples, the longer lines are wrapped because they are too wide for the page.

23.5.1 Assumptions

The assumptions are the same as for the WSCP scenario. See 23.2.1, “Assumptions” on page 911 for the list.

23.5.2 Creating the virtual host

Example 23-33 shows the part of our XMLConfig file that creates the virtual host. Note that to save space we only show one of the entries in the MIME Types list in the example. See Appendix D, “Additional material” on page 1083 to obtain the full listing of createWebbankEnv.xml.

Example 23-33 Creating the virtual host

```
...
   <virtual-host action="update" name="webbank_vhost">
      <mime-table>
         ...
         <mime type="text/html">
            <ext>htm</ext>
            <ext>html</ext>
         </mime>
         ...
      </mime-table>
      <alias-list>
         <alias>www.webbank.itso.ibm.com:90</alias>
         <alias>www.webbank.itso.ibm.com:443</alias>
         <alias>www.webbank.itso.ibm.com:9090</alias>
         <alias>www.webbank.itso.ibm.com:9091</alias>
      </alias-list>
   </virtual-host>
...
```

With these settings the application server will accept URLs connecting to hostname `www.webbank.itso.ibm.com` on port 90, 9090, 9091, and 443 (for HTTPS). If you do not specify the MIME table, then the default MIME table will be created.

23.5.3 Creating the JDBC driver and data sources

Example 23-34 shows the part of our XMLConfig file that creates the JDBC driver and data sources. Our example uses two data sources, one for the CMP beans and one for session persistence. See Appendix D, “Additional material” on page 1083 to obtain the full listing of createWebbankEnv.xml.

Example 23-34 Creating the JDBC driver and data sources

```
...
   <jdbc-driver action="update" name="WebbankDB2Driver">

<implementation-class>COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource</implementat
ion-class>
      <description></description>
      <data-source action="update" name="WebbankDS">
```

```
            <database-name>WEBBANK</database-name>
            <minimum-pool-size>1</minimum-pool-size>
            <maximum-pool-size>10</maximum-pool-size>
            <connection-timeout>180</connection-timeout>
            <idle-timeout>1800</idle-timeout>
            <orphan-timeout>1800</orphan-timeout>
            <statement-cache-size>100</statement-cache-size>
            <default-user>db2admin</default-user>
            <default-password>db2admin</default-password>

<disable-auto-connectioncleanup>false</disable-auto-connectioncleanup>
            <description></description>
            <jndi-name>jdbc/webbank</jndi-name>
            <config-properties/>
         </data-source>
         <data-source action="update" name="WebbankSessions">
            <database-name>SESSIONS</database-name>
            <minimum-pool-size>1</minimum-pool-size>
            <maximum-pool-size>10</maximum-pool-size>
            <connection-timeout>180</connection-timeout>
            <idle-timeout>1800</idle-timeout>
            <orphan-timeout>1800</orphan-timeout>
            <statement-cache-size>100</statement-cache-size>
            <default-user>db2admin</default-user>
            <default-password>db2admin</default-password>

<disable-auto-connectioncleanup>false</disable-auto-connectioncleanup>
            <description></description>
            <jndi-name>jdbc/webbanksessions</jndi-name>
            <config-properties/>
         </data-source>
         <install-info>
            <node-name>$node$</node-name>

<jdbc-zipfile-location>D:\SQLLIB\java\db2java.zip;</jdbc-zipfile-location>
         </install-info>
      </jdbc-driver>
...
```

Passwords specified in clear text in the XML file will be encrypted during import. Make sure you protect the XML file itself in an appropriate way.

Important: Make sure you have created your application database, as advised in 23.2.1, “Assumptions” on page 911. For the Webbank application, see 19.5, “Creating and populating the Webbank database” on page 705.

23.5.4 Creating the server group and clones

Example 23-34 shows the part of our XMLConfig file that creates the server group and clones. To save space, we have left out most of the details of the example listing. See Appendix D, “Additional material” on page 1083 to obtain the full listing of createWebbankEnv.xml.

Example 23-35 Creating the server group

```
...
   <server-group action="update" name="WebbankSG">
      <server-group-attributes>
         ...
         <module-visibility>1</module-visibility>
         <web-container>
            ...
            <session-manager>
               ...
               <write-contents>0</write-contents>
               <write-frequency>0</write-frequency>
               <write-interval>0</write-interval>
               ...
               <persistent-sessions>true</persistent-sessions>
               <data-source name="WebbankSessions">
                  <default-user></default-user>
                  <default-password></default-password>
               </data-source>
               ...
            </session-manager>
         </web-container>
         <orb-config>
            ...
         </orb-config>
         <custom-service-config-list>
            ...
         </custom-service-config-list>
      </server-group-attributes>
      <clone name="WebbankServer01">
         <parent-node>$node$</parent-node>
      </clone>
      <clone name="WebbankServer02">
         <parent-node>$node$</parent-node>
      </clone>
   </server-group>
...
```

Important points to understand when creating the server group and clones are:

- XMLConfig allows you to create a server group and its clones in one action. With the administrative console or with WSCP you need to use separate steps to first create the application server, create a server group from the application server, and finally create clones.
- Our Webbank application requires application server module-visibility set to 1 (or “Application”). The XMLConfig module visibility setting uses the same numbering scheme as WSCP, described in “Changing module visibility” on page 915.
- Our example configures the session manager service for session persistence. The tuning settings used are equivalent to selecting the **Medium** option in the administrative console.

  Note: With form login, only set persistent-sessions to true if you are using LTPA authentication mechanism. Persistent sessions with form login will not work with local operating system authentication mechanism.

- Our example creates two clones, WebbankServer01 and WebbankServer01. These clones inherit the server-group-attributes. Some attributes can be separately modified for individual clones. We modify the clones’ transport-port and log files in 23.5.7, “Updating clone attributes” on page 941.

23.5.5 Creating the enterprise application

Example 23-34 shows the part of our XMLConfig file that creates the Webbank enterprise application. This application is created from the webbank_deployed.ear enterprise application archive. Note that to save space we do not show the ejb-module-binding information in the example. See Appendix D, “Additional material” on page 1083 to obtain the full listing of createWebbankEnv.xml.

Example 23-36 Installing the enterprise application

```
...
   <enterprise-application action="create" name="Webbank">
      <source-node>$node$</source-node>

<ear-file-name>$server_root$$dsep$installableApps$dsep$webbank_deployed.ear</ea
r-file-name>
      <enterprise-app-install-info>
         <node-name>$node$</node-name>

<ear-install-directory>$server_root$$dsep$installedApps$dsep$Webbank.ear</ear-i
nstall-directory>
      </enterprise-app-install-info>
      <application-binding>
```

```
<!-- use role mappings set in the ear file
         <authorization-table>
            ...
         </authorization-table>
-->
         <run-as-map/>
      </application-binding>
      <ejb-module name="webbankEJBs">
         <jar-file>webbankEJBs.jar</jar-file>
         <module-install-info>
            <server-group-name>WebbankSG</server-group-name>
         </module-install-info>
         <ejb-module-binding>
            ...
         </ejb-module-binding>
      </ejb-module>
      <web-module name="webbankWeb">
         <war-file>webbankWeb.war</war-file>
         <context-root>/webbank</context-root>
         <module-install-info>
            <server-group-name>WebbankSG</server-group-name>
         </module-install-info>
         <web-module-binding>
            <virtual-host-name>webbank_vhost</virtual-host-name>
            <ejb-ref-binding name="EjbRefBinding_1">
               <jndi-name>webbank/Transfer</jndi-name>
            </ejb-ref-binding>
         </web-module-binding>
      </web-module>
      <web-module name="UTC">
         <war-file>UTC.war</war-file>
         <context-root>/UTC</context-root>
         <module-install-info>
            <server-group-name>WebbankSG</server-group-name>
         </module-install-info>
         <web-module-binding>
            <virtual-host-name>webbank_vhost</virtual-host-name>
         </web-module-binding>
      </web-module>
   </enterprise-application>
...
```

Notice that we do not provide authorization-table security role bindings in the enterprise application-binding section. Our application is created with the security role bindings specified in the deployment descriptor packaged with the enterprise application. See Chapter 18, “Packaging an application” on page 639 for more information.

Note: Our attempts to set security role bindings using XMLConfig failed. It appears that the roles granted using XMLConfig were not recognized, as authorization failed for successfully authenticated users. Our workaround was to not specify security role mappings with XMLConfig, instead using the default mappings provided in the enterprise application archive.

23.5.6 Importing the XMLConfig file

We have described the basic elements used in the createWebbankEnv.xml XMLConfig file that allow us to:

- Create the virtual hosts used for the application
- Create the JDBC driver and data sources needed by the application
- Create the server group and clones that will host the application
- Create and deploy the enterprise application

Now we can import the createWebbankEnv.xml XMLConfig file with the following command:

```
XMLConfig -import createWebbankEnv.xml -adminNodeName itsohost -substitute
"node=itsohost"
```

Some of the attributes of the clones created when importing our XMLConfig file are generated by default. We need to update a few of these attributes next.

23.5.7 Updating clone attributes

We use the setWorkAndPort.xml XMLConfig file given in Example 23-37 to update the working directory name and HTTP transport-port for each of our clones.

Example 23-37 setWorkAndPort.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <node action="update" name="$node$">
      <application-server action="update" name="$server$">
         <working-directory>D:/webbank/$server$</working-directory>
         <web-container>
            <transport name="http">
               <transport-host>*</transport-host>
               <transport-port>$port$</transport-port>
               <http-transport>
                  <connection-timeout>5</connection-timeout>
```

```
                    <backlog-connections>511</backlog-connections>
                    <keep-alive-timeout>5</keep-alive-timeout>
                    <maximum-keep-alive>25</maximum-keep-alive>
                    <maximum-req-keep-alive>100</maximum-req-keep-alive>
                    <ssl-enabled>false</ssl-enabled>
                 </http-transport>
              </transport>
              <thread-maximum-size>50</thread-maximum-size>
              <thread-minimum-size>10</thread-minimum-size>
              <thread-inactivity-timeout>10</thread-inactivity-timeout>
              <thread-is-growable>false</thread-is-growable>
           </web-container>
        </application-server>
     </node>
</websphere-sa-config>
```

Notice that we set individual clone attributes using the application-server XMLConfig element.

We set the WebbankServer01 clone working directory name and HTTP transport-port with the following command:

```
XMLConfig -import setWorkAndPort.xml -adminNodeName itsohost -substitute
"node=itsohost;server=WebbankServer01;port=9090"
```

We set the WebbankServer02 clone working directory name and HTTP transport-port with the following command:

```
XMLConfig -import setWorkAndPort.xml -adminNodeName itsohost -substitute
"node=itsohost;server=WebbankServer02;port=9091"
```

23.5.8 Starting and stopping the server group

You can use the actionServerGrp.xml XMLConfig file given in Example 23-38 to start or stop a server group and its clones.

Example 23-38 actionServerGrp.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <server-group action="$action$" name="$svrgrp$">
   </server-group>
</websphere-sa-config>
```

To start the WebbankSG server group and its clones, use:

```
XMLConfig -import actionServerGrp.xml -adminNodeName itsohost -substitute
"action=start;svrgrp=WebbankSG"
```

To stop the WebbankSG server group and its clones, use:

```
XMLConfig -import actionServerGrp.xml -adminNodeName itsohost -substitute
"action=stop;svrgrp=WebbankSG"
```

23.5.9 Regenerating the Web server plug-in configuration

XMLConfig will regenerate the Web server plug-in configuration after importing a configuration when the `-generatePluginCfg` command-line option is set to true. Start the server group *and* regenerate the plug-in configuration with the following command:

```
XMLConfig -import actionServerGrp.xml -adminNodeName itsohost
-generatePluginCfg true -substitute "action=start;svrgrp=WebbankSG"
```

23.5.10 Enabling security

We use the enableSecurity.xml XMLConfig file given in Example 23-39 to enable global security and set LTPA and IBM SecureWay LDAP directory as the authentication mechanism.

Example 23-39 enableSecurity.xml

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM
"file:///$XMLConfigDTDLocation$$dsep$xmlconfig.dtd" >
<websphere-sa-config>
   <security-config security-cache-timeout="600" security-enabled="$enabled$">
      <app-security-defaults>
         <realm-name>Default</realm-name>
         <challenge-type ssl-enabled="false">
            <basic-challenge/>
         </challenge-type>
      </app-security-defaults>
      <auth-mechanism>
<!-- auth-mechanism can use localos or ltpa-config
         <localos>
            <user-id>wasadmin</user-id>
            <password>wasadmin</password>
         </localos>
-->
         <ltpa-config>
            <ltpa-password>wasadmin</ltpa-password>
            <ltpa-timeout>7200000</ltpa-timeout>
            <sso-enabled ssl-enabled="false">
               <domain-name>itso.ibm.com</domain-name>
```

```
            </sso-enabled>
            <ldap-config type="secureway">
               <user-id>wasadmin</user-id>
               <password>wasadmin</password>
               <ldap-host>itsohost.itso.ibm.com</ldap-host>
               <ldap-port>389</ldap-port>
               <ldap-basedn>o=ibm,c=us</ldap-basedn>
            </ldap-config>
         </ltpa-config>
      </auth-mechanism>
      <ssl-config>
         <key-file-name>${WAS_HOME}/etc/DummyServerKeyFile.jks</key-file-name>
         <key-file-password>WebAS</key-file-password>
         <key-file-format>0</key-file-format>
         <client-authentication>false</client-authentication>
         <security-level>0</security-level>
         <crypto-hardware-enabled>false</crypto-hardware-enabled>
         <crypto-library-file></crypto-library-file>
         <crypto-password></crypto-password>
         <crypto-token-type></crypto-token-type>

<trust-file-name>${WAS_HOME}/etc/DummyServerTrustFile.jks</trust-file-name>
         <trust-file-password>WebAS</trust-file-password>
         <property name="com.ibm.ssl.protocol" value="SSLv3"/>
      </ssl-config>
   </security-config>
</websphere-sa-config>
```

We enabled global security with the following command:

```
XMLConfig -import enableSecurity.xml -adminNodeName itsohost -substitute
"node=itsohost;enabled=true"
```

If you want to try this example, make sure you edit enableSecurity.xml to suit your LDAP server. You will need to change the ldap-host setting as a minimum.

Also note that we could have set up global security in the same XMLConfig file as we used to set up the Webbank application environment. We chose to separate the two because global security settings affect all applications in the administrative domain, not only the Webbank application.

Attention: After importing this XMLConfig file and restarting the administrative server, you will be prompted for your administrative user ID and password before you can do any administrative task using WSCP, WebSphere Administrative Console, or XMLConfig.

23.5.11 Configuring security for the enterprise application

As discussed in 23.5.5, “Creating the enterprise application” on page 939, we decided to create our enterprise application using the security role bindings specified in the deployment descriptor packaged with the enterprise application.

23.5.12 Creating the whole environment in one shot

As we have seen, the steps used to create our Webbank environment are:

1. Create the virtual hosts, JDBC driver and data sources, server group and clones, and the enterprise application:

   ```
   XMLConfig -import createWebbankEnv.xml -adminNodeName itsohost -substitute
   "node=itsohost"
   ```

2. Update each clone’s attributes for working directory name and HTTP transport-port, for clone 1:

   ```
   XMLConfig -import setWorkAndPort.xml -adminNodeName itsohost -substitute
   "node=itsohost;server=WebbankServer01;port=9090"
   ```

   and for clone 2:

   ```
   XMLConfig -import setWorkAndPort.xml -adminNodeName itsohost -substitute
   "node=itsohost;server=WebbankServer02;port=9091"
   ```

3. Start the new server group and regenerate the Web server plug-in configuration:

   ```
   XMLConfig -import actionServerGrp.xml -adminNodeName itsohost
   -generatePluginCfg true -substitute "action=start;svrgrp=WebbankSG"
   ```

4. Enable and configure global security, if required:

   ```
   XMLConfig -import enableSecurity.xml -adminNodeName itsohost -substitute
   "node=itsohost;enabled=true"
   ```

5. Restart the node for the global security changes to take effect:

   ```
   XMLConfig -import restartNode.xml -adminNodeName itsohost -substitute
   "node=itsohost"
   ```

XMLConfig files do not step through a configuration procedure but are instead a snapshot of a WebSphere configuration. XMLConfig allows you to export, edit, and import complete or partial WebSphere configurations. It does not replay the steps in a configuration procedure. If one step must be done after another (such as not starting an application server until after it is created), you need to use a separate XMLConfig import for each step.

You may also prefer to make XMLConfig files more reuseable at the expense of added steps, as with our setWorkAndPort.xml file.

You can use a UNIX shell script or Windows batch file to create a sequence of XMLConfig file imports. We use the createWebbankEnv.bat Windows batch file given in Example 23-40 to create the whole Webbank environment in one shot.

Example 23-40 createWebbankEnv.bat

```
echo "Create Webbank application"
call XMLConfig -import createWebbankEnv.xml -adminNodeName %node% -substitute
"node=%node%"
echo "Set log files and http port for clone 1"
call XMLConfig -import setWorkAndPort.xml -adminNodeName %node% -substitute
"node=%node%;server=WebbankServer01;port=9090"
echo "Set log files and http port for clone 2"
call XMLConfig -import setWorkAndPort.xml -adminNodeName %node% -substitute
"node=%node%;server=WebbankServer02;port=9091"
echo "Start server group and regen plugin"
call XMLConfig -import actionServerGrp.xml -adminNodeName %node%
-generatePluginCfg true -substitute "action=start;svrgrp=WebbankSG"
echo "Enable security"
call XMLConfig -import enableSecurity.xml -adminNodeName %node% -substitute
"node=%node%;enabled=true"
echo "Restart node"
call XMLConfig -import restartNode.xml -adminNodeName %node% -substitute
"node=%node%"
...
```

To create the whole Webbank environment in one shot, use:

```
createWebbankEnv itsohost
```

See Appendix D, “Additional material” on page 1083 to obtain the createWebbankEnv.bat file.

23.5.13 Deleting the whole environment in one shot

You may want a script to delete an application environment so you can back out a failed installation, or to prepare for the installation of a new application release.

The steps needed to delete our Webbank environment are:

1. Stop the server group so the enterprise application can be deleted:

   ```
   XMLConfig -import actionServerGrp.xml -adminNodeName itsohost -substitute
   "action=stop;svrgrp=WebbankSG"
   ```

2. Delete the enterprise application:

   ```
   XMLConfig -import deleteApp.xml -adminNodeName itsohost -substitute
   "app=Webbank"
   ```

3. Delete each clone in the server group, for clone 1:

```
XMLConfig -import deleteAppServer.xml -adminNodeName itsohost -substitute
"node=itsohost;server=WebbankServer01"
```

and for clone 2:

```
XMLConfig -import deleteAppServer.xml -adminNodeName itsohost -substitute
"node=itsohost;server=WebbankServer02"
```

4. Delete the virtual hosts, JDBC driver and data sources, and server group, then regenerate the Web server plug-in configuration

```
XMLConfig -import deleteWebbankEnv.xml -adminNodeName itsohost
-generatePluginCfg true
```

5. Disable global security, if required:

```
XMLConfig -import enableSecurity.xml -adminNodeName itsohost -substitute
"node=itsohost;enabled=false"
```

6. Restart the node for the global security changes to take effect:

```
XMLConfig -import restartNode.xml -adminNodeName itsohost -substitute
"node=itsohost"
```

We use the deleteWebbankEnv.bat Windows batch file given in Example 23-41 to delete the whole Webbank environment in one shot.

Example 23-41 deleteWebbankEnv.bat

```
...
echo "Stop server group"
call XMLConfig -import actionServerGrp.xml -adminNodeName %node% -substitute
"action=stop;svrgrp=WebbankSG"
echo "Delete enterprise application"
call XMLConfig -import deleteApp.xml -adminNodeName %node% -substitute
"app=Webbank"
echo "Delete clone 1"
call XMLConfig -import deleteAppServer.xml -adminNodeName %node% -substitute
"node=%node%;server=WebbankServer01"
echo "Delete clone 2"
call XMLConfig -import deleteAppServer.xml -adminNodeName %node% -substitute
"node=%node%;server=WebbankServer02"
echo "Delete vhost, server group, jdbc provider and regen plugin"
call XMLConfig -import deleteWebbankEnv.xml -adminNodeName %node%
-generatePluginCfg true
echo "Disable security"
call XMLConfig -import enableSecurity.xml -adminNodeName %node% -substitute
"node=%node%;enabled=false"
echo "Restart node"
call XMLConfig -import restartNode.xml -adminNodeName %node% -substitute
"node=%node%"
...
```

To delete the whole Webbank environment in one shot, use:

```
deleteWebbankEnv itsohost
```

See Appendix D, “Additional material” on page 1083 to obtain the deleteWebbankEnv.bat file.

24

Troubleshooting

When errors occur in WebSphere we must be able to debug them. One of the most important parts of problem determination (PD) is to determine whether the error is a configuration problem, an application problem, or WebSphere problems. In this chapter we discuss the methods for tracing errors in WebSphere and also give suggestions for debugging different types of problems:

- WebSphere installation problems
- Problems starting the administrative server after installation
- Problems starting the administrative console
- Problems starting the application server
- Problems accessing an application (HTML, Servlets, JSPs or EJBs)
- Incorrect application behavior

24.1 Version 3.5 vs. Version 4.0 problem determination

The following problem determination improvements are provided in WebSphere V4.0:

- The “Troubleshooting” section in the WebSphere InfoCenter has been reworked for WebSphere V4.0. The “Problem determination quick reference” article helps you identify and solve problems.
- Hundreds of new messages have been added, while most existing messages have been reworded and enhanced. Around 50% of messages now have an explanation and user action documented in the InfoCenter.
- Log Analyzer is now officially part of the product and documented in the InfoCenter.
- The latest version of OLT/OLD adds usability improvements and is available in both components (OLT and OLD) across all supported WebSphere platforms.
- E-fix installer that eliminates error-prone manual application of e-fixes. It makes backups for easy un-installation of the e-fix, and it updates product.xml to track fixpak and e-fix levels.
- Increased focus on serviceability through team education, usability assessment and review of messages and GUI, expanding the Log Analyzer symptom database, and reviews of problems and defects from a serviceability standpoint.
- dumpNameSpace utility replaces the Java Name Tree Browser (unsupported tool) from WebSphere V3.5.

Enhancements introduced in WebSphere V3.5.2 included:

- RAS infrastructure for standard error record logging to the activity.log file in <WAS_HOME>/logs directory. It supports JSR47 - JLOG, and includes the showlog utility to convert the binary log to ASCII.

  The JRas IBM logging/trace API previously used internally in WebSphere can now be used in customer applications.
- Introduced Log Analyzer as a technology preview, allowing error analysis using a symptom database kept up-to-date by IBM support.
- Runtime message enhancements to include a component ID that identifies message source, unique message number, and message severity.

24.2 Resources for identifying problems

WebSphere provides the following sources of feedback to help with problem determination. Each one of them will be described separately in subsequent sections:

- **Console messages**. These provide a high-level view of important events, such as successful completions and fatal errors. They are an important starting point to determine the cause of any configuration problem.
- **Error pop-up windows (console)**. In some cases, the console triggers an error pop-up window. A Java exception will provide a stack trace that may give some indication of the cause of the problem.
- **Log files**. Log files collect a fairly large amount of information about the administrative server and the application servers as they initialize and run. These logs are always active, so no specific action is needed to activate them. The **Log Analyzer** tool, which is described in 24.10, “Log Analyzer” on page 998, is now part of WebSphere Application Server V4.0, Advanced Edition.
- **Traces**. Traces provide more detailed information about WebSphere components to determine what is wrong with your WebSphere environment. Traces need manual activation, and you need to remember to turn off tracing when you are done collecting the information. Traces may impact performance rather severely.

24.3 Console messages

Console messages are displayed at the bottom of the WebSphere Administrative Console. They provide a high-level view of important events, such as successful completions and fatal errors. The performance impact is minimal to none. Message events are generated by WebSphere Application Server code in response to system events occurring in the application server environment. They are always collected.

Console messages are immediate and well correlated with a user-initiated action, therefore enabling fast problem isolation.

There are five categories of message events, in order of increasing severity:

- **Audit** indicates a significant event that must be recorded.
- **Terminate** indicates that a process has terminated normally (a 0 is returned to the shell).
- **Warning** indicates that a problem has occurred that must be fixed to prevent a more serious problem.

- **Error** indicates that a severe error occurred, but did not cause processing to terminate.
- **Fatal** indicates that a process has encountered a fatal error and has terminated abnormally. When a process terminates in this way, the trace service writes its internal ring buffer to the Serious Error file and returns a -1 to the shell.

24.3.1 How to view messages

To view details on the messages in the console messages area, select the message and click the **Details** button, as shown in Figure 24-1.

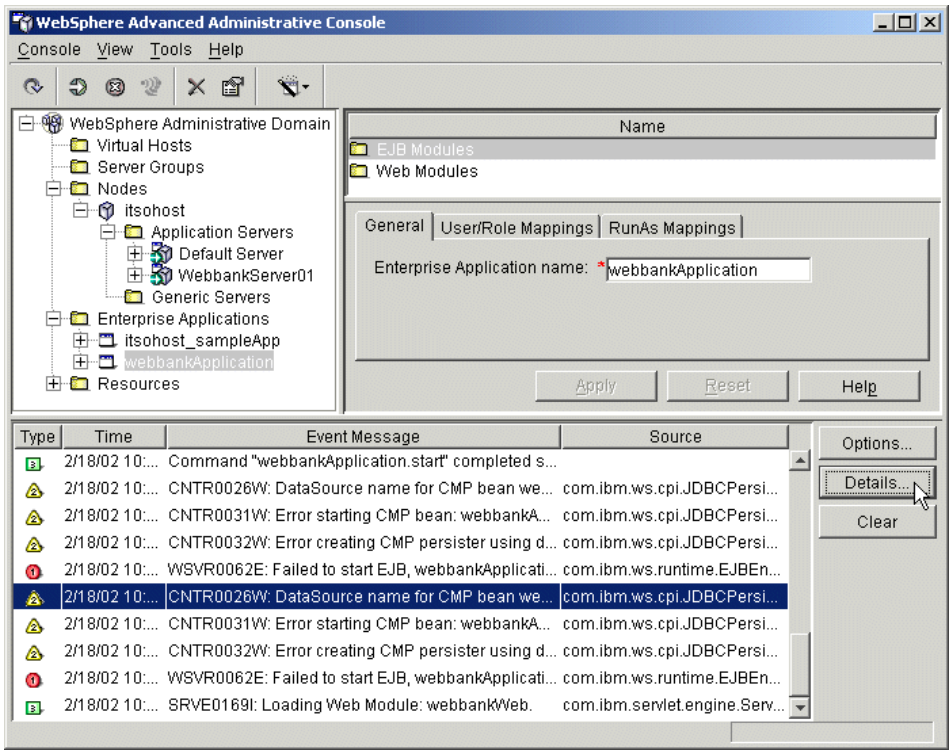


Figure 24-1 Console messages area

The Event Details window opens, as shown in Figure 24-2.

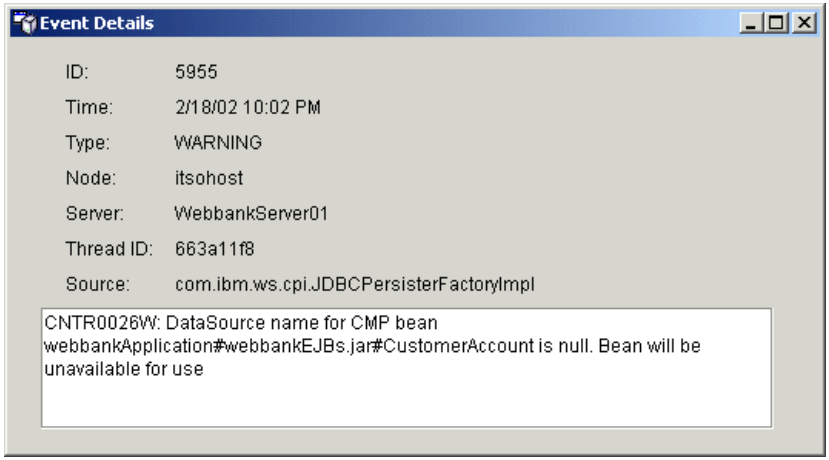


Figure 24-2 Event Details

Use the **Clear** button in the administrative console, next to the console messages area, as seen in Figure 24-1 on page 952, to clear messages from the display. Messages are not removed from the database.

To select the types of messages the console message area displays, select the **Options** button in the administrative console, next to the console messages area, as seen in Figure 24-1 on page 952. The Event Viewer Options window will be displayed, as shown in Figure 24-3.

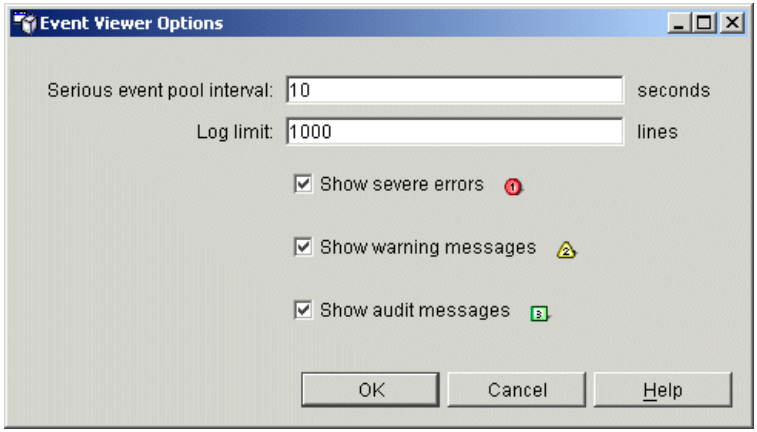


Figure 24-3 Event Viewer Options window

You can select message types for display as follows:

- **Show audit messages** option to displays audit and terminate message events.
- **Show warning messages** option to displays warning message events.
- **Show severe errors** option to display error and fatal message events.

Note: This selection does not affect the type of record that gets written to the database. All error types, audit, warning and severe, are written to the database.

Some of these messages are flagged with a prefix that relates them to a specific component of WebSphere, as listed in Table 24-1. This helps to diagnose problems and select the WebSphere component that needs to be traced.

Table 24-1 Message IDs

Prefix	Component
AATL	Application Assembly Tool
ADGU	Administrative GUI
ADMR	Administrative Repository
ADMT	Administrative Tasks
ADMS	Administrative Server
ALRM	Alarm
CHKJ	IBM Validation Tool
CHKW	WebSphere Server Validation
CNTR	EJB Container
CONM	Connection Manager
DBMN	Database Manager
DRSW	Data Replication Service
DYNA	Cache Management
INST	Install
J2CA	Connector Architecture (J2C)
JORB	IBM Java ORB
JSAS	Security Association Service
JSPG	Java Server Pages

Prefix	Component
LTXT	Localizable Text
MSGS	Messaging
MIGR	WebSphere Migration Tools
NMSV	JNDI - Name Services
PLGN	Web Server Plug-ins and Native Code
PMON	Performance Monitor
SECJ	WebSphere Security
SESN	Session and User Profiles
SMTL	WebSphere Systems Management Utilities
SRVE	Servlet Engine
TRAS	Tracing Component
WCMD	WebSphere Systems Management Commands
WINT	Request Interceptors
WOBA	WebSphere Object Adapter
WPRS	WebSphere Persistence
WSCL	Client
WSCP	WebSphere Control Program Command Line
WSVR	WebSphere Server Runtime
WTRN	WebSphere Transactions
WTSK	WebSphere Systems Management Tasks
WWLM	EJB Work Load Management
XMLC	XML Configuration

The Log limit property in the Event Viewer Options window, shown in Figure 24-3, refers to how many severe events records (audit messages, warning messages, severe errors) we want to keep in the administrative repository database (SE_Table). When the log limit is reached and a new serious event occurs, the record for the least recent event is discarded. A record is created for the new event.

24.4 Error pop-up windows

Some errors are reported by pop-up windows. Figure 24-5 is an example of a pop-up window which appeared after a critical error was logged in the console window messages area (Figure 24-4):

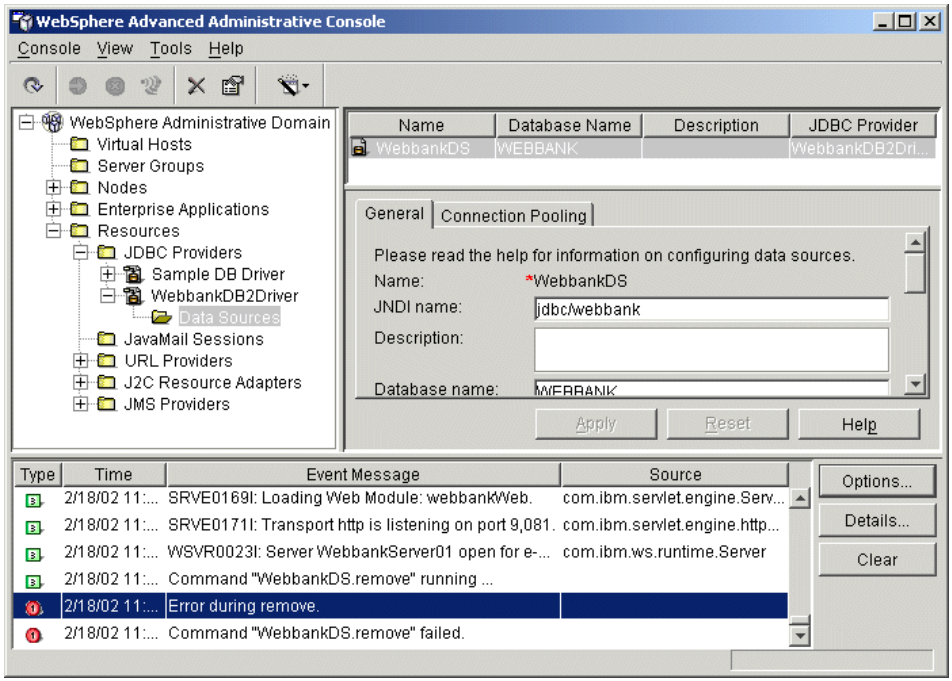


Figure 24-4 Some severe errors shown in the console generate pop-up windows

In the Critical Error pop-up window, shown in Figure 24-5, click the **Details** button to view the Java exception details.

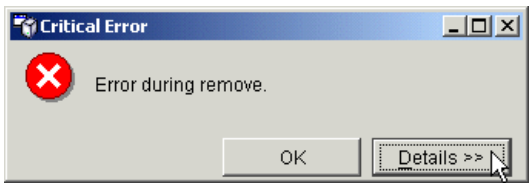


Figure 24-5 Critical Error pop-up window

The Java exception details are shown in Figure 24-6. The stack trace gives a first indication of the WebSphere code components that are causing the problem

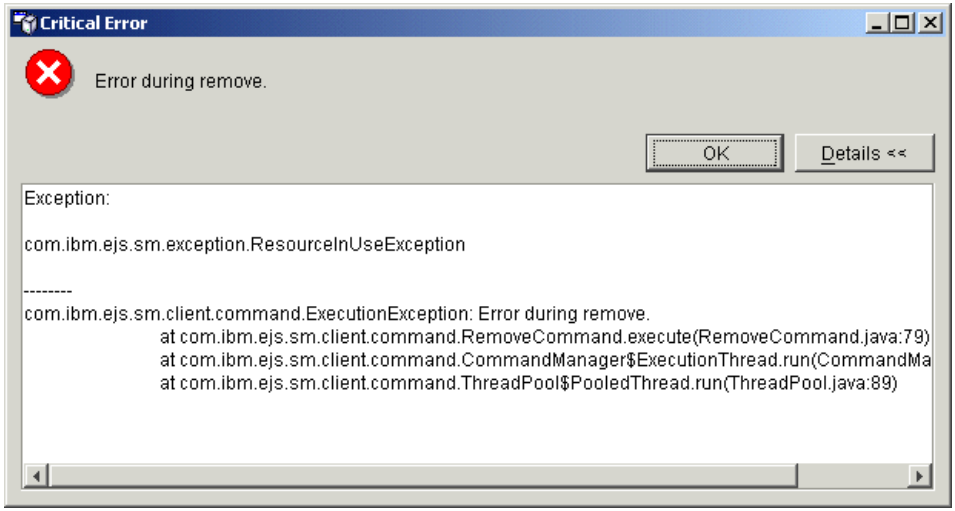


Figure 24-6 Java exception stack trace

As you can see, the above error was thrown when we tried to remove a data source that was in use.

24.5 Log files

WebSphere logs provide information about WebSphere Application Server components, including the administrative server and clients, application servers and other processes in the environment, as they initialize and run. They cause low-to-medium performance impact and can be reviewed after an error or problem condition occurs. Logs are always enabled.

The com.ibm.ws.ras.MessageFilterLevel property located in the <WAS_HOME>/properties/logging.properties file allows you to specify which levels of messages are logged by any of the system logging facilities. Valid values are:

- Error: only fatal and terminate messages are logged.
- Warning: only warning, error, fatal, and terminate messages are logged.
- Audit: all messages are logged.

The default value is audit.

Analyzing logs is a significant step in the problem determination process. This section explains log entry formats, types of logs, and when and how to use them.

24.5.1 Log entry format

Example 24-1 illustrates the basic format of a log or trace entry (the default).

Example 24-1 Basic format

```
[02.02.18 22:02:03:485 EST] 663a11f8 JDBCPersister W CNTR0026W: DataSource name
for CMP bean webbankApplication#webbankEJBs.jar#BranchAccount is null. Bean
will be unavailable for use
```

A description of each field is shown in Table 24-2.

Table 24-2 Log entry format description

Field	Example	Description
TS	`[02.02.18 22:02:03:485 EST]`	The timestamp in fully qualified date (YYMMDD), Time (Millisecond precision), and Time Zone Format
TID	`663a11f8`	The thread ID or the hash code of the thread issuing this message
COMPONENT	`JDBCPersister`	The short name of component issuing this message
LEVEL	`W`	The level of the message. Possible levels are: **>** Entry to a method (debug) **<** Exit a method (debug) **A** Audit **W** Warning **X** Error **E** Event (debug) **D** Debug (debug) **T** Terminate (exits process) **F** Fatal (exits process) **I** Informational
MESSAGE	`CNTR0026W: DataSource name for CMP bean webbankApplication#webbankEJBs.jar#BranchAccount is null. Bean will be unavailable for use`	The text of the message and message arguments

24.5.2 Product logs

WebSphere Application Server provides a comprehensive range of logs to help you diagnose problems:

- Installation logs
- Administrative server logs
- Application server logs
- WebSphere activity log
- Web server and Web server plug-in logs

24.5.3 Installation logs

The following installation logs can be found in the <WAS_HOME>/logs directory:

- wssetup.log (NT)
- install.log (UNIX)
- wasdb2.log (NT)

These logs can be reviewed to check that the installation of WebSphere completed successfully. You would want to scan these logs for errors if:

- The installation hangs.
- The installation finishes but components are missing.
- The WebSphere administrative server does not start for the first time.
- The plug-in does not seem to have installed properly.
- After installation, just to check that there were no problems during the install.

wssetup.log (NT) / install.log (UNIX)

The wssetup.log and install.log files are created during the install process. Review this log to ensure that the install process was successful. The install process consists of:

- Verifying prerequisites.
- Copying files.
- Updating the configuration files for both WebSphere Application Server and the Web server.

wasdb2.log (NT)

On Windows NT, when you install WebSphere Application Server, the installation program creates a DB2 database to store the administrative configuration that is used when your system starts up after rebooting. Further, the installation program should create the WAS database with its DB2 application heap size set to 256. The createdb2.bat program runs under the covers, and the wasdb2.log file is created. Review this log to determine if the WAS database was created correctly. Errors creating the system management repository tables will be logged in this file.

On UNIX, the database needs to be created manually; see Part 3, “Installing WebSphere” on page 125.

24.5.4 Administrative server logs

The following administrative server log files can be found in the <WAS_HOME>/logs directory by default:

- tracefile
- nanny.trace
- adminserver_stderr.log

tracefile

The tracefile describes the startup and shutdown of the administrative server and the application servers. It also shows interactions of different WebSphere Application Server components with the administrative server. This file needs to be gathered almost *always*, in order to find the source of a problem.

Use the tracefile to identify a problem and to review events preceding the error situation.

Note: Always review tracefile entries prior to the error. Trace entries recorded after the error has occurred represent program recovery and will not help with problem determination.

nanny.trace

On UNIX systems, a separate process called “nanny” launches the administrative server and attempts to relaunch it if it fails.The com.ibm.ejs.sm.util.process.Nanny.maxtries property in the admin.config file tells the nanny process how many times it should attempt to restart the administrative server. Attempts to start the administrative server are logged in the nanny.trace file.

On Windows NT, the nanny service is part of the IBM WS AdminServer 4.0 service that is registered with the operating system. Starting the IBM WS AdminServer 4.0 service invokes adminservice.exe.

A nanny trace is only available on UNIX platforms.

On Windows, use the Event Viewer to view entries related to the WebSphere nanny service.

1. On Windows NT, select **Start** -> **Programs** -> **Administrative Tools**.

   On Windows 2000, select **Start** -> **Programs** -> **Administrative Tools** -> **Computer Management**.
2. Select **Event Viewer**.
3. View application events related to WebSphere Application Server.

adminserver_stderr.log

Administrative server startup errors are usually placed in this file. If this file is not empty, it usually means an error condition occurred.

24.5.5 Application server logs

WebSphere Application Server creates standard output and standard error logs for each application server. They contain application server communication.

The location and name of the standard output and standard error files are defined by the application server administrator. To see the path name of these logs:

1. Locate the required application server in the administrative console tree view.
2. Right-click the application server and select **Properties** from the pop-up menu.
3. The standard output and standard error details can be found under the File tab, as shown in Figure 24-7. Combine relative log file paths, as shown in Figure 24-7, with the Working directory setting on the General tab to find the absolute path.

   We recommend that you set a unique standard output and standard error log file for each application server on a node.

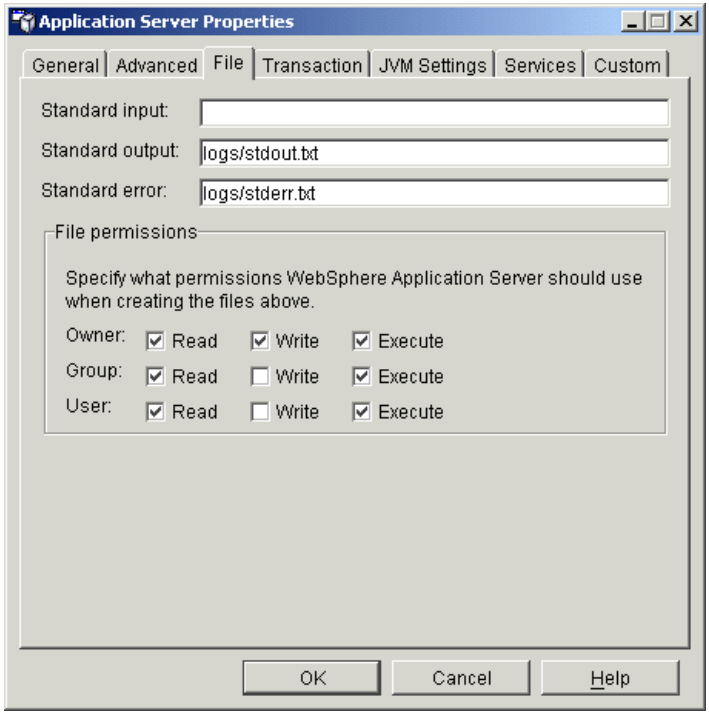


Figure 24-7 Standard output and standard error logs

Standard output log

The standard output file contains System.out messages from the application server, indicating whether the code running on the application server started and stopped successfully.

If the System.out.println statement is added to the user code (such as servlets or EJB components), then output from the println statement also appears in the application server standard output log.

Standard error log

The standard error file contains errors thrown by the application server. It identifies exceptions thrown by the code running on the application server or System.err messages from the application server.

If the System.err.println statement is added to the user code (such as servlets or EJB components), then output from the println statement also appears in the application server standard error log.

24.5.6 Activity log

The activity log captures events that show a history of WebSphere Application Server's activities. Some of the entries in the log are informational, while others report on system exceptions, such as returned CORBA exceptions. When you encounter WebSphere Application Server runtime errors, you will often find it useful to use Log Analyzer to read the activity log and try to diagnose the problem.

The activity.log file is a binary file located in the <WAS_HOME>/logs directory and cannot be viewed with an ASCII editor. Use either the **showlog** command or the Log Analyzer to view the file:

- **showlog**. Follow these instructions to view the activity file using **showlog**:
  a. Change directory to <WAS_HOME>/bin.
  b. Run the showlog tool with no parameters to display the usage instructions.
     - To direct the activity log contents to standard output, use the invocation:

       ```
       showlog ../logs/activity.log
       ```

     - To dump the activity.log to a text file that can be viewed using a text editor, use the invocation:

       ```
       showlog ../logs/activity.log > text_file_name
       ```

- **Log Analyzer**. To view the activity.log file with the Log Analyzer, follow these instructions:
  a. Change the directory to <WAS_HOME>/bin.
  b. Run the waslogbr script file to start the Log Analyzer graphical interface.
  c. Select **File** -> **Open...** from the Log Analyzer main menu.
  d. Navigate to the directory containing the activity.log file.
  e. Select the activity.log file.
  f. Select **Open**.

  The Log Analyzer is discussed in 24.10, "Log Analyzer" on page 998.

If the platform on which you are running WebSphere Application Server does not support the use of a GUI, you could transfer the file in binary to the system on which you are running the administrative console and use the Log Analyzer tool there. Alternatively use **showlog** as per above.

Note: The Log Analyzer does not provide access to remote files, but remote file systems can be used.

All application servers, including the administrative server, write error records to the activity.log file. There is one activity log for each machine.

The activity.log file grows to a predetermined size and then wraps. The default size is 1MB. Follow these steps to change the log size:

1. Open the properties file in a text editor: <WAS_HOME>/properties/logging.properties.
2. For the com.ibm.ws.ras.ActivityLogSize property, specify the value you would like in kilobytes. The minimum is 8 KB and maximum is 1048576 KB (1 gigabyte).

   For example, specify a log size of 2 MB as follows (without spaces):

   ```
   com.ibm.ws.ras.ActivityLogSize=2048
   ```

   The size change will take effect at the next server startup.

Tip: In most cases, IBM support personnel will ask you to gather the standard output, standard error, tracefile and the activity.log files in order to debug WebSphere problems.

24.5.7 Other logs

Plug-in trace log

The plug-in log file is created by the WebSphere plug-in running in the Web server process. The default plug-in log file is <WAS_HOME>/logs/native.log.

The plug-in log contains error and informational messages generated from the Web server plug-in. This information reflects Web server startup and server status change requests (start/stop/restart). Different levels of information can be placed in this log via the plug-in configuration file, plugin-cfg.xml. The default setting is `error`. See Appendix C, “The plugin-cfg.xml file definitions” on page 1071.

Note: If the Web server is configured on a different machine from the application server, then the native.log and the plugin-cfg.xml files reside on the machine where the Web server is installed.

You will want to gather this log if you have problems accessing servlets, JSPs or HTML from a browser (HTTP or HTTPS). If that is the case, you will also want to gather the Web server error logs.

Web server log files

There will be additional log files produced by your chosen Web server. You will need to refer to the product-specific configuration to locate and interpret those files (x:\IBM HTTP Server\logs for the IBM HTTP Server).

24.6 Traces

Tracing can be useful if you are having problems with particular components of WebSphere Application Server and the log files don't provide you with enough information to determine what the problem exactly is.

The application server product has built-in tracing statements, and as the code executes, traced information is sent to a specified file or stream so it can be analyzed.

WebSphere Application Server supports multiple log and trace formats, which are specifiable via the following property in the <WAS_HOME>/properties/logging.properties file:

```
com.ibm.ws.ras.TraceFormat
```

There are three formats:

- **Basic** generates the sparse legacy trace format, as seen in Example 24-1.
- **Advanced** is a more verbose style patterned after the basic format, as shown in Example 24-2. It is recommended in most cases.
- **Loganalyzer** generates a trace output that can be parsed by the Log Analyzer tool. As shown in Example 24-3, the trace files generated are larger than the other options.

Example 24-2 Advanced format

```
[02.02.19 11:34:29:909 EST] 31e1be6a W UOW=1-2936:itsohost/WebbankServer01
source=com.ibm.ws.cpi.JDBCPersisterFactoryImpl org=IBM prod=WebSphere
          CNTR0026W: DataSource name for CMP bean
webbankApplication#webbankEJBs.jar#BranchAccount is null. Bean will be
unavailable for use
```

Example 24-3 Loganalyzer format

```
ComponentId:
ProcessId:    2900
ThreadId:     6d66ffa3
FunctionName:
ProbeId:
SourceId:     com.ibm.ws.cpi.JDBCPersisterFactoryImpl
Manufacturer: IBM
```

```
Product:      WebSphere
Version:      AE 4.0.1 a0131.07
SOMProcessType:
ServerName:   itsohost/WebbankServer01
ClientHostName:
ClientUserId:
TimeStamp:    2002-02-19 11:44:56.540000000
UnitOfWork:   1-2900:itsohost/WebbankServer01
Severity:     2
Category:     WARNING
FormatWarning:
PrimaryMessage:
ExtendedMessage: CNTR0026W: DataSource name for CMP bean
webbankApplication#webbankEJBs.jar#BranchAccount is null. Bean will be
unavailable for use
RawDataLen:   0
```

The loganalyzer format is recommended when doing cross-process trace. Use the following steps to do a cross-process trace:

1. Move all the trace files to a directory on a single system. The trace files must have been generated in loganalyzer format.
2. Launch the Log Analyzer, select **File** -> **Open** from the main menu, navigate to that directory and open one of the trace files.
3. Next, click **File** -> **Merge with** and select another trace file. This will merge the contents of the two files in the Log Analyzer display. Log Analyzer is discussed in 24.10, “Log Analyzer” on page 998.

The trace system provides the most extensive information. However, tracing a server (either the Administrative server or the Application server) is very demanding on system resources. Enable it only temporarily for problem determination and remember to disable it once you are done with the tracing.

The following are a few tips for reading a trace file:

- Timestamps are good when comparing traces from different processes.
- Look for exceptions. Events prior to the exception are probable causes.
- It is often useful to follow a single thread.

When tracing a server that is running, trace is transported by an in-memory circular buffer called a ring buffer. The ring buffer must be dumped to a file in order for the trace to be viewed, as illustrated in Figure 24-8.

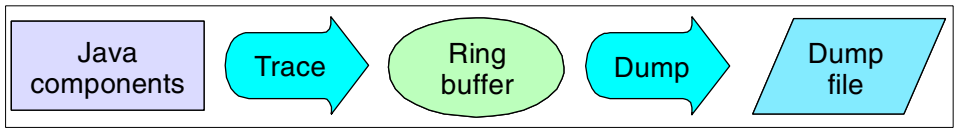


Figure 24-8 Tracing information is stored in memory ring buffer

24.6.1 Tracing a running application server from the console

If you need to trace a server that is already active, you can enable it from the administrative console as follows:

1. Locate the required application server in the tree view.
2. Right-click the application server and select **Properties** from the pop-up menu.
3. In the Application Server Properties window, select the **Services** tab.
4. In the service list, select **Trace Service** and click the **Edit properties...** button, as shown in Figure 24-9.

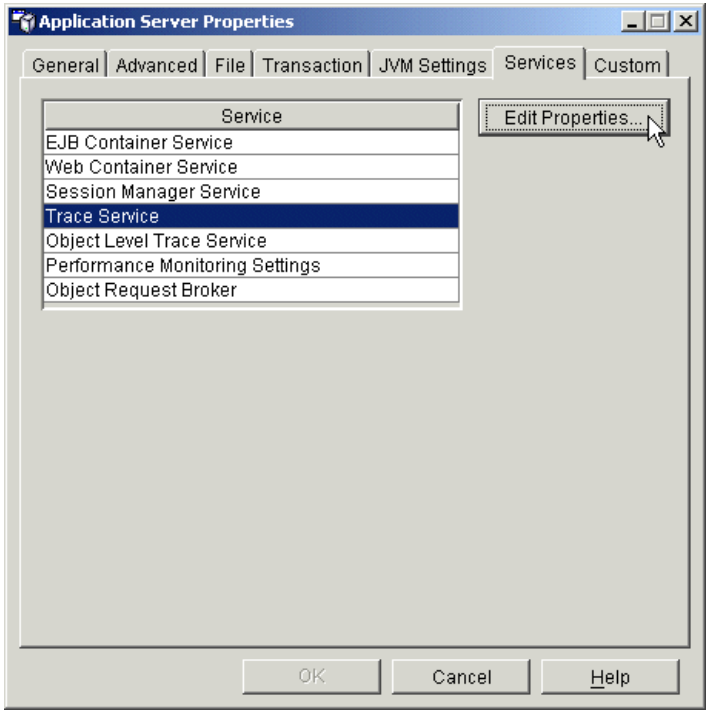


Figure 24-9 Application Server Properties window

5. In the Trace Service window (shown in Figure 24-11 on page 970) click the [...] button in the Trace specification field.
6. The Trace window displays the classes and groups of classes that can be traced. As shown in Figure 24-10, choose the components to trace by right-clicking and selecting the trace type.

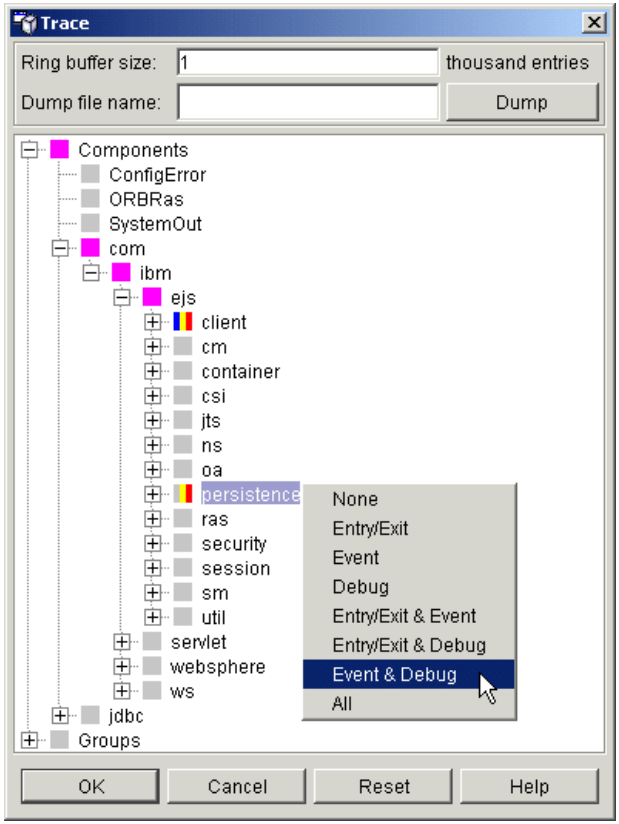


Figure 24-10 Trace window

Table 24-3 summarizes the various components that can be traced. So if you are experiencing problems with a particular component, you can just trace that component.

Table 24-3 Components to trace

Abbreviation	Component
client	Administrative client
cm	Connection pooling manager, data sources

Abbreviation	Component
container	EJB container - TransactionRolledBackException
csi	Container Service Interface
jts	Java Transaction Service
ns	Naming Service, Java Naming Directory Interface
oa	ORB implementation
persistence	EJB persistence layer
security	Security manager, security context
session	Session manager
sm	Administrative server process, application server process
servlet	Web application engine
websphere	Data source factory, JNDI context factory
ws	Classloader, HTTP transports, WLM

Depending on the trace type selected, you will see different colors beside the selected component, as listed in Table 24-4.

Table 24-4 Trace type colors

Trace type	Color
None	Gray
Entry/Exit	Blue
Event	Yellow
Debug	Red

7. Set the ring buffer size in thousands of lines.
8. Click the **OK** button.

The components and trace types are now listed in the Trace specification field in the Trace Service window, as shown in Figure 24-11.

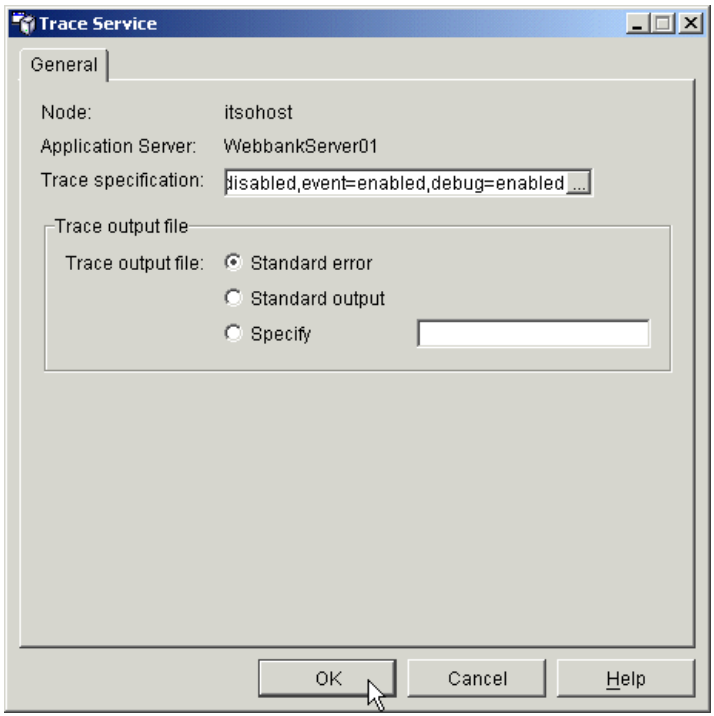


Figure 24-11 Trace Service window

The following is the entry for the selections made in Figure 24-10 on page 968:

```
com.ibm.ejs.client.*=all=enabled:com.ibm.ejs.persistence.*=all=disab
led,
   event=enabled,debug=enabled
```

Note the trace specification string format:

```
<component>=<type>=[enabled|disabled]
```

9. Click **OK** in the Trace Service window.
10. Click **OK** in the Application Server Properties window to start the trace. The trace information will be sent to a ring buffer in memory.

When you have completed testing, dump the ring buffer to a file as follows:

1. Right-click the required application server in the console tree view, and select **Properties** from the pop-up menu.
2. In the Application Server Properties window, select the **Services** tab.
3. In the service list, select the **Trace Service** and click the **Edit properties...** button.

4. Click the [**...**] button in the Trace specification field. You should see the classes and packages that were selected for tracing, as shown previously in Figure 24-10 on page 968.
5. Specify a dump file name and click the **Dump** button.
6. Click **OK** in the Trace window.
7. Click **OK** in the Trace Service window.
8. Click **OK** in the Application Server Properties window.

You should see the message `Command "dumpRingBuffer" completed successfully` in the administrative console message view, as shown in Figure 24-12.

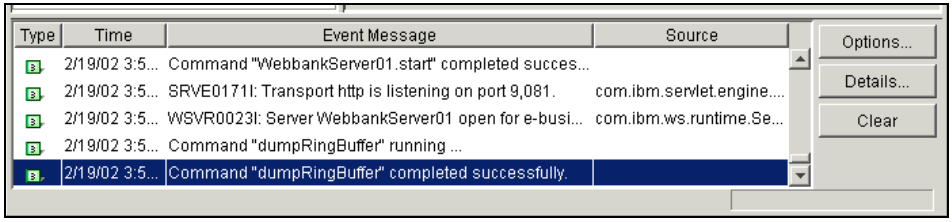


Figure 24-12 Dump ring buffer completed successfully

You can now examine the dump file with a text editor.

To turn tracing off:

1. Right-click the required application server in the console tree view, and select **Properties** from the pop-up menu.
2. In the Application Server Properties window, select the **Services** tab.
3. In the service list, select **Trace Service** and click the **Edit properties...** button.
4. Click the [**...**] button in the Trace specification field to open the Trace window. You should see the classes and packages that were selected for tracing, as shown previously in Figure 24-10 on page 968.
5. Right-click the required components and set the trace type to **None**.
6. Click **OK** in the Trace window.
7. Click **OK** in the Trace Service window.
8. Click **OK** in the Application Server Properties window.

24.6.2 Tracing a running administrative server from the console

To trace a running administrative server from the console, follow the procedure described in 24.6.1, “Tracing a running application server from the console” on page 967, but in this case replace the first two steps as follows:

1. Locate the required node in the console tree view.
2. Right-click the node and select **Trace...** from the pop-up menu, as shown in Figure 24-13. This opens the **Trace** window, as seen in Figure 24-10 on page 968.

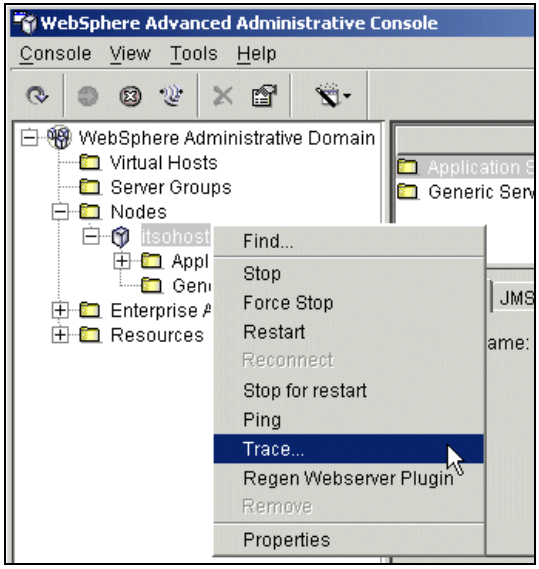


Figure 24-13 Starting tracing for the running administrative server

Continue as described in 24.6.1, “Tracing a running application server from the console” on page 967.

24.6.3 Tracing application server startup with the console

If you need to trace an application server that does not start, you can use the console to specify a *trace string*, which indicates the components that you want to trace, as an application server startup parameter. The trace will be activated during the process of bringing up the server.

If you need to trace a server that does not start, you can enable it from the administrative console as follows:

1. Locate the required application server in the tree view.

2. Right-click the application server and select **Properties** from the pop-up menu.
3. In the Application Server Properties window, select the **Services** tab.
4. In the service list, select **Trace Service** and click the **Edit properties...** button.
5. In the Trace Service window, note that the [**...**] button in the Trace specification is disabled for a stopped application server. The trace specification needs to be manually specified. As shown in Figure 24-14 on page 974, specify the trace string, for example:

   ```
   com.ibm.ejs.client.*=all=enabled
   ```

   The trace specification string format is as follows:

   ```
   component=[(level=state),(level=state),...]:component=[(level=state),...]
   ```

   Where:

   – `component` is the fully qualified Java class name, for example com.ibm.ejs.sm.server.AdminServer, or a Java package such as Servlet_Engine.

     Components can be specified with a * wildcard. In the example above, we are tracing all administrative client subcomponents.

   – `level` is the type of tracing to perform and can be `entryExit`, `event`, `debug` or `all`. You can specify more than one type by separating them with commas, for example:

     ```
     com.ibm.ejs.persistence.*=all=disabled,event=enabled,debug=enable
     d
     ```

     You can also disable subcomponents of traced components, for example:

     ```
     com.ibm.ejs.sm.*=all=enabled:com.ibm.ejs.sm.client.*=all=disabled
     ```

     Note how the components are separated by colons.

   – `state` is either `enabled` or `disabled`.
6. In the Trace Service window, also specify the Trace output file. If you do not specify a path, the file will be created in the <WAS_HOME>/bin directory.

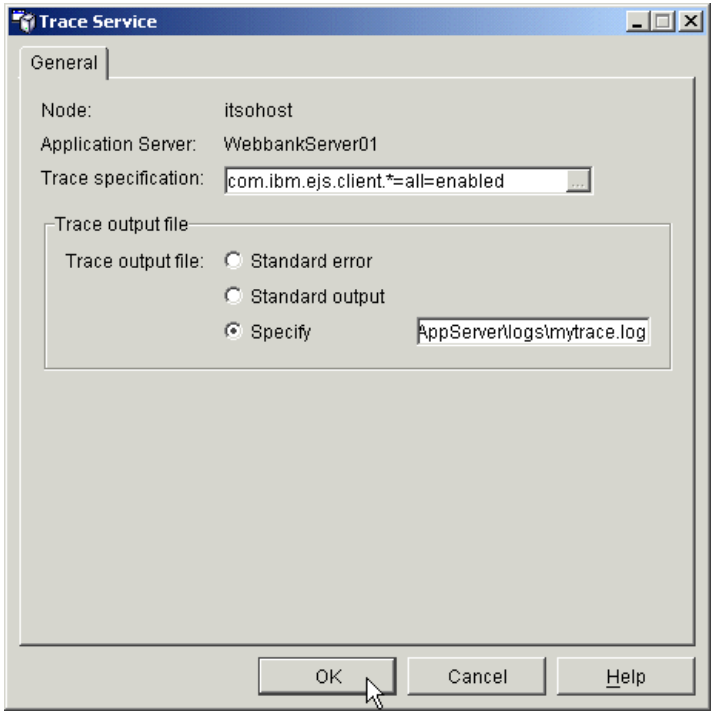


Figure 24-14 Trace Service window - application server stopped

7. Click **OK** in the Trace Service window.
8. Click **OK** in the Application Server Properties window.

Collect trace data by trying to start up the application server. You can examine the trace output file with a text editor.

Turn tracing off by repeating steps 1 to 8, deleting the Trace specification string from the Trace Service window.

Note: Tracing to a file is likely to be slower than tracing to an in-memory ring buffer, so we suggest that you only use this option when the application server does not start.

24.6.4 Tracing administrative server startup

If you have problems starting the administrative server, you can enable tracing for more detail.

To trace the administrative server from startup, do the following:

1. Open the admin.config file in the <WAS_HOME>/bin directory.
2. Un-comment the two following lines:

   ```
   # Trace settings
   #com.ibm.ejs.sm.adminServer.traceString=com.ibm.ejs.*=all=enabled
   #com.ibm.ejs.sm.adminServer.traceOutput=<WAS_HOME>/logs/admin.trace
   ```

   `traceString` allows you to specify the startup trace string. The default is to trace everything.

   `traceOutput` allows you to specify a file for the trace output. You can send the output to a file or standard output.

   – Standard output appends the trace output to the trace file.

   – If trace output is sent to a file, specify the full path to the file. The default is <WAS_HOME>/logs/admin.trace.
3. Start the administrative server.
4. View the generated trace output for information regarding the failure.

Note: Tracing the administrative server will have a severe impact on performance.

24.6.5 Tracing the administrative console

To trace the administrative client:

1. From the <WAS_HOME>/bin directory, invoke the administrative console as follows:

   ```
   adminclient debug > adminclient.out
   ```

   or:

   ```
   adminclient -v > adminclient.out
   ```
2. Review the entries in the output file adminclient.out.

Note: Run adminclient.sh on UNIX and adminclient.bat on Windows.

24.6.6 Tracing from the command line using DrAdmin

DrAdmin is a service (server thread) that runs in each of the servers, administrative server and application servers, that is dedicated to servicing the trace facility. Each time a server starts, DrAdmin registers itself on a different, next available, port number.

You should always use the administrative console trace facilities to debug a problem. DrAdmin is a diagnostic tool of last resort, used primarily to dump the stack of running or hung JVMs. It was designed to provide IBM support personnel with another way of obtaining traces. Use DrAdmin in these situations only:

- When input to the administrative console is not accepted.
- When the administrative server is in a wait state.
- When the administrative server is not responding.
- When you have to dump the thread stacks in a server (a Java dump provides information about the executing Java classes and allows the problem determination process to reference the Java source code).
- When the administrative console topology tree disappears.

The DrAdmin interface is the same on all platforms. Since DrAdmin is another way of turning on trace, the tracing mechanism is the same as the one used by the administrative console trace facilities. Therefore, whether you are looking at the trace file or a DrAdmin output file, the trace entries will have the same format.

The following example shows how to invoke DrAdmin to generate a Java thread dump:

1. Find the port number for DrAdmin.The port number is recorded in:
   - The <WAS_HOME>/logs/tracefile file, for the administrative server.
   - The standard output file defined for each of the application servers. The location of this files is set through the administrative console, as described in 24.5.5, “Application server logs” on page 961.

   The DrAdmin port number entry will look like the following:

   ```
   [02.02.19 19:11:45:117 EST] 159ef406 DrAdminServer I WSVR0053I: DrAdmin
   available on port 1865
   ```

2. Start DrAdmin from the <WAS_HOME>/bin directory, and provide the DrAdmin port number from the previous step as a command option:

   ```
   DrAdmin -serverport <port_number> -dumpThreads
   ```

3. View the thread dump that gets appended to the end of the standard error log file:
   - <WAS_HOME>/logs/adminserver_stderr.log for the administrative server.
   - Standard error file for the application servers.
4. View also the javacore file that gets created. The name and location of the javacore file is recorded at the bottom of the error log file, as follows:

   ```
   Writing java dump to C:\WINNT\system32/javacore2624.1014171140.txt...  OK
   ```

Note: On UNIX platforms, **`kill -3`** can also be used to obtain Java Thread Dumps. As the root user, issue **`kill -3`** against the process ID of the WebSphere JVM in which you need to see the thread dump. The location of the thread dump will depend on the operating system (<WAS_HOME>/bin/javacore.txt on AIX).

DrAdmin can be used for tracing as follows:

1. To enable tracing, use:

   ```
   DrAdmin -serverPort <port number> -setTrace <trace specification>
   ```

   For example:

   ```
   DrAdmin -serverPort 2108 -setTrace com.ibm.ejs.client.*=all=enabled
   ```

2. To dump the ring buffer to file, use:

   ```
   DrAdmin -serverPort <port number> -dumpRingBuffer <Dump file>
   ```

   For example:

   ```
   DrAdmin -serverPort 2108 -dumpRingBuffer ring.txt
   ```

3. To disable tracing, use:

   ```
   DrAdmin -serverPort <port number> -setTrace <trace specification>
   ```

   For example:

   ```
   DrAdmin -serverPort 2108 -setTrace com.ibm.ejs.client.*=all=disabled
   ```

DrAdmin has a help file available. You can access the DrAdmin help by using the -help option from the command line:

```
DrAdmin -help
```

Note: DrAdmin is an internal interface that is used to assist users with problem determination. As an internal interface, it is subject to change at any time, and there is no national language support for it.

24.7 System thread dumps vs. Java thread dumps

If possible, Java processes will produce dumps when they crash. These dumps can provide useful information as to why the process crashed. Two files will usually be generated in the process's working directory:

- Javacore file or Java thread dump: `javacorennnn.mmmmmmmmmm.txt`, where:
  - `nnnn` will be the process ID of the failed process.
  - `mmmmmmmmmm` is the time (in internal format) to ensure javacores are unique.

- Core dump or system thread dump.

Java thread dumps provide a Java view of a failing JVM process. They provide information about the executing Java classes and allow the problem determination process to reference the Java source code.

System thread dumps provide a system view of a failing JVM (Java Virtual Machine) process. They do not understand Java classes. Everything in a system dump is C library oriented. The system dump information provided for JVM processes refers to Java's C libraries and not the reference class files. Mapping this information back into Java source code is very difficult.

Javacore files are text files and are largely readable in themselves.

Core files need to be examined with the **dbx** command. On UNIX systems, they usually appear as core files. On Window's systems they appear as drwtsn32.log files.

System dumps should only be interrogated when a Java thread dump is unavailable.

See 24.7.1, “Looking at core files on UNIX with dbx” on page 978 and 24.7.2, “Core files on the Windows platform” on page 980 for explanations of how to interrogate system dumps.

24.7.1 Looking at core files on UNIX with dbx

Core files on UNIX systems can be interrogated using the dbx and gdb tools. Dbx is a tool that is part of the AIX install (it might not be installed by default). On Sun, dbx can be installed for an additional expense. The gdb (GNU debugger) is freeware and can be downloaded.

To make sure that a good core file gets generated:

1. Ensure that the system core file size specification is unlimited (ulimit)
2. Ensure that the file system containing the core file has enough space (df).

The common procedure for interrogating a core file is to change the directory to where the core file resides. You can then issue the **dbx** command with the binary executable file as the parameter. It is important that the binary executable is used.

The commands listed in Example 24-4 show how to find the binary executable and invoke the **dbx** command. It also shows that an illegal instruction was executed (that is, Invalid opcode). The sample was taken from an AIX 4.3.3 system.

Example 24-4 Invoking dbx

```
# cd /usr/jdk_base
# find . -name java -print
./bin/aix/native_threads/java
./bin/java
# cd /usr/WebSphere/AppServer/bin
# ls -l core
-rw-r--r-- 1 root system 191495883 Aug 07 15:08 core
# dbx /usr/jdk_base/bin/aix/native_threads/java
Type 'help' for help.
Warning: The core file is truncated. You may need to increase the
ulimit for file and core dump, or free some space on the file system.
Reading symbolic information ...Warning: no source compiled with -g
[using memory image in core]
Illegal instruction (reserved addressing fault) in . at 0x0
($t29)0x00000000 00000001 Invalid opcode.
```

If you do not know where the Java binary is located, the following command will display the true Java executable name of the core:

```
# strings core | more
```

After you start dbx, the **where** command provides a stack trace of where the error occurred, as shown in Example 24-5.

Example 24-5 dbx where command

```
(dbx) where
warning: could not locate trace table from starting address 0x0
ExecuteJava(??, ??) at 0xd2f9913c
do_execute_java_method_vararg(??, ??, ??, ??, ??, ??, ??, ??) at
0xd2fabd30
execute_java_dynamic_method(0x20e355e0, 0x3002fdb0, 0xd3016aa4,
0xd3016aa8, 0x0, 0x0, 0x0, 0x0) at 0xd2fabef4
ThreadRT0(0x3002fdb0) at 0xd300cd88
sysThread_shell(??) at 0xd2fb50a8
pthread._pthread_body(??) at 0xd010f358
```

Type `help` for help on a command or topic and `quit` to exit dbx.

Another use of the dbx command is to monitor a running process. The **-a** parameter allows the user to attach to a process. The **catch** and **run** commands can be used to walk through the processing of the JVM process and see all signals that are caught. Use the **help** command for additional information on each of the above commands.

Note: In 24.6.6, "Tracing from the command line using DrAdmin" on page 975, we explained how to dump threads using DrAdmin or the **kill** command. If a process appears to hang, it is probably a good idea to get both a Java thread dump and a system thread dump:

1. To generate the Java thread dump (generates a javacore file):

   ```
   # kill -QUIT <java_pid>
   ```

2. To generate the system thread dump:

   ```
   # dbx -a <pid>
   ```

24.7.2 Core files on the Windows platform

Windows Dr. Watson log files are similar to core files on UNIX. To find out about the format of Windows Dr. Watson log files:

1. Open a command prompt and enter the following command:

   ```
   C:\>drwtsn32
   ```

2. Click **Help** then **Dr. Watson log file overview**.

24.8 Tracing user code

The trace subsystem does not trace user code (that is, servlets or EJB components) unless System.err.println or System.out.println statements are added to the code. Output from the println statements appears either in the application server standard output or standard error logs.

WebSphere Application Server V4.0 provides the Object Level Trace/Object Level Debugger (OLT/OLD) and JRAS debugging tools for tracing and debugging user code.

24.8.1 Object Level Trace/Object Level Debugger (OLT/OLD)

The Object Level Trace/Object Level Debugger is a comprehensive tool that allows tracing and debugging of user-written code running a distributed environment, from a single workstation. This tool can be used to debug a variety of environments and different programming languages, including the Java applications that run on WebSphere.

The tool has two main components:

1. The Object Level Trace (OLT) is a distributed object tracing facility that allows you to see the interactions between objects involved in a Web application request. For example, if you traced a servlet that called another servlet in your application server, in the OLT GUI you would see a representation of both servlets and arrows between them showing the method call relationships. It provides a graphical (time-oriented format) representation of the interactions between distributed objects, that is Java clients, EJBs, JSPs and Servlets.

   It consists of:

   – OLT event server. It records method calls from your application to distributed business objects, servlets, JSPs, or EJBs residing on the WebSphere Application Server.

   – OLT viewer. The viewer communicates with the server and provides a graphical representation of the interactions between a client and the objects servicing that client.

   – OLT runtime. It provides the APIs that a particular application may call in order to trace events and method calls on its objects. An application server process that runs in WebSphere “hooks up” with the OLT runtime once you activate the Object Level Trace service.

   – Debugger daemon. Started automatically by OLT when you register a client executing in Trace and Debug or Debug Only mode. This daemon launches a debug session (OLD) when you choose to step into a debuggable method.

   OLT helps in isolating communication errors between the components of the application, and it gives performance information through the OLT trace window.

   To be able to use this tracing tool in a WebSphere environment, it is necessary to enable it in the application server.

2. The Object Level Debugger (OLD), is a client/server application that allows step-by-step debugging of the source code both locally or through a network connection. It works like most debuggers. You can step through source code, set breakpoints, inspect variables and perform other actions.

   It is composed of:

– **Debugger engine**. It runs on the same system as the program you want to debug. This system can be your workstation or a system accessible through a network.
– **Distributed debugger client**. A graphical user interface where you can set up breakpoints and control the execution of the debugged applications.

  It allows you to debug multiple applications, which may be written in different languages, from a single debugger session.

  The debugger engine and the client interface can be installed in the same or in different machines.

Note: The Java classes to be debugged should be compiled with the -g option to include debugging information.

Figure 24-15 shows how all the different OLT/OLD components interact.

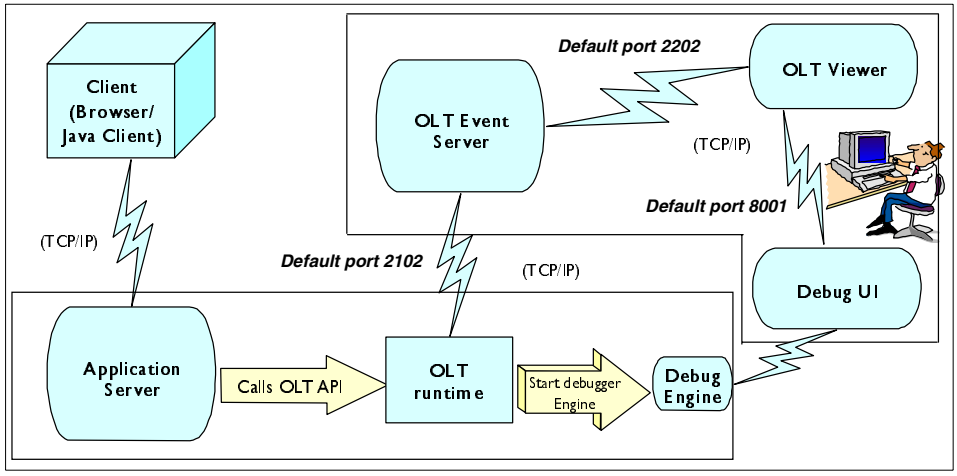


Figure 24-15 OLT/OLD component interaction

You can debug your application from the client machine, the server, your development workstation, or any other machine on which the IBM distributed debugger is installed.

It is beyond the scope of this book to discuss OLT/OLD in detail, but we will show you how to enable OLT and the Distributed Debugger from the administrative console, so you can debug code running on the application server. For more information on OLT/OLD, refer to the *WebSphere Version 4 Application Development Handbook*, SG24-6134, the documentation in InfoCenter, or the OLT/OLD documentation.

Activating object level trace

To enable tracing, do the following from the administrative console:

1. Locate the required application server in the tree view.
2. Right-click the application server and select **Properties** from the pop-up menu.
3. In the Application Server Properties window, select the **Services** tab.
4. In the service list, select the **Object Level Trace Service**, and click the **Edit properties...** button, as shown in Figure 24-16.

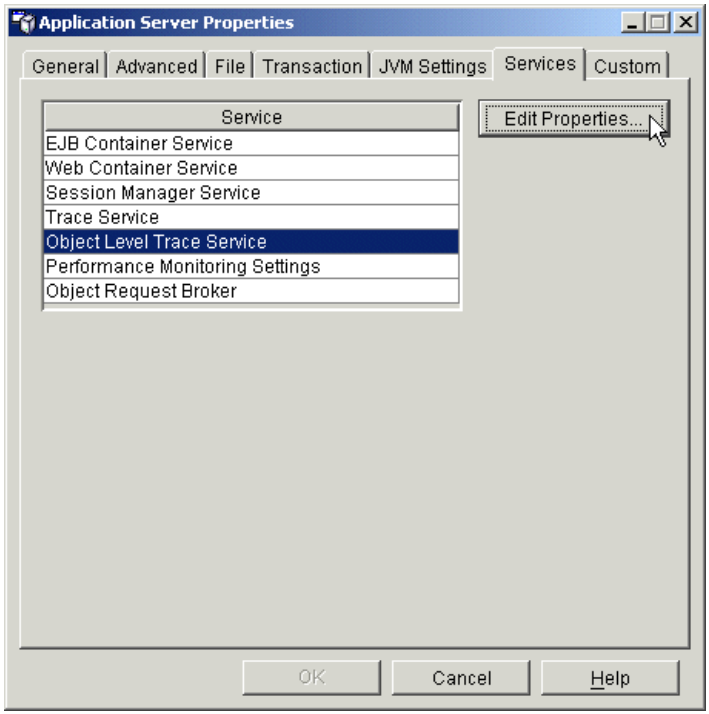


Figure 24-16 Activating Object Level Trace

5. In the Object Level Trace Service window, shown in Figure 24-17:
   a. Check the **Object Level Tracing enabled** box.
   b. Specify where the OLT server runs. This is the fully qualified host name of the machine on which the OLT trace server is located. Trace results will be directed to that machine (it could be a different machine or the same system where the application server runs).
   c. Specify the port in which the OLT server listens to the events occurring within the application server (default 2102).

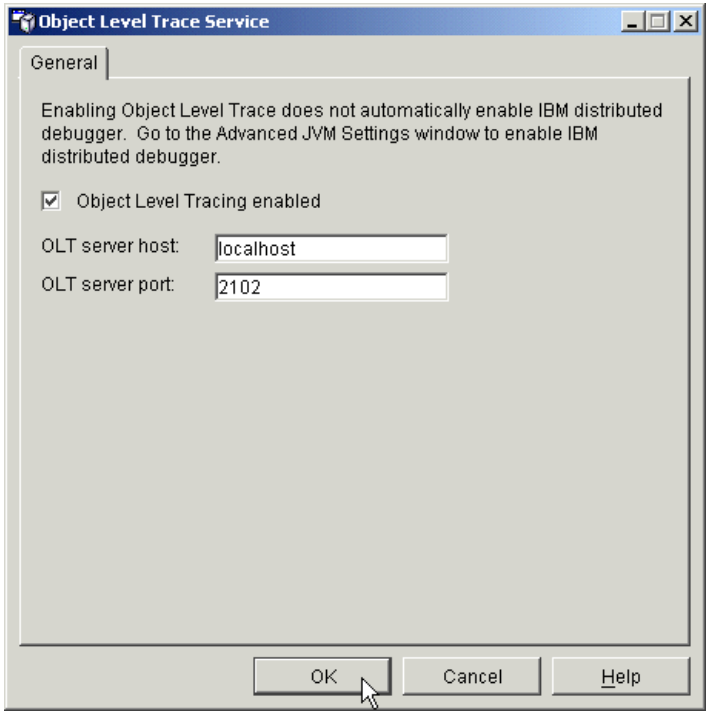


Figure 24-17 Object Level Trace Service window

This enables the application server for object level tracing only.

Activating Object Level Debugger

To enable the application server for both tracing and debugging, you need to activate the Object Level Debugger.

1. In the Application Server Properties window, select the **JVM Settings** tab.
2. Click the **Advanced JVM Settings** button, as shown in Figure 24-18.

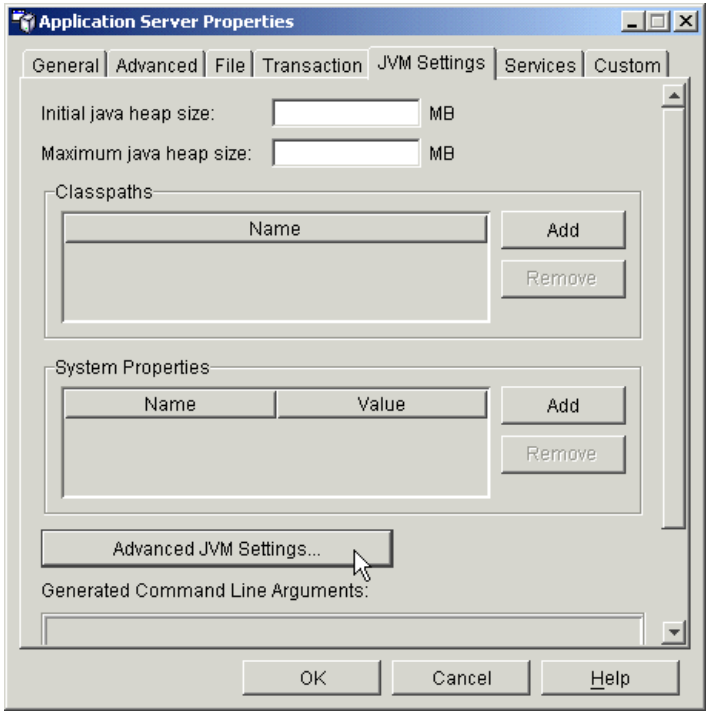


Figure 24-18 Application Server Properties - JVM Settings

3. On the Advanced JVM Settings window, shown in Figure 24-19:
   a. Check the **Enable IBM distributed debugger** box.
   b. Specify the Source path where the source code is located. The source code can be located either where the application is running, the application server machine, or at the debug viewer's site. If you don't specify a path, or if the code is not found at runtime, the OLD will prompt you for the source code location.

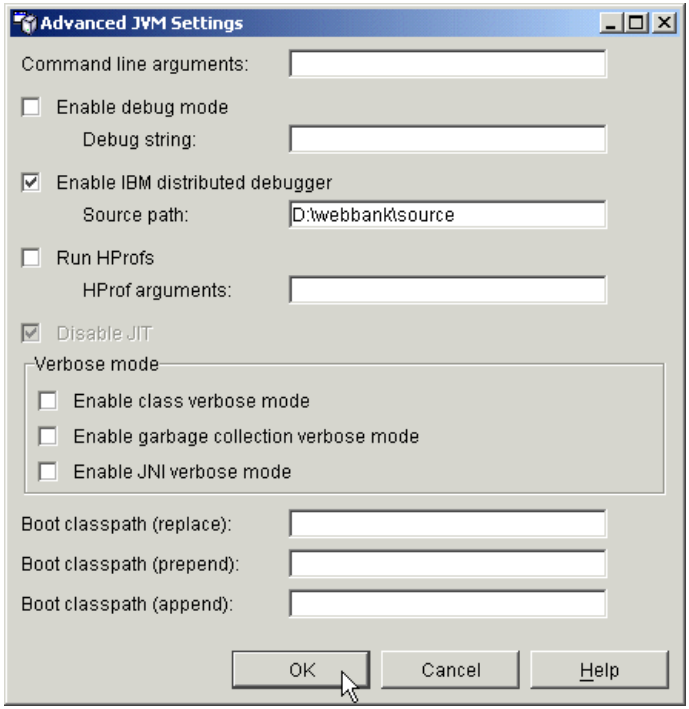


Figure 24-19 Advanced JVM Settings window

4. Click **OK** in the Advanced JVM Settings window.
5. Click **OK** in the Application Server Properties window.

Using OLT/OLD

To start troubleshooting using OLT/OLD:

1. Start OLT on the machine where you want to view the trace using the following command:

   ```
   C:\IBMDebug\bin\olt
   ```

2. In the OLT Viewer window, shown in Figure 24-20, set the following parameters:

   a. Select the **Trace and debug** option in the Execution Mode field.

   The various executions modes are:

   - **No trace and debug**. Use this mode when you do not want to trace or debug your application.
   - **Trace only** (default). As your application runs, OLT monitors events and creates a trace.

- **Debug only**. This mode enables you to debug your application without producing a trace. In this mode, the debugger steps into every debuggable method without first prompting you.
- **Trace and debug**. This mode provides access to both the trace facility and the distributed debugger. It also gives you the greatest control over the debugging process.

In debug modes, OLT operates in conjunction with the distributed debugger, so that breakpoints set in OLT trigger the debugger and we can see at the same time the flow of the application and the possible code errors.

b. Enter the host name and TCP/IP port of the workstation where you want the debugger to open.

c. Click the **Apply** button in order for options to be set.

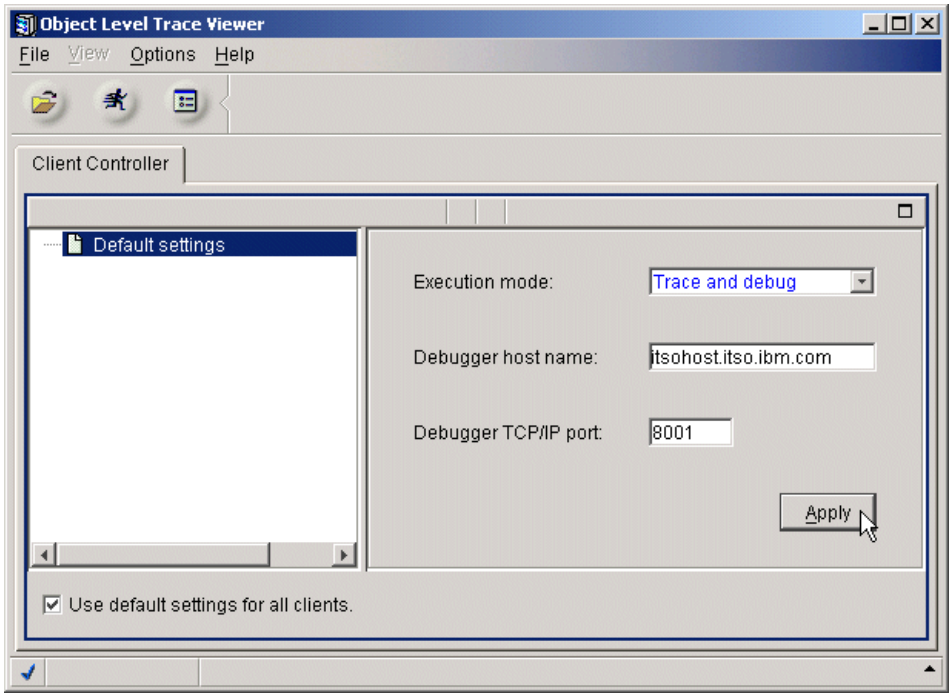


Figure 24-20 Object Level Trace Viewer

3. In the Options menu, verify that **Step-by-step Debugging Mode** is selected.
4. Select **Options** -> **Online Mode** from the menu. A new tab called OLT Trace is added to the OLT viewer.

5. Start your application server. It will connect to the OLT Client Controller, as seen in Figure 24-21, and tracing will stop in the first method breakpoint.

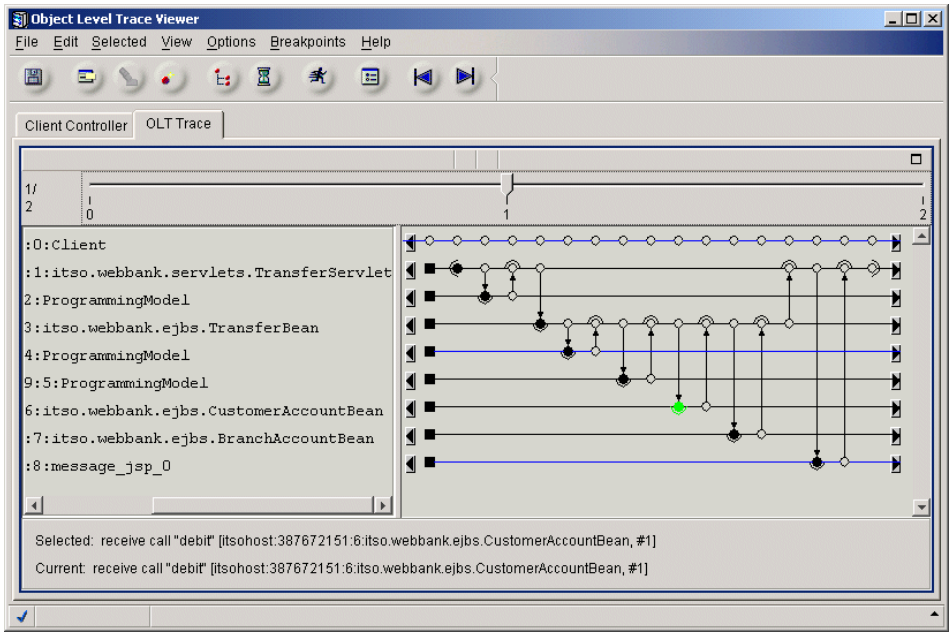


Figure 24-21 OLT trace

If you selected the step-by-step trace and debug mode, OLT automatically launches the debugger on the designated workstation when it reaches the first debuggable method, as shown in Figure 24-22.

OLT then asks you whether you want to step into or over the method.

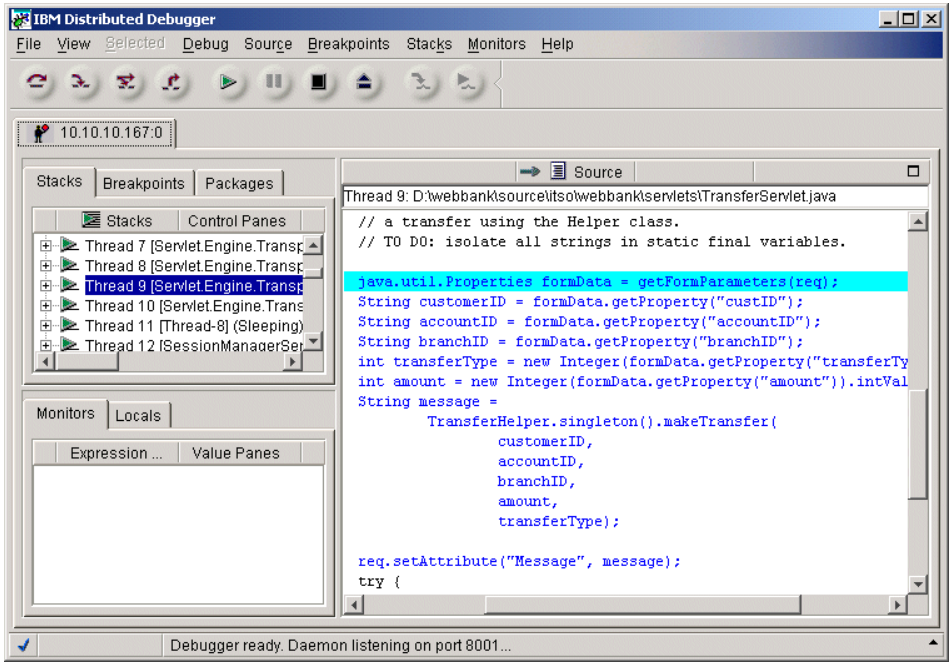


Figure 24-22 Distributed debugger

For information on how to start the debugger on a different workstation and more information on the usage of OLT/OLD refer to the *WebSphere Version 4 Application Development Handbook*, SG24-6134, the documentation in InfoCenter, or the OLT/OLD documentation.

24.8.2 JRas

Applications often need to provide information about their internal operations to users, system administrators, programmers, and other interested parties. This information is typically provided as text that can be sent to a console or terminal, written to a log file, directed to a standard output or error device, or all three.

The IBM JRas toolkit is a set of Java packages that enables developers to incorporate message logging and trace facilities into Java applications.

Although JRas is a stand-alone product, it has been customized for use with the WebSphere Application Server V4.0, Advanced Edition. The WebSphere JRas implementation is integrated with the internal logging system used by the application server code and has been present in the product for some time.

If you use JRas to log application events, then messages from the application will be fully integrated with messages from the application server. Severe messages from the application will be reported to the WebSphere administrative console. Messages will be collated in the same place and you will be able to use the same tools to control trace information from both sources.

This integration can assist with debugging, particularly if you are concerned about how your application interacts with the application server, since messages from your application and WebSphere are interleaved in the same location, in the correct sequence and using the same message format.

When we log application messages using the WebSphere JRas we are able to use the same WebSphere resources to control the logging of messages from the application as we use to control logging of messages from WebSphere itself. These WebSphere resources were discussed in 24.2, “Resources for identifying problems” on page 951.

To view messages and trace text, you must read the appropriate log files.

The JRas packages implement objects called *loggers*, *handlers*, *formatters* and *managers* to provide messaging and trace capabilities, as shown in Figure 24-23.

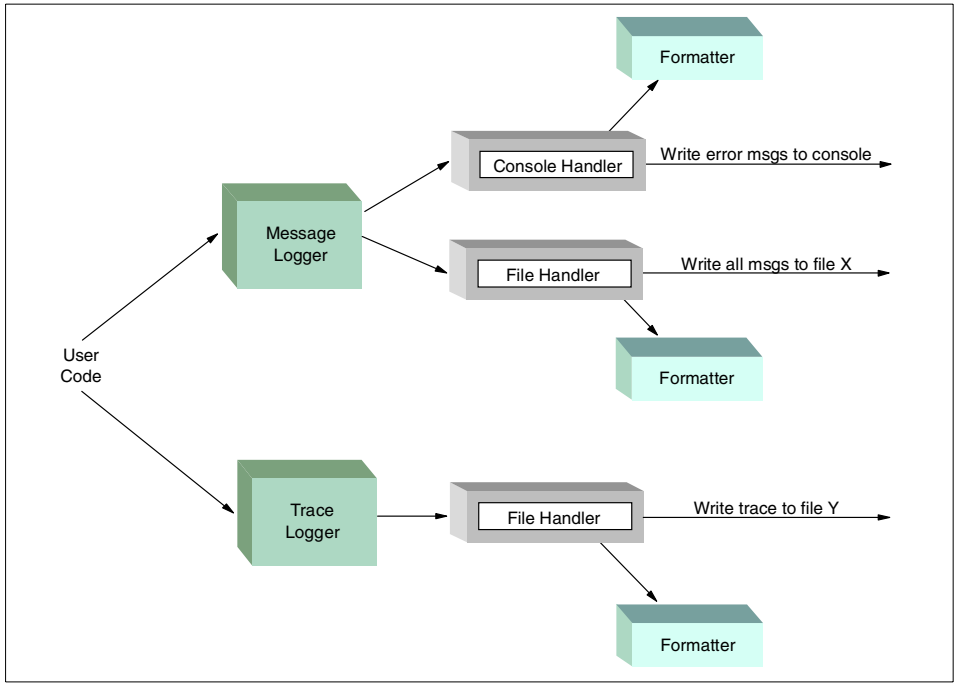


Figure 24-23 Sample JRas logger configuration

It is beyond the scope of this book to discuss the WebSphere implementation of JRas and the JRas programming model. Check the *WebSphere Version 4 Application Development Handbook*, SG24-6134 for code examples, or the documentation in InfoCenter.

24.9 Other tracing

Tracing can have a serious impact on performance and should never be used in a normally functioning environment. However if there is a need for it (in most cases when requested by IBM personnel), you can also enable ORB tracing, plug-in tracing, or SAS tracing as described in the following sections.

24.9.1 Enabling ORB tracing

An Object Request Broker (ORB) manages interaction between clients and servers. It "brokers" a client request for a service from a distributed object or component.

The interfaces and services that an ORB must provide are defined by CORBA. Under CORBA, one CORBA object never talks directly with another. Instead, an object requests an interface to the ORB running on the local machine. The local ORB then passes the request to an ORB on another machine. The remote ORB then locates the appropriate object and passes back an object reference to the requesting object. Having ORB support in a network means that a client program can request a service without having to understand where the server is in a distributed network or what the interface to the server program specifies.

If you have ORB communication problems, such as JORB errors in any of the log files, you can enable ORB tracing. ORB tracing is done in two parts:

- Enabling CORBA.CommTrace allows you to see the IIOP messages that are being exchanged.
- Tracing the ORBRas captures the flow of execution through the ORB code.

You will want to get both for debugging purposes.

In the following sections we explain:

- Tracing the WebSphere administrative server
- Tracing an application server
- Tracing the administrative console

Tracing the WebSphere administrative server

To trace the WebSphere administrative server from startup, follow these steps:

1. Stop the WebSphere node by right-clicking the required node in the console tree view and selecting **Stop** from the pop-up menu.
2. Make a backup copy of the admin.config file, <WAS_HOME>/bin/admin.config.
3. Add the following lines to the admin.config file:

   ```
   com.ibm.CORBA.Debug=true
   com.ibm.CORBA.CommTrace=true
   com.ibm.ejs.sm.adminServer.traceString="ORBRas=all=enabled"
   com.ibm.ejs.sm.adminServer.traceOutput=<trace_file_name>
   ```

   Where `<trace_file_name>` is D:/WebSphere/AppServer/logs/admin.trace, for example.
4. Start the administrative server.
5. View the resulting trace file, D:/WebSphere/AppServer/logs/admin.trace, in our case.

Tracing an application server

To enable ORB tracing for an application server, follow these instructions:

1. Locate the required application server in the console tree view.
2. Right-click the application server and select **Properties** from the pop-up menu.
3. In the Application Server Properties window, select the **Services** tab.
4. In the service list, select **Object Request Broker** and click the **Edit properties...** button.
5. In the Object Request Broker window, check the **Enable ORB tracing** box, as shown in Figure 24-24.

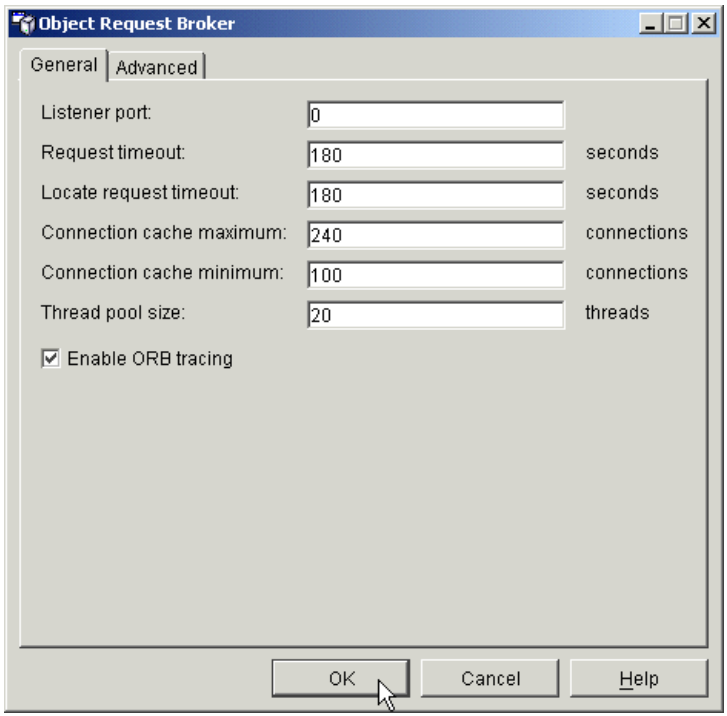


Figure 24-24 Object Request Broker window

6. Click **OK** in the Object Request Broker window.
7. Back in the service list, select the **Trace Service** and click the **Edit properties...** button.
8. In the Trace Service window, set the Trace specification to “ORBRas=all=enabled”, as shown in Figure 24-25.

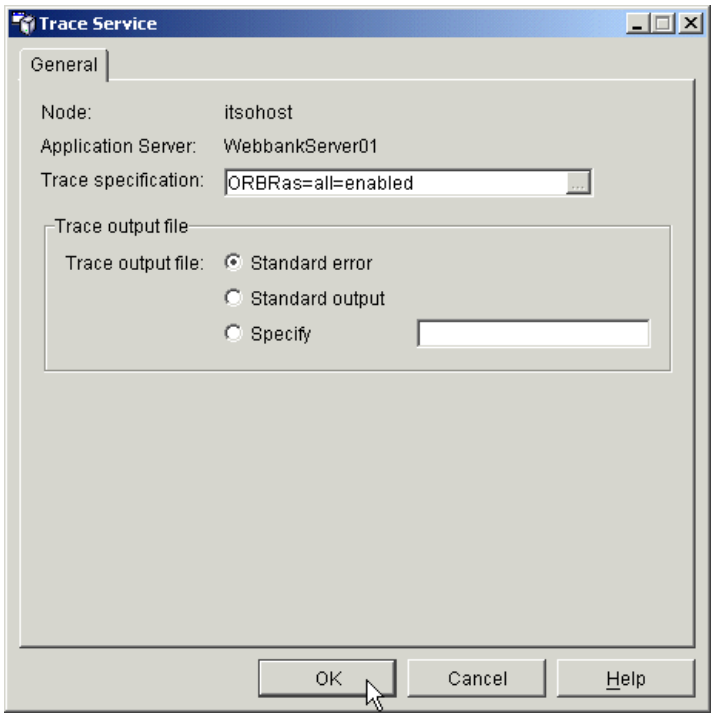


Figure 24-25 Trace Service window

9. Click **OK** in the Trace Service window.
10. Click **OK** in the Application Server Properties window.
11. Restart the application server for the change to take effect.
12. Review the standard error file for the application server (the default, unless there is a Trace output file specified via the Trace Service).

Tracing the administrative console

To enable ORB tracing for the administrative console, follow these instructions:

1. Go to the <WAS_HOME>/bin directory and make a backup copy of adminclient.bat/sh file.
2. Edit the adminclient.bat/sh file as follows:
   a. Locate where the `DEBUGOPTS` environment variable is set to:

      ```
      -traceString "com.ibm.*=all=enabled"
      ```

      and modify the `traceString` parameter as follows:

      ```
      -traceString "com.ibm.*=all=enabled:ORBRas=all=enabled"
      ```

b. Locate the line starting with:

- On Windows:

```
%JAVA_HOME%\bin\java ...
```

- On UNIX:

```
$JAVA_HOME/bin/java ...
```

and add the following two parameters:

```
-Dcom.ibm.CORBA.Debug=true
-Dcom.ibm.CORBA.CommTrace=true
```

3. Make sure the administrative server has been started.
4. From the <WAS_HOME>/bin directory, start the administrative console as follows:
   – On Windows:

```
adminclient debug > adminclient.trace
```

   – On UNIX:

```
./adminclient.sh debug 2>&1 | tee adminclient.trace
```

5. View the resulting trace file, <WAS_HOME>/bin/adminclient.trace. See also <WAS_HOME>/logs/adminclient_audit_messages.log on UNIX.

24.9.2 Plug-in tracing

If you have problems that relate to the WebSphere plug-in, you can turn on tracing using the plugin-cfg.xml file. To turn on plug-in tracing:

1. Stop the Web server.
2. Open the <WAS_HOME>/config/plugin-cfg.xml file for editing.
3. Locate the Log element as follows:

```
<Log LogLevel="Error" Name="<WAS_HOME>/logs/native.log"/>
```

   and set the LogLevel attribute to Trace as follows:

```
<Log LogLevel="Trace" Name="<WAS_HOME>/logs/native.log" />
```

   `Name` is the full path to the file that you want the plug-in to write messages to.

   `LogLevel` is the level of detail that you want written to the log file. The expected values are `Trace`, `Warn` or `Error`. `Trace` allows you to see the request processing steps in detail. `Warn` and `Error` means that only information about abnormal request processing will be logged.

4. Save the new configuration file.
5. Restart the Web server.

6. Send an HTTP request from your browser.
7. Review the information in the plug-in log file (native.log by default).

See Chapter 14, “Configuring the Web server interface” on page 481 for more information on the WebSphere Web server plug-in.

Note: A lot of messages are logged at Trace level, which can cause the disk space to fill up very quickly.

24.9.3 SAS tracing

When global security is enabled in WebSphere Application Server, all requests from clients to Enterprise JavaBeans are sent as RMI/IIOP messages via the Object Request Broker (ORB) to the server that hosts the enterprise beans. As part of every such request and response, the ORB invokes the Secure Association Service (SAS) on the client and the server sides. On the client side, SAS intercepts requests before they are sent, obtains the client's security credentials, attaches the credentials to the request as part of the security context, and sends the request. On the server side, SAS intercepts the incoming request, extracts the security context from the message, authenticates the client's credentials, and passes the request to the enterprise bean container, where the request is authorized.

Collecting information about SAS messages is often crucial for debugging security problems. SAS provides a set of properties that govern the collection of SAS messages, including type of messages and the destination of the collected messages. These properties are set in the property file sas.server.props, which is used by the administrative server and secured application servers. See Chapter 21, “Configuring security” on page 739 for more information on securing WebSphere.

When you have a security problem, it is best to both enable Secure Association Services (SAS) tracing and to trace com.ibm.ejs.security in all components that might be affected. SAS is typically used for authentication of Java clients. Security is typically used for all authorization and Web authentication.

Enabling SAS tracing

To enable Secure Association Service tracing:

1. Stop the WebSphere node by right-clicking the required node in the console tree view and selecting **Stop** from the pop-up menu.
2. Delete the <WAS_HOME>/properties/sas.server.props.future file. If this file is present when the server restarts, information in the sas.server.props.future

file is copied into the sas.server.props file, overwriting your changes to the sas.server.props file.

3. Open the <WAS_HOME>/properties/sas.server.properties file for editing.
4. Set the following properties:
   - **debug flag**. Possible values are false and true:

     ```
     com.ibm.CORBA.securityDebug=true
     ```

   - **trace level**. Possible values are none, basic, intermediate and advanced:

     ```
     com.ibm.CORBA.securityTraceLevel=advanced
     ```

   - **output mode**. Possible values are none, console, file and both:

     ```
     com.ibm.CORBA.securityTraceOutputMode=file
     ```

   - **output file**. If the output target is file, then the target file name must be specified:

     ```
     com.ibm.CORBA.securityTraceOutput=<trace_file_name>
     ```

     Where <trace_file_name> is D:/WebSphere/AppServer/logs/sas_server.log, for example.
5. Save your changes
6. Repeat steps 1 to 5 for the <WAS_HOME>/properties/sas.client.properties file.
7. Before restarting the node, we recommend that administrative server or application server com.ibm.ejs.security tracing be enabled.

Tracing WebSphere administrative server security

To enable com.ibm.ejs.security tracing for the administrative server, follow these instructions:

1. Stop the WebSphere node by right-clicking the required node in the console tree view and selecting **Stop** from the pop-up menu.
2. Make a backup copy of the admin.config file, <WAS_HOME>/bin/admin.config.
3. Add the following lines to the admin.config file:

   ```
   com.ibm.ejs.sm.adminServer.traceString="com.ibm.ejs.security.*=all=enabled"
   com.ibm.ejs.sm.adminServer.traceOutput=<trace_file_name>
   ```

   Where <trace_file_name> is D:/WebSphere/AppServer/logs/admin.trace, for example.
4. Start the administrative server.
5. View the resulting trace files, in our case:
   - D:/WebSphere/AppServer/logs/admin.trace

- D:/WebSphere/AppServer/logs/sas_server.log
- D:/WebSphere/AppServer/logs/sas_client.log

Tracing application server security

To enable com.ibm.ejs.security tracing for an application server follow these instructions:

1. Locate the required application server in the console tree view.
2. Right-click the application server and select **Properties** from the pop-up menu.
3. In the Application Server Properties window, select the **Services** tab.
4. In the service list, select **Trace Service** and click the **Edit properties...** button.
5. In the Trace Service window, set the Trace specification to “com.ibm.ejs.security.*=all=enabled”.
6. Click **OK** in the Trace Service window.
7. Click **OK** in the Application Server Properties window.
8. Restart the application server for the change to take effect.
9. View the resulting trace files, in our case:
   - The standard error file for the application server
   - D:/WebSphere/AppServer/logs/sas_server.log
   - D:/WebSphere/AppServer/logs/sas_client.log

24.10 Log Analyzer

The Log Analyzer is a GUI tool that permits the user to view any logs generated with loganalyzer TraceFormat, such as the activity.log file and other traces using this format, as discussed in 24.6, “Traces” on page 965. It can take one or more activity or trace logs, merge all the data, and display the entries in sequence.

More importantly, this tool is shipped with an XML database, the *symptom database*, which contains strings for some common problems, reasons for the errors, and recovery steps. The Log Analyzer compares every error record in the log file to the internal set of known problems in the symptom database and displays all the matches. This allows the user to get error message explanations and information such as why the error occurred and how to recover from it, as shown in Figure 24-26.

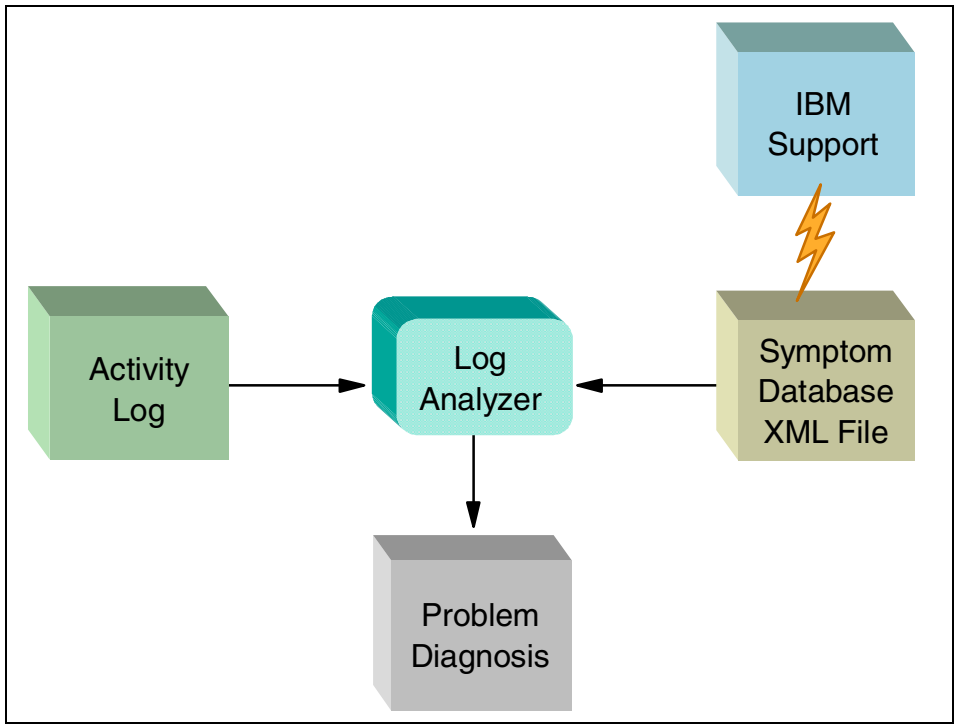


Figure 24-26 Log Analyzer

Note: It is recommended that you update your symptom database from time to time, as the database will get updated by IBM support with new messages and recovery instructions all the time. See “Updating the symptom database” on page 1005 for details.

To start using the Log Analyzer:

1. Select **Tools** -> **Log Analyzer** from the administrative console main menu, as shown in Figure 24-27.

   The Log Analyzer can also be started from the <WAS_HOME>/bin directory using the **`waslogbr.bat/sh`** command.

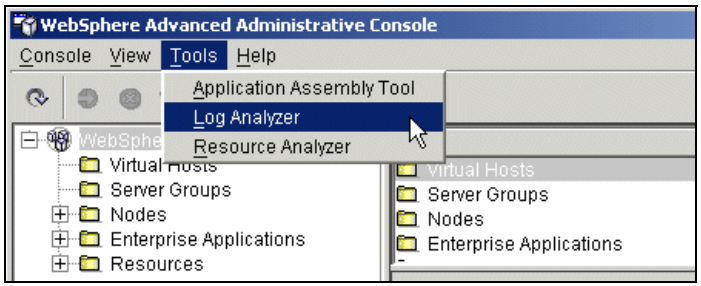


Figure 24-27 Starting Log Analyzer from the console

2. When the Log Analyzer GUI starts, select **File** -> **Open...** from the main menu. Navigate to the <WAS_HOME>/logs directory, select the activity.log (or another trace file using the loganalyzer TraceFormat), and click **Open**.

You should now see the open activity log, similar to the view in Figure 24-28. As discussed in 24.5.6, “Activity log” on page 963, all application servers and the administrative server write log records to this file.

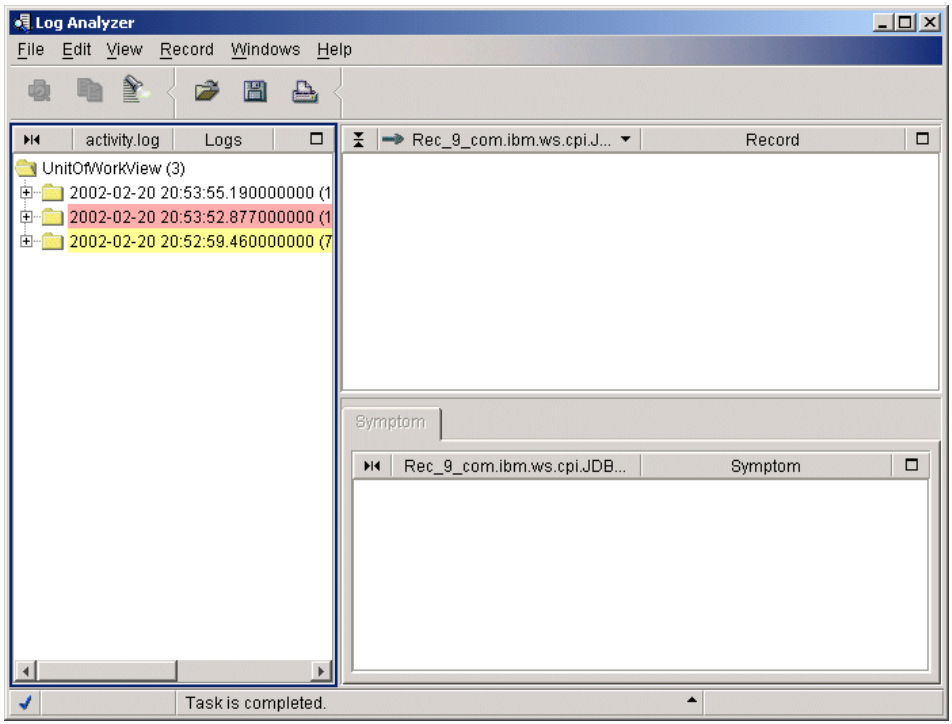


Figure 24-28 Log Analyzer

3. To see the record details for a log entry, click entry under a unit of work folder.
4. To analyze a log entry, right-click the entry and select **Analyze** from the pop-up menu, as shown in Figure 24-29.

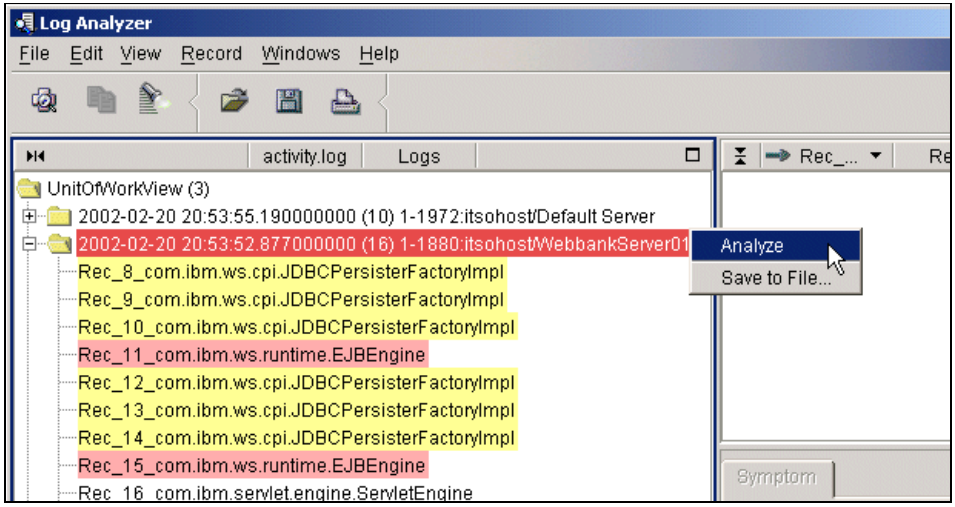


Figure 24-29 Selecting an entry to analyze

After the analyze action has been invoked, each analyzed log entry has an icon indicating whether analysis information is available. The Tick icon next to our selected entry in Figure 24-30 indicates that analysis information is available in the analysis pane.

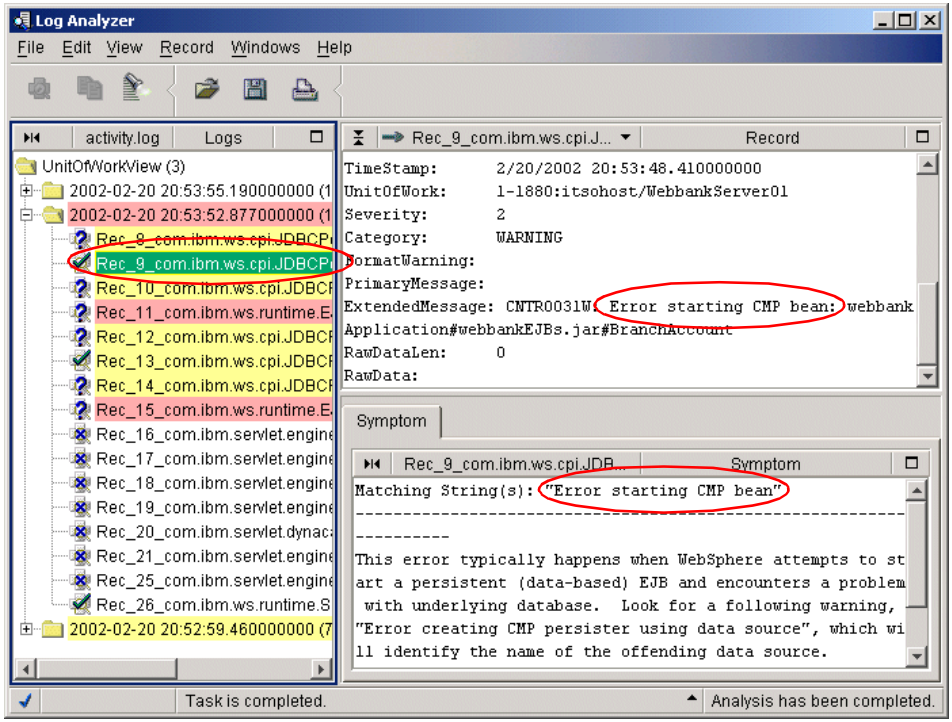


Figure 24-30 Log Analyzer information

As shown in Figure 24-30, the Log Analyzer's main window has three panes:

- Logs pane, on the left.
- Record pane, on the upper right.
- Analysis pane, on the lower right.

Logs pane

By default, Log Analyzer's logs pane displays log entries by unit of work. It lists all the unit of work (UOW) instances and its associated entries from the logs that you have opened. The UnitOfWorkView can be useful when you are trying to find related entries in the activity log or when you are debugging problems across multiple machines.

Under the root UnitOfWorkView folder is a folder for each UOW, which you can expand to show all the entries for that UOW. All log entries without a UOW identification are grouped into a single folder. The UOW folders are sorted to show the UOW with the latest timestamp at the top of the list. The entries within

each UOW are listed in the reverse sequence, that is the first (earliest) entry for that UOW is displayed at the top of the list. If you have merged several logs in the Log Analyzer, all the log entries are merged in timestamp sequence within each UOW folder, as if they all came from the same log.

Each UOW line has the following format:

2002-02-20 20:53:52.877000000 (16) 1-1880:itsohost/WebbankServer01

Where:

- 2002-02-20 20:53:52.877000000 is the timestamp.
- (16) is the number of entries.
- 1-1880:itsohost/WebbankServer01 is the unit of work.

Click the + icon next to the UOW folder to see all the log entries for the UOW. Each log entry's identification has the following format:

Rec_9_com.ibm.ws.cpi.JDBCPersisterFactoryImpl

Where:

- Rec_9 is the entry number.
- com.ibm.ws.cpi.JDBCPersisterFactoryImpl is the class name.

Every log entry is assigned an entry number, Rec_nnn, when a log is opened in the Log Analyzer. If more than one file is opened in the Log Analyzer (merged files), the Rec_nnn identification will not be unique because the number is relative to the entry sequence in the original log file and not to the merged data that the Log Analyzer is displaying. This Rec_nnn also appears in the first line in the Records pane.

By default, each entry in this pane is color-coded to help you quickly identify the ones that have high severity errors.

Non-selected log entries have a background color of:

- Pink if it has a severity 1 error.
- Yellow if it has a severity 2 error.
- White if it has a severity 3 error.

Selected log entries have a background color of:

- Red if it has a severity 1 error.
- Green if it has a severity 2 error.
- Blue if it has a severity 3 error.

These colors are configurable and can be changed in the Log Analyzer's Preferences Log page. Select **File** -> **Preferences...** -> **Logs** -> **Severity**.

The Log Analyzer can also display the log entries in different sorting sequences. Select **File** -> **Preferences** -> **Logs**.

After the analyze action has been invoked, each analyzed log entry has the following icons:


The tick icon indicates that the entry has some analysis information in one or more pages in the analysis pane.


The plus icon indicates that the entry has some analysis information and that it has a re-raised or re-mapped exception. You may want to look at the log entry prior to this one when diagnosing problems.


The question mark icon indicates that the entry has either a severity 1 or 2 error but no additional analysis information is available for it.


The cross icon indicates that the entry has a severity 3 error and it has no analysis information.

Record pane

When you select an entry under the unit of work in the logs pane, you see the details of the entry in the record pane (ProcessId, ThreadId, SourceId, FunctionName, ExtendedMessage, among others). The entry's identification is shown in the pane's title bar. Right-click in this record pane to see the actions that you can perform on the entry (Analyze, Save to File..., Find..., Select All).

There is a drop-down arrow next to Record in the pane’s title bar, which allows you to go back to look at the last ten records that you have viewed. The default cache size for the historical data is 10. Select **File** -> **Preferences...** -> **General**.

Analysis pane

When the analyze action has been invoked, if information is found in the symptoms database, <WAS_HOME>/properties/logbr/symptoms/adv/symptomdb.xml, for the selected log entry, the information will appear in the symptom page. If the page tab is grayed out, there is no information in that page.

There is a status line at the bottom of the window showing the status of actions.

Merging activity logs

To merge different activity.log files from different machines where your transaction occurred:

1. Make sure that <WAS_HOME>/properties/logging.properties file includes the following:

   ```
   com.ibm.ws.ras.UnitOfWork=true
   ```

   The property com.ibm.ws.ras.UnitOfWork allows you to specify whether or not a correlation ID should be generated and included in message events and diagnostic trace entries. If set to true, each application client request is assigned a unique identifier that is propagated to all servers touched as part of servicing that request. This allows correlation of events across multiple server processes.

2. Open one of the files in Log Analyzer and select **File** -> **Preferences...** -> **Logs** to sort the log records in UnitOfWork and TimeStamp order to have a distributed log view.

3. Use the **File** -> **Merge with...** option to merge files.

Updating the symptom database

The latest symptom database is available from the IBM Web site:

ftp://ftp.software.ibm.com/software/websphere/info/tools/loganalyzer/symptoms/adv/symptomdb.xml

Place the symptomdb.xml file in the <WAS_HOME>/properties/logbr/symptoms/adv directory on your workstation.

Alternatively, download the symptoms database using the Log Analyzer GUI. Select **File** -> **Update Database** -> **Advanced Symptom Database** from the main menu, as shown in Figure 24-31.

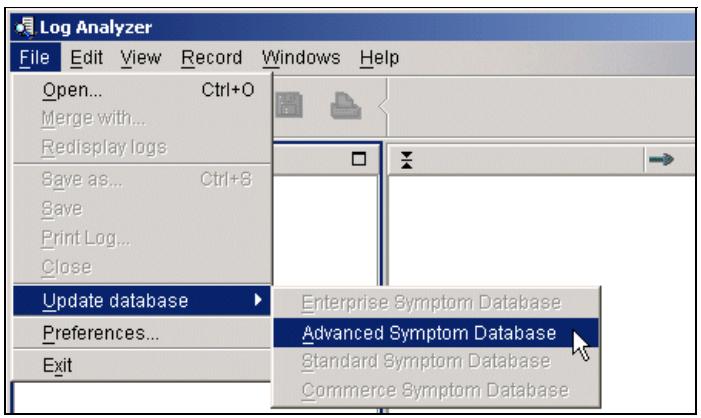


Figure 24-31 Updating the symptom database

If you are behind a firewall, then you need to edit the <WAS_HOME>/bin/waslogbr.bat file to include the proxy, for example:

```
...
%JAVA_HOME%\bin\java -DIVB_HOME="%USERPROFILE%/logbr" ^
   -DPROXY=proxy.itso.ibm.com:8080 ^
   -DIVB_DRIVER_PATH="%TEMPIVBDIR%" ^
...
```

Symptom databases are edition-specific.

24.11 Other tools

You can use WebSphere Application Server servlets and other tools to help diagnose problems. In this section we describe some of them.

Also see the InfoCenter article “Using internal tools“.

24.11.1 DumpNameSpace

The name space stored by a given name server can be dumped with the DumpNameSpace utility that is shipped with WebSphere Application Server. This utility can be invoked from the command line or from a Java program. The naming service for the WebSphere Application Server host must be active when this utility is invoked.

To invoke this utility using the class com.ibm.websphere.naming.DumpNameSpace API, see the API documentation in the InfoCenter.

To invoke the utility through the command line, enter the following command from the <WAS_HOME>/bin directory:

- UNIX:

  ```
  dumpNameSpace.sh [[-keyword value]...]
  ```

- Windows:

  ```
  dumpNameSpace [[-keyword value]...]
  ```

The following command shows how to invoke the DumpNameSpace utility from the command line:

```
dumpNameSpace -host myhost.mycompany.com -port 901
```

This command will dump the name space for the WebSphere Application Server on host myhost.mycompany.com, bootstrap port 901. Refer to the InfoCenter for a full list of the keywords and associated values used with the dumpNameSpace utility.

The generated output will look like Example 24-6, which is the *short* dump format.

Example 24-6 dumpNameSpace output

```
Getting the initial context
Getting the starting context

================================================================================
Name Space Dump
   Provider URL: iiop://localhost:900
   Context factory: com.ibm.websphere.naming.WsnInitialContextFactory
   Requested root context: legacy
   Starting context: (top)=requested root context
   Formatting rules: jndi
   Time of dump: Wed Feb 20 23:58:33 EST 2002
================================================================================


================================================================================
Beginning of Name Space Dump
================================================================================

    1 (top)
    2 (top)/itsohost                                    javax.naming.Context
    3 (top)/itsohost/resources                          javax.naming.Context
    4 (top)/itsohost/resources/sec                      javax.naming.Context
...
   55 (top)/webbank                                     javax.naming.Context
   56 (top)/webbank/Transfer
itso.webbank.ejbs._TransferHome_Stub
   57 (top)/webbank/Consultation
itso.webbank.ejbs._ConsultationHome_Stub
...

================================================================================
End of Name Space Dump
================================================================================
```

24.11.2 Sample applications

Table 24-5 describes sample applications that can be used as debug tools. If the samples are not already installed, then install the sampleApp.ear enterprise application located in the <WAS_HOME>/installableApps directory.

Table 24-5 Sample applications

Servlet	Location	Description
HitCount	http://localhost/webapp/examples/HitCount	Verifies correct operation of servlets, JSP files, enterprise beans, and HTTP session.
Snoop	http://localhost/servlet/snoop	Useful for examining request parameters coming from the client and for verifying the operation of the Web container.
ShowCfg	http://localhost/webapp/examples/showCfg	Useful for validating the configuration of the system.
BeenThere	http://localhost/webapp/examples/BeenThere	Useful for demonstrating and testing WebSphere WLM, both “Plug-in WLM” (for servlet requests) and “EJS WLM” (for EJB requests).

Note: If you are planning to use BeenThere to test workload management, it is recommended that you have a look at the README that comes with it, (`http://localhost/webapp/examples/beenthere.html`). The README contains instructions on how to create and test different configurations.

24.11.3 E-fix installer

The e-fix installer is a tool used by IBM developers to create executable e-fix JAR files, which can then be used to apply e-fixes. The aim is to make the process of installing e-fixes as easy as installing fixpaks, eliminating the need to install e-fixes manually and establishing a stable process by which changes can be delivered.

In versions prior to WebSphere V4.0, e-fixes were sent to customers and applied to the front of the WebSphere classpath. This ensured that the new class was found before the original version of the class.

The e-fix installer uses a filter file created by a developer to build an executable JAR file. The executable JAR file is sent to the customer as part of a ZIP or TAR file. The customer executes the JAR file and the files, classes, scripts, and so on are applied directly to the customer's WebSphere environment. There is no need to update classpaths with e-fix JARs. The class updates are applied directly to the system JARs.

The e-fix installer provides the following benefits:

- Less likelihood of errors due to manual application of e-fixes.
- Changes are applied to the system and not the classpath.
- Detailed logging during installation. The log files can be found in <WAS_HOME>/eFix/<APAR number>.
- It updates the product.xml file to track the level of fixpaks and e-fixes. The product.xml file contains all history events, including e-fix installs and e-fix removals.
- It does an automatic backup before the installation starts. The backup files can be found in <WAS_HOME>/eFix/<APAR number>. These files should not be removed. If the backup files are removed, it will not be possible to remove the e-fix.
- It proceeds with environment safety checks such as version, edition and classpath.
- Allows fixpak installer to uninstall e-fixes prior to fixpak install (obsoletes manual uninstall of e-fixes).

Note: E-fixes are individual fixes for critical problems. They have been individually tested, but not integration tested and should only be applied if you have a critical problem without a valid workaround. All e-fixes are rolled into the next scheduled fixpak.

24.12 What kind of problem do I have?

This section provides an approach to problem determination depending on the type of problem that you have. However it is recommended that you check the "Problem determination quick reference" article in the InfoCenter, which provides a quick reference to help identify and solve your problems.

WebSphere problems can be broadly classified under one of the following tasks:

- WebSphere installation
- Starting WebSphere

- Configuration application
- Accessing application

We cover troubleshooting of these tasks in the following sections.

24.12.1 WebSphere installation problems

Successful installation means that no errors occur during the install process and, more importantly, that the product runs correctly the first time you start it.

In this section we look at problems where the installation hangs, displays an error message, or finishes but components are missing.

Install hangs

1. Check <WAS_HOME>\logs\wssetup.log on Windows, or <WAS_HOME>/logs/install.log on UNIX.
2. If wssetup.log does not display errors, look at the command session from which the installation was launched. In a Windows NT environment, this means executing **`setup.exe`** from a command window instead of double-clicking it from within Windows Explorer. When the installation hangs, look at the window, does it display an operating system error or Java exception?
3. Check that the user ID, under which the installation is running, has the authority to read the installation files or to write to (or overwrite files in) the target installation directory.
4. Installation can be unsuccessful if WebSphere Application Server and related processes have not been stopped before installation. Stop the related processes before installing WebSphere Application Server (all but for migration installs). Web servers and Web administrative servers should also be stopped.

Install displays an error message

If the installation fails with an error message, the message will be an indicative of the type of problem:

1. **Insufficient disk space**. Check that the drive has enough available space. On UNIX, check that there is sufficient available space in the WebSphere Application Server file system as well as in the /tmp directory.
2. **Insufficient software prerequisite level**. Check that the supporting software is at the right level. Visit the IBM WebSphere Application Server prerequisites site for the latest prerequisites:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

If you plan to use a Web server or database at a level that exceeds the current version required by WebSphere Application Server, you need to disable the WebSphere Prerequisite Checker before installing WebSphere Application Server:

a. Copy the prereq.properties file from the product CD to a temporary directory.

b. Edit the file by changing the value of the `prereq_checker` parameter from 1 to 0.

c. Pass the prereq.properties file to the installation program using the -prereqfile argument, for example on Windows 2000:

```
setup.exe -prereqfile <tmp>/prereq.properties
```

Refer to 8.2.5, “Customize prerequisite checking” on page 143 for further details.

3. **File permission problems**. Check that the user ID under which the installation is running has the authority to read the installation files or to write to (or overwrite files in) the target installation directory.

   Make sure that WebSphere Application Server and related processes are stopped before installation.

Installation finishes but components are missing

If the installation is complete, but the product directories have not been created, there are files missing, or Windows services entries or Start menu icons have not been added:

1. Check <WAS_HOME>\logs\wssetup.log on Windows, or <WAS_HOME>/logs/install.log on UNIX.
2. Look for errors indicating that the installation program was unable to read from a file, write to a file, or overwrite a file (see the file permission problems above).

Refer also to Part 3, “Installing WebSphere” on page 125.

24.12.2 Administrative server fails to start after installation

After installing WebSphere Application Server for the first time, try starting the administrative server as the WebSphere administrative user (root on UNIX and wasadmin on Windows, for example).

Successfully starting the administrative server for the first time indicates that the installation of WebSphere Application Server completed successfully. It also means that the following tasks were completed:

- System management repository tables were created in the database.
- A node object and host aliases were defined in the repository.

Therefore, if the administrative server cannot be started after WebSphere is installed, check the following first:

- Check that there are no errors in the installation logs (<WAS_HOME>\logs\wssetup.log on Windows, or <WAS_HOME>/logs/install.log on UNIX).
- Check that the <WAS_HOME> directory has been created.
- On Windows, check that the WebSphere administrative server entry in the Services panel is not missing.
- On Windows, check that there are no items missing from the WebSphere Application Server Start menu.

If any of the above is true, you might want to uninstall and install WebSphere again.

Otherwise, if the installation is complete, check the following sections for possible reasons.

Note: WebSphere Application Server will log the message **"**`open for e-business`**"** in the <WAS_HOME>/logs/tracefile log when up and running.

The port is in use

WebSphere Application Server will fail to start if certain ports are in use. Typical port problem descriptions follow:

1. When the bootstrap port is in use, you may see the following error when starting WebSphere:

   ```
   009.765.6005c5b F Nameserver Failed to start the Bootstrap server
   org.omg.CORBA.INTERNAL: minor code: 8 completed: No
   ```

   To fix the problem change the bootstrap port entry (the default is 900) in the admin.config file:

   ```
   com.ibm.ejs.sm.adminServer.bootstrapPort=<bootstrap_port>
   ```

   If the property does not exist add it.

2. Port 9000 is the default port of the administrative server location service daemon. Port 9000 is also used by many system resources including AIX X-windows manager. If you see the error message:

   ```
   Port 9000 in use-select another port
   ```

   when executing the ./startupServer.sh command on AIX, the administrative server process cannot start because the port 9000 is in use. You can change the port the location service daemon listens on by specifying the `-lsdPort` option on the administrative server command line or by setting the com.ibm.ejs.sm.adminServer.lsdPort property in the admin.config file:

   ```
   com.ibm.ejs.sm.adminServer.lsdPort=<location_service_daemon_port>
   ```

Another administrative server is running

Verify the WebSphere java processes that are running using the Task Manager on Windows and **`ps -ef`** on UNIX. You should see one Java process for the administrative server and one for each running application server or clone.

On Windows also look at the Services window and check if the WebSphere administrative server is started.

WebSphere administrative repository does not exist

The first time you start the administrative server process, the administrative server will attempt to create the default configuration in the WAS40 or ORCL databases as per the following properties in the admin.config file:

```
com.ibm.ejs.sm.adminServer.createTables=true
install.initial.config=true
```

The first property creates the administrative server database tables and the second one populates the tables with the default configuration.

If the database specified for WebSphere Application Server to use as its repository has not been created, you will see a Specific error 10 message.

If you are using DB2, check that the WAS40 database exists as follows:

1. Start the DB2 Control Center by selecting **Start** -> **Programs** -> **IBM DB2** -> **Control Center** on Windows.
2. In the DB2 Control Center, expand the tree under Systems. Your DB2 databases are listed under Databases. Examine the list to see if WAS40 is there.

   Alternatively, use the following command:

   ```
   $ db2 list db directory
   ```

If the DB2 database WAS40 has not been created, do the following:

1. Start the DB2 Command Line Processor (CLP):
   - In Windows, select **Start** -> **Programs** -> **IBM DB2** -> **Command Line Processor**.
   - In UNIX, enter the command:

     ```
     $ db2
     ```
2. Enter the command:

   ```
   db2 => CREATE DATABASE WAS40
   ```
3. Wait a minute to allow time for DB2 to create the database.
4. Enter the command:

   ```
   db2 => UPDATE DB CFG FOR WAS40 USING APPLHEAPSZ 256.
   ```
5. Type quit to leave the CLP, and then exit to finish the command prompt.
6. Restart the machine, making sure that the following parameters are still set in the admin.config file:

   ```
   com.ibm.ejs.sm.adminServer.createTables=true
   install.initial.config=true
   ```

Note: If you are not using DB2, replace the basic steps given here with the equivalent steps for your DBMS.

Verifying database access is also discussed in Part 3, “Installing WebSphere” on page 125.

Connection to database fails

In addition to making sure that the database exists, you need to ensure that DB2 is running and that the connection to the WAS40 database is successful.

To test the DB2 connection, from a DB2 command window, type:

```
DB2 connect to was40
```

If you cannot connect to DB2, verify the following:

1. Ensure that the right level of code is installed on the WebSphere Application Server machine.
2. Ensure that the database manager has been started.
3. For a remote repository, ensure that the DB2 client is configured properly to point to the DB2 server for the WAS40 database.
4. Configure TCP/IP connectivity to the database server.

5. Check that the user ID and password specified at installation for access to WebSphere Application Server's database are correct. The values for these fields can be found in the admin.config file. The password is encoded, but can be temporarily over-typed with a new value. (It will be overwritten with an encoded value when WebSphere Application Server starts successfully.)

   ```
   com.ibm.ejs.sm.adminServer.dbuser=db2admin
   com.ibm.ejs.sm.adminServer.dbpassword={xor}Oz1tPjsyNjE=
   ```

   Check also that the dbuser has the proper authority to access the database.

   On Windows for example, check that the dbuser is part of the administrator's group and has the following user rights: act as part of the operating system, can create tokens, increase quotas, replace a process level token and log on as a service. Follow the database configuration steps in the install guide to ensure proper authority.

Database configuration and verification are also discussed in Part 3, "Installing WebSphere" on page 125.

Problems loading the database driver

Check for an error message in <WAS_HOME>/logs/tracefile similar to:

```
[02.02.22 09:34:52:805 EST] 15b92ab2 AdminServer   F ADMS0007E: Could not
load database driver COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource
```

This error occurs when you are not using the correct JDBC driver level. For DB2 on Windows see "Update the JDBC level" on page 212, or on AIX see "Update the JDBC level" on page 295.

User does not have the authority to start administrative server

It might be possible that the user ID (specified at installation time) under which WebSphere Application Server is being started lacks sufficient authority:

- In UNIX, log on as root to start the administrative server.
- In Windows, verify that the following conditions are true:
  a. The user is part of the administrator's group.
  b. The administrative server is registered as a Windows service. To manually add the administrative server as a service, at a command prompt, enter:

     ```
     <WAS_HOME>\bin\adminservice.exe install <WAS_HOME>\bin\admin.config
     <HostName>\<User> <Password>
     ```

  c. The user ID has proper rights to start the administrative server:
     - Log on as a service
     - Act as part of the operating system

On Windows NT, select **Start -> Programs -> Administrative Tools -> User Manager -> Policies -> User Rights**.

On Windows 2000, select **Start -> Programs -> Administrative Tools -> Local Security Policy -> Local Policies -> User Rights Assignment**.

d. If the machine is on a Windows NT domain, the user ID should be defined (as an administrative ID) both on the local system and on the domain.

Follow the configuration steps in Part 3, “Installing WebSphere” on page 125 to ensure proper authority.

Logs

Check the following logs for more information on why the administrative server does not start:

1. Check the entries in the log files in the <WAS_HOME>/logs directory:
   - tracefile
   - adminserver_stderr.log
   - activity.log
   - nanny.trace (UNIX)

   See 24.5.4, “Administrative server logs” on page 960 for further details on these log files.

   If the loganalyzer format is in use (see 24.6, “Traces” on page 965) run the logs through the Log Analyzer to obtain more information regarding the errors.

   Errors such as `DB2 service not started`, `incorrect JDBC URL (DB2)`, `Incorrect UID (DB2)` or `incorrect PWD (DB2)` will be easily identified in the above files.
2. Trace the administrative server from startup if necessary, as described in 24.6.4, “Tracing administrative server startup” on page 974.
3. Try to diagnose the problem by launching the administrative server from the command line by executing **`adminservice.exe`** (Windows) or **`startupServer.sh`** (UNIX) and look for errors displayed in the command window.

24.12.3 Problems starting the administrative console

If you attempt to start the administrative console, and the interface does not appear, or it appears briefly and then shuts down, verify the following:

Local administrative console

If you are running the console locally, on the same machine as the administrative server, verify the following:

1. Verify that the WebSphere Application Server's administrative server has successfully started.
2. If using the default port to connect to the administrative server, check that the name service port has not been changed from the default of 900. The administrative server might not be listening for client requests on the default port.
3. Check for memory problems.
4. On UNIX systems verify that adminclient.sh is executable
5. Search the activity.log and tracefile for error messages.
6. If necessary trace the administrative console's startup process as explained in 24.6.5, "Tracing the administrative console" on page 975.

Remote administrative console

If you are running the console remotely, on a different machine from the administrative server, verify the following:

1. Follow the debug steps for "Local administrative console" on page 1017.
2. Check that there is TCP/IP connectivity between the machine that is running the console and the machine that is running WebSphere Application Server. Can you access the server machine's IP address or host name?

   ```
   ping <hostname>
   ```

3. If there is a firewall between the two machines, check that the WebSphere Application Server name service port (default 900) is opened in the firewall:

   ```
   telnet <hostname> 900
   ```

   The Network Address Translation (NAT) function in firewalls cannot be used with a remote administrative client.

4. If you are accessing the console GUI indirectly through a display export utility such as Hummingbird Exceed, install the administrative client locally (you can install just this component of WebSphere Application Server) and launch it against the server by executing the command:

   ```
   adminclient <remote_hostname>
   ```

   You cannot run an administrative console remotely through the X Windows client using an authorized, non-root account with global security enabled. The error message `FATAL Could not bind to the administrative server on {0}{1}` appears when the adminclient.sh or .bat file is executed.

5. Check that the version of the WebSphere Application Server is the same as the version of the administrative console. Different versions, revisions and different fixpak levels between the two can cause failure. Typically this results in a low-level Java exception such as ClassCastException or NullPointerException in the window from which the client is launched.

   To determine the version of WebSphere Application Server, browse the file <WAS_HOME>/properties/com/ibm/websphere/product.xml and look for the version element, for example:

   ```
   <version>4.0.1</version>
   ```

Cannot access administrative server after enabling security

If WebSphere Application Server security is enabled, did a login prompt come up when the client was launched?

If not, it could be that security was enabled since the client was installed and the client does not know that the server is now secured. This can happen, for example, if the client is running remotely.

Check the file <WAS_HOME>/properties/sas.client.props on the client machine if you are running the client from a remote machine. Make sure that the loginSource is set to prompt:

```
com.ibm.CORBA.loginSource=prompt
```

Also, ensure that securityEnabled is set to true:

```
com.ibm.CORBA.securityEnabled=true
```

24.12.4 An application server will not start

If the application server fails to start check the following:

1. View any errors or exceptions thrown in the administrative console.
2. View the <WAS_HOME>/logs/tracefile.
3. Check the application server's standard output and standard error files. Make sure that the path given for the standard output and standard error files while configuring the application server do actually exist.
4. View the activity.log file.
5. If necessary, trace the application server as explained in 24.6.3, "Tracing application server startup with the console" on page 972.

There are two main reasons why an application server would fail to start:

1. A subcomponent of the application server, such as an enterprise application or an enterprise bean, cannot load or start. This is indicated by one or more of the following symptoms:
   - Windows that pop up while the administrative server is attempting to start, with messages such as:

     ```
     Command <server name>.start Sub-command error.
     ```

   - Error messages in the bottom pane of the administrative console (signified by a red triangle) with a text such as:

     ```
     Command <server name>.start Sub-command error,
     ```

   - Messages in the tracefile such as:

     ```
     ADSM0104W: Failed to initialize a server: <server_name>
     ```

   These messages are followed by Java exceptions that point to specific components.
2. The application server is misconfigured in a way that will prevent it from starting. For example:
   - An invalid Java option has been specified as a command-line argument. The offending argument will appear in the application server's standard error file as **"**`Unrecognized option`**"**.

     View the arguments to the Java command from the console, by clicking **Application Server Properties -> JVM Settings -> Generated Command Line Arguments**.
   - There is an invalid trace specification argument. From the application server's perspective this is also an invalid command-line argument. See 24.6, "Traces" on page 965 for information on how to enable component tracing and for the correct syntax.

Following are some of the reasons why a subcomponent would fail to load:

- An enterprise application has been installed, but the actual EAR file is missing.

  The problem is identified by an error in the tracefile or the administrative console such as:

  ```
  com.ibm.ejs.sm.exception.OpException: file:
  <WAS_HOME>/installedApps/sampleApp.ear does not exist.
  ```

  Refer to Chapter 19, "Deploying an application" on page 687 or to the InfoCenter for documentation on how to install enterprise applications.

- The database upon which a persistent (CMP or BMP) enterprise bean depends does not exist, or the application server does not have sufficient authority to access it.

  The following kind of errors will be logged in the application server's standard output and standard error files:

  ```
  Error creating CMP persister using data source <data_source_name>
  Error starting CMP bean <bean_name>
  ```

  When this happens verify the data source name, user ID, and password for accessing the data source. (From the console, under the application server's Installed EJB Modules folder, click the bean's **General** and **CMPBean Datasource** tabs.)

24.12.5 Problems accessing the application

The following sections outline some of the things to check if you are having problems accessing your application from a browser, whether it is static HTML, servlets, JSPs or EJBs. Please refer to the Troubleshooting section of the InfoCenter for more problem scenarios, specially if you are having problems accessing the application when security is enabled. In the following sections we will assume that security is not enabled.

Cannot access any servlet, JSP, or HTML file from a browser

If you are unable to access any Web resources from a browser, examine the following files for clues:

- The Web server error log.
- The plug-in log file (name and location specified in <WAS_HOME>/config/plugin-cfg.xml, as seen in 24.5.7, "Other logs" on page 964). The default is <WAS_HOME>/logs/native.log.
- The plug-in configuration file, <WAS_HOME>/config/plugin-cfg.xml.
- Turn on tracing for the plug-in if necessary, as described in 24.5.7, "Other logs" on page 964.
- Try accessing your resources using the WebSphere Application Server's built-in HTTP server, bypassing the production Web server. If successful, the problem relates to the Web server or to its WebSphere plug-in.

  To access a resource through an application server's HTTP server, specify the application server's HTTP listener port in the URI, for example:

  ```
  http://localhost:9080/servlet/snoop
  ```

  where 9080 is the port number used by the application server under which the enterprise application that contains snoop is installed.

Find the port number through the administrative console by clicking **Application Server Properties -> Services tab -> Web Container Service -> Edit Properties -> Transport**. You can also check the port number in the plugin-cfg.xml file, as shown in Example 24-7.

Example 24-7 Application server's port number from plugin-cfg.xml

```
<ServerGroup Name="itsohost/Default Server">
   <Server CloneID="tOsg92dm" Name="Default Server">
      <Transport Hostname="itsohost" Port="9080" Protocol="http"/>
   </Server>
</ServerGroup>
```

The following could all be causes of the problem:

1. There is a problem with the Web server used to serve WebSphere Application Server-bound requests.

   Check the following:

   a. The physical Web server is available. Can you ping the Web server machine from the machine on which the browser is running to ensure that it is available?

   b. If the Web server and the WebSphere Application Server are running on separate machines, can they communicate with each other?

   c. Is the Web server running?

      On Windows, click **Control Panel -> Services**, to see if the service has started. On UNIX, look for the Web serving processes by issuing the command **`ps -ef | grep http`**.

   d. Can you access the welcome page of the HTTP server, "`http://http_server_name`"?

2. The plug-in has not been properly installed or configured, causing the Web server to communicate improperly with WebSphere Application Server. This is likely if the browser displays a message such as:

   "`The page cannot be found`**"**

   or:

   "`The requested URL/ <app/servlet or JSP file name> was not found on this server`**"**

   Check the following:

   a. Plug-in configuration settings in plugin-cfg.xml that manage Web server/ application server communications, such as virtual host names and URI group names.

      Restart the Web server after editing this file.

b. The plug-in lines that get added to the Web server's configuration file as a result of the WebSphere installation process.

Note: If you need to re-install the plug-in and the Web server is remote, make sure that after installing the plug-in on the Web server machine, you copy the plugin-cfg.xml file from the WebSphere Application Server machine to the Web server machine, as described in Part 3, "Installing WebSphere" on page 125.

3. The application server in which the resources have been installed is not started. This can be the case if you are seeing messages such as this in the browser:

   "Internal Server Error: server encountered an internal error or misconfiguration and was unable to complete your request"

4. The enterprise application or Web module cannot be loaded by WebSphere Application Server. This is indicated by an error such as this in the browser:

   "Error: Failed to load target servlet [&like; <servlet name>]"

   and an error in the WebSphere Application Server console such as:

   "Servlet Error (Root + 1) - [<servlet name>]: Failed to load servlet: javax.servlet.Servlet.ServletException:Servlet [<servlet name>]: Cannot find required servlet class - <classname>.class"

5. Verify that the host name used in the browser is registered as an alias on the virtual host. The virtual host must be correctly configured in both your Web server and in the WebSphere Application Server. Check Host Aliases in the administrative console under the Virtual Hosts, on the General tab.

   If a request is made from your browser, but the server name and port number entered do not match the list of known aliases, a 404 error will be returned to your browser.

6. Check the firewall configurations if there are any firewalls between the browser and the application server.

Cannot access a servlet, JSP, or HTML file from browser

If it is only a specific servlet, JSP file or HTML file that cannot be accessed from a browser, look for errors in the following locations:

1. The application server's standard error and standard output files.

   Look for errors indicating that the application server has problems loading the resource.

2. The plug-in log file (<WAS_HOME>/logs/native.log by default). Check whether there are errors associated with the resource.

3. Check for plug-in problems as explained in the previous section.
4. If necessary turn on tracing for the plug-in and/or the application server.
5. Check the Web server's error log and access log files.

   Look for errors and messages indicating that the Web server attempted to forward the request to the WebSphere Application Server and if it received a response.
6. Check the activity.log and tracefile.

Possible causes of the problem are:

1. The class file that contains the servlet code is missing, is not in the right location or it has a name which does not match what the WebSphere Application Server is looking for.

   This is most likely if you get an "Unable to load servlet" type of message in the administrative console the first time the servlet is accessed.
2. A supporting class (that is, an EJB client class, third-party database access class, and so on) is missing or is not on the classpath of the servlet/JSP. This is indicated by **"**`Missing class`**", "**`java.lang.classNotFound`**"** exceptions in the application server's standard error file.
3. The URI being entered for the servlet is not correct. Make sure that it matches the servlet Web path. This is explained in "Verifying the servlet URI, class file, and classpath" on page 1023.
4. The servlet or JSP file is malfunctioning. This might be the case if the browser, tracefile, administrative console, or activity log display the message:

   "`Uncaught service ( ) exception thrown by servlet`**"**

   This might happen also if the class throwing the exception is not a WebSphere Application Server class (has a package name beginning with com.ibm).

For a description of all of WebSphere Application Server's classpaths, the classpath hierarchy, and when classes get reloaded, see 3.7, "Classloaders" on page 46, or the InfoCenter "Setting classpaths" article.

Verifying the servlet URI, class file, and classpath

To verify that the URI that you are entering correctly matches the servlet served by WebSphere Application Server, and that the class file is in the right place, use the following procedure:

1. Locate the required application server in the administrative console tree view.
2. Open the application server's Installed Web Modules folder.

3. Select the required Web module in the console details view. As you can see in Figure 24-32, we have selected the webbankWeb module from the webbankApplication on the WebbankServer01 application server.

   For our example you can see that the Context root property of this Web module is /webbank. This means that the URI for servlets in this Web module start with “/webbank”.

Note: Servlets are contained within Web applications, which are stored as .war files (Web application archive). Web applications are logically contained in enterprise applications, which are stored as .ear files (enterprise application archive).

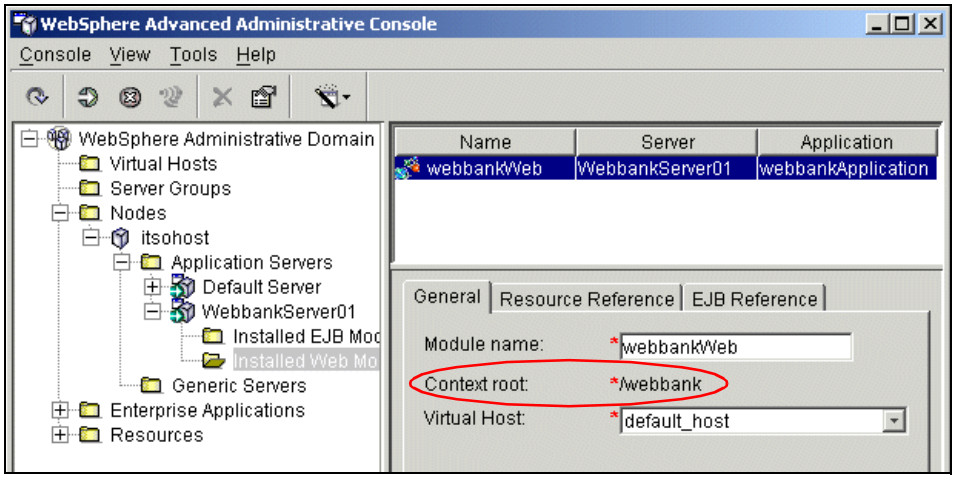


Figure 24-32 Locating the Web module context root

4. Next, open the deployment descriptor for the Web module by right-clicking the Web module and selecting **View Deployment Descriptor** from the pop-up menu. As you can see in Figure 24-33, the webbankWeb module has a servlet named “TransferServlet” mapped to URL pattern “/TransferServlet”.

   If you combine the context root from the previous step with the URL pattern from this step, then the URI used to access TransferServlet will be “/webbank/TransferServlet”. The URL for accessing this servlet is:

```
http://<hostname>//webbank/TransferServlet
```

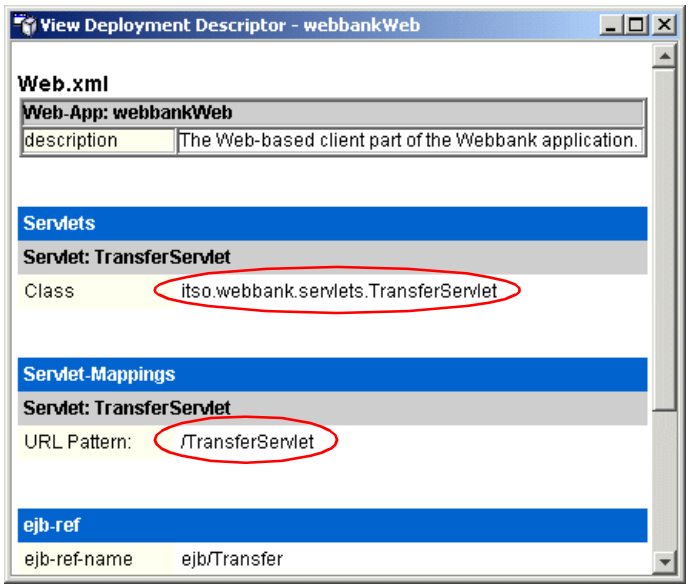


Figure 24-33 Servlet class and URL pattern

In Figure 24-33, we can also see that the class name for this servlet is “itso.webbank.servlets.TransferServlet”. WebSphere Application Server will attempt to find and load this class the first time it is invoked, or optionally when the application server is started.

The class file for this servlet should be included in the deployed enterprise application archive, under the directory:

<WAS_HOME>/installedApps/<application_name>/<web_module_name>/WEB-INF/classes

In our example, the directory is:

D:\WebSphere\AppServer\installedApps\webbankApplication.ear\webbankWeb.war\WEB-INF\classes

Note: Note that the WEB-INF\classes folder is automatically included in the Web application’s classpath. See also 3.7, “Classloaders” on page 46 and 18.14, “Packaging recommendations” on page 683.

5. To check the configuration of classpath dependencies to other modules, view the Web module’s META-INF\MANIFEST.MF file.

   The Class-path setting in our webbankWeb.war\META-INF\MANIFEST.MF file is:

Class-Path: webbankCommon.jar webbankEJBs.jar

The Web module has access to the webbankCommon.jar and webbankEJBs.jar files located in the same application, if allowed by the application server's module visibility setting. See 18.14, "Packaging recommendations" on page 683 for more information.

6. Check also that the virtual host that you are using is correctly configured in both the Web server and WebSphere Application Server. See 14.2, "Virtual hosts" on page 491.

See also 13.2.6, "Finding a URL for a servlet or JSP" on page 473.

EJB access

If you are having problems accessing enterprise bean methods from a client Java application, servlet, or other enterprise bean, here are some things to look at:

- You can use the Application Assembly Tool (AAT) to verify an EJB module. We recommend that you *do not* open an enterprise application from the <WAS_HOME>/installedApps folder. A better option is to export the installed application to a test folder. Use AAT to verify an EJB module as follows:

  a. Locate your application under the Enterprise Applications folder in the console tree view. Right-click your application and select **Export Application...** from the pop-up menu, as shown in Figure 24-34.

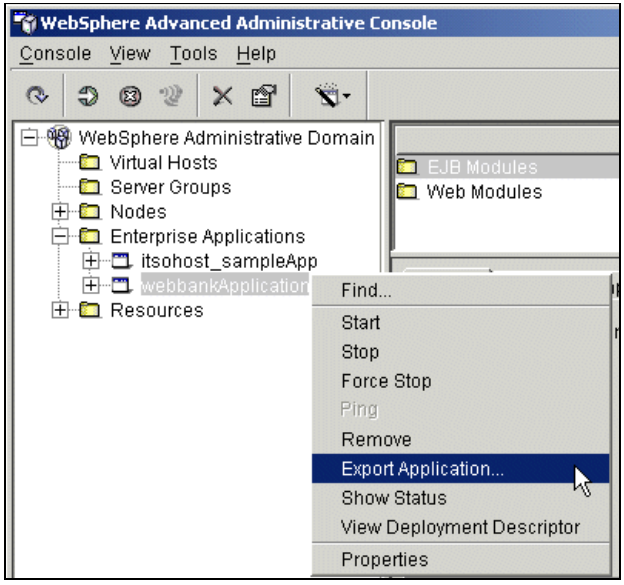


Figure 24-34 Exporting an enterprise application

b. Select an export directory, note the EAR file name, and click **OK**.

c. Open the exported EAR file in AAT. You can start AAT by selecting **Tools -> Application Assembly Tool** from the console main menu.

d. In AAT, right-click the required EJB module and select **Verify...** from the pop-up menu, as shown in Figure 24-35.

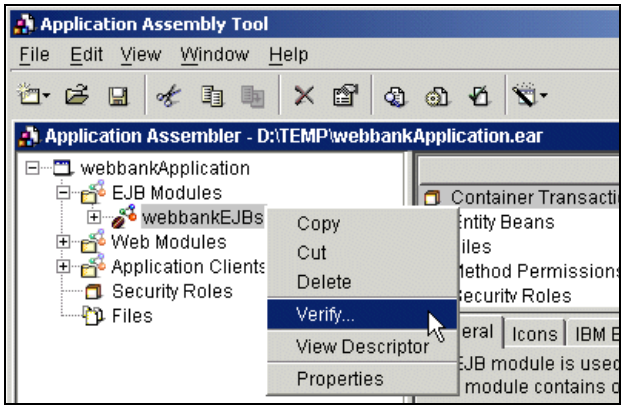


Figure 24-35 Verifying an EJB module in AAT

As shown in Figure 24-36, the verification process checks the following:

- That all the class files referenced in the deployment descriptor exist in the JAR file.
- That method signatures for enterprise bean home, remote, and implementation classes are compliant with the EJB 1.1 specification.

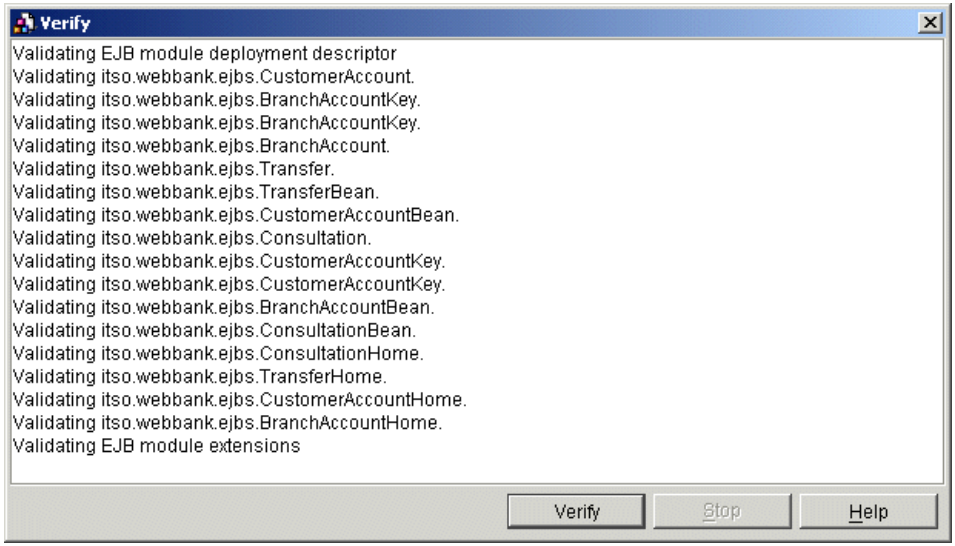


Figure 24-36 Verify module results

e. While in AAT, you can also browse the following items for correctness:

- View the deployment descriptor by right-clicking the EJB module and selecting **View Descriptor** from the pop-up menu.
- View the EJB module properties by right-clicking the EJB module and selecting **Properties** from the pop-up menu.
- View an enterprise bean's properties by right-clicking a bean under the EJB module and selecting **Properties** from the pop-up menu.

- If you do not have access to the AAT, use the command **`jar -xvf`** to extract the files ejb-jar.xml and ibm-ejb-jar-bind.xmi from the EJB module's JAR file. Browse the files in a text or XML editor.

  Verify that the name used by clients to access the bean matches either the original JNDI bean name or one of the bind names.
- If the enterprise bean was originally created for a previous release of WebSphere Application Server, review the "Migrating to supported EJB specification" article in the InfoCenter.
- Ensure that the enterprise bean name is in a package (that is, "com.mycom.MyBean" instead of "MyBean").
- Verify that there is communication between the client and server machines.
- If the enterprise bean is an entity bean, verify access to the database. Use the same user ID and password specified in the data source with which the bean is associated.

- If you have VisualAge for Java, use the VisualAge for Java test client to test the EJBs.
- Use the dumpNameSpace tool to ensure that the EJB is available. See 24.11.1, “DumpNameSpace” on page 1006.
- If there is a CORBA error, look up for an explanation of the CORBA exception’s minor code in the InfoCenter.
- Review the following logs and files:
  - The standard output and standard error files of the application server in which the enterprise bean is deployed.
  - The activity.log.
  - The tracefile.
  - Turn on tracing for the application server if necessary.

24.12.6 Incorrect application behavior

If the application behaves incorrectly or throws exceptions:

1. Check for exceptions in the browser or client.
2. Check the application server standard output and standard error files.
3. If the exception appears to be WebSphere related, turn on tracing for the application server.
4. If the exception appears to be application related or incorrect output occurs, use OLT/OLD or JRas to debug the application, as seen in 24.8, “Tracing user code” on page 980.

25

Migration

This chapter discusses the migration of existing WebSphere V3.02 or V3.5.x Standard or Advanced Edition environments to IBM WebSphere Application Server V4.0, Advanced Edition.

As shown in Figure 25-1 on page 1032 there are two flavors of migration:

- Version migration from WebSphere V3.0.2 and later

  For documentation about migrating from WebSphere V2.0x to WebSphere V3.0x see the information in InfoCenter.

- Edition migration from WebSphere Application Server V4.0, Advanced Edition Single Server (AEs) to WebSphere Application Server V4.0, Advanced Edition (AE)

  Edition upgrade from WebSphere Application Server V4.0, Advanced Edition Single Server to WebSphere Application Server V4.0, Advanced Edition involves the conversion of system configuration from AEs XML-based to AE repository-based configuration. This is mainly handled by the product installation program and it is not covered in this chapter, since documentation can be found in the InfoCenter.

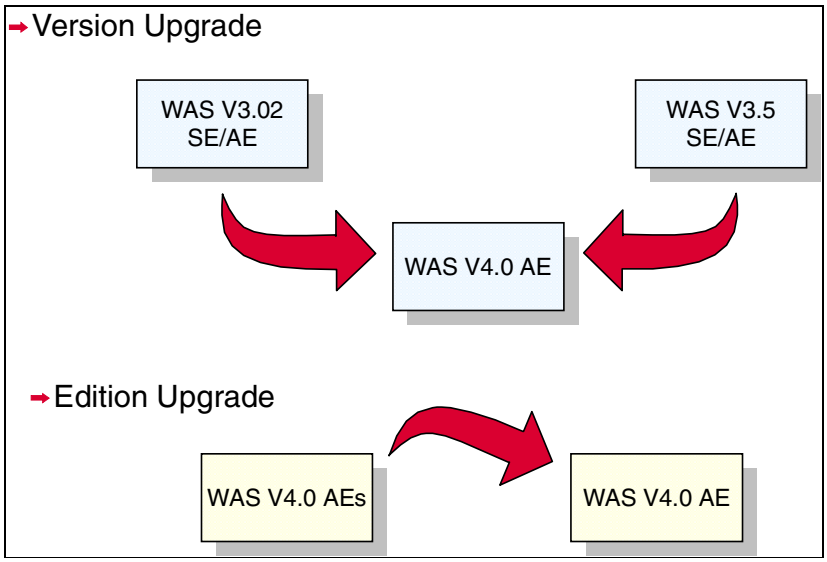


Figure 25-1 Two flavors of migration

The following sections describe the WebSphere Application Server V4.0, Advanced Edition migration wizard and the main steps of the migration process on the Microsoft Windows platform.

25.1 Main steps in WebSphere migration

The main steps in migrating from WebSphere Application Server V3.x to WebSphere Application Server V4.0 are:

- Upgrade product prerequisites to supported versions
- Upgrade to IBM WebSphere Application Server V4.0
- Migrate administrative configurations
- Update application code to supported specification and API levels
- Redeploy applications on WebSphere V4.0

25.2 Migrating product prerequisites

WebSphere Application Server V4.0 prerequisites differ from those at V3.x, and most likely you will need to upgrade corequisites products such as Oracle or DB2 databases, Web servers, JDBC drivers, and so on.

You are strongly urged to check the following link for the most current and accurate information on supported hardware and software components:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

25.2.1 Supported operating systems

The following is a list of operating system levels that are supported:

- Windows NT Server 4.0 (SP 6a)
- Windows 2000 Server (SP 1 or SP 2)
- Solaris 7 (October 2000) or 8 (October 2000)
- AIX 4.3.3 ML4330-07, 4.3.3 ML4330-08, or 5.1
- HP-UX Operating Environment 11.0
- Linux 2.4 Kernel (Red Hat Linux 7.1, SuSE Linux for Intel 7.1)
- OS/400 4.5 or 5.1

25.2.2 Supported databases

The supported levels for databases include:

- DB2 Enterprise Edition 7.2 + FP4
- DB2 Workgroup Edition 7.2 + FP4
- Oracle 8i Enterprise Release 3 (8.1.7)
- Sybase Adaptive Server Enterprise 12.0
- Informix Dynamic Server 9.2.1
- SQL Server 7.0 SP 2
- SQL Server 2000

Note: You must use the JDBC 2.0 driver. This driver is required by the WebSphere V4.0 product administrative server in order to connect to its administrative database. WebSphere V4.0 no longer supports JDBC 1.x. To select the JDBC 2.0 driver for DB2 databases refer to:

- “Update the JDBC level” on page 212 for Windows
- “Update the JDBC level” on page 295 for AIX
- “Update the JDBC level” on page 421 for Linux

To see whether WebSphere V3.x is using JDBC 1.x or 2.0, do the following:

- On Windows:

a. View the DB2 X:\SQLLIB\java12\inuse file.

b. Check that it contains "JDBC 2.0".

- On UNIX:

a. View the WebSphere Application Server <WAS_HOME>/bin/admin.config file.

b. Check the adminServerJvmArgs setting. For JDBC 2.0 it includes the DB2 java12 directory. For JDBC 1.x it includes the DB2 java directory.

Important: On UNIX, perform these steps before installing or starting WebSphere Application Server V4.0.

25.2.3 Supported Web servers

These are the Web server products and levels that are supported:

- IBM HTTP Server 1.3.19
- iPlanet Web Server, Enterprise Edition 4.1 SP7
- Apache Server 1.3.20
- Microsoft Internet Information Server 4.0 (Windows NT) or 5.0 (2000)
- Lotus Domino Enterprise Server (as HTTP Server) 5.0.5, 5.0.6

The migration wizard is designed to check for existing installations and to indicate which components need to be changed, for instance database upgrade, Web server upgrade, and operating system where possible. The JDK is automatically upgraded. The IBM HTTP Server can also be installed from the wizard.

The following software levels were installed in our test environment before migration to IBM WebSphere Application Server V4.0.1, Advanced Edition:

- Windows 2000 Server with Service Pack 1
- IBM WebSphere Application Server V3.5.4, Advanced Edition PTF40118.05
- IBM DB2 Universal Database V7.2.1, Enterprise Edition for Windows

```
D:\> db2level
DB21085I  Instance "DB2" uses DB2 code release "SQL07021" with level
identifier "03020105" and informational tokens "DB2 v7.1.0.43", "n010504"
and "WR21254a"
```

- IBM HTTP Server 1.3.19

25.3 Automated migration to WebSphere V4.0

In IBM WebSphere Application Server V4.0, automated migration support from WebSphere V3.x is part of the installation program. The installation program runs through three phases:

- **Pre-migration**, where the installation program detects previously installed versions and exports the current administrative configuration.
- **Installation**. After you have updated product prerequisites, phase two proceeds with the installation of the new version.
- **Post-migration**. The installation program imports the backed-up administrative configuration.

To migrate your installation from WebSphere V2.0x or from WebSphere Application Server V4.0, Advanced Edition Single Server, read the InfoCenter documentation.

25.3.1 Migration steps

Before starting the installation process of WebSphere V4.0, make sure that the administrative server is running on the existing WebSphere V3.x installation (use Services from the Control Panel in Windows, or `ps -ef` in UNIX).

This is required in order for the migration process to take place automatically.

25.3.2 Pre-migration

The installation program (setup.exe on Windows, install.sh on UNIX) automatically detects previous installed versions of the product and displays them in a list. If the installation program supports migration from a selected version, a Perform Migration check box appears above the list. This is shown in Figure 25-2 on page 1036.

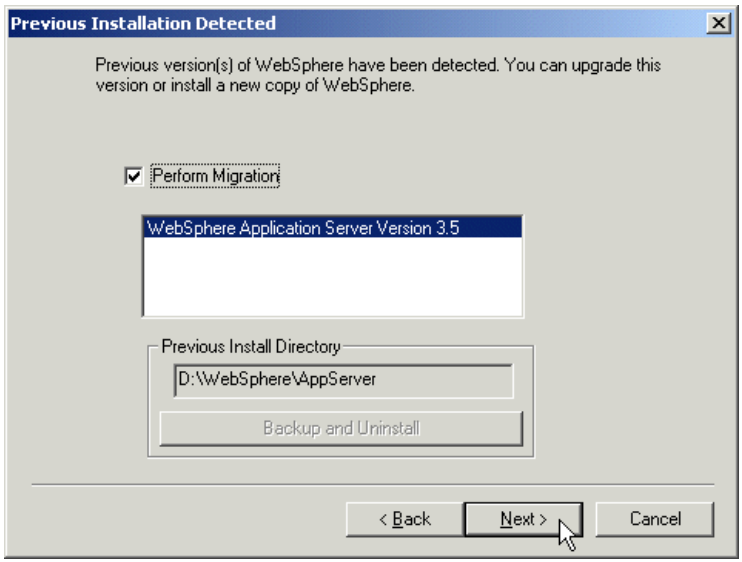


Figure 25-2 Migration window

1. Make sure you check **Perform Migration** to convert the WebSphere V3.x configuration to WebSphere V4.0.

   Clicking the **Backup and Uninstall** option saves a copy of <WAS_HOME>\hosts, \logs, \properties and \properties\servlets into the C:\WebASBackup directory, but configuration in the WebSphere V3.x repository is not saved.

2. As shown in Figure 25-3 on page 1037, the installation program prompts you for the following information:

   – Directory for the new installation (default is the current <WAS_HOME>).

     Note in Figure 25-3 that we specify a directory other than the current <WAS_HOME>: D:\WebSphere40\AppServer.

   – Backup directory (default is wasx_backup).

   – Directory for temporary staging (default is was_migration).

   – Directory for migration log (default is was_migration\logs).

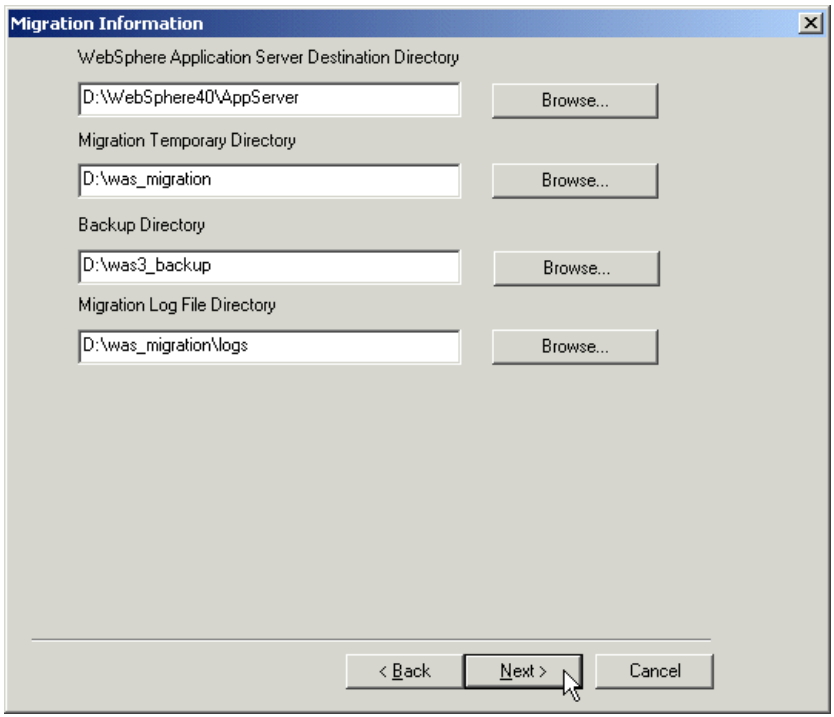


Figure 25-3 Migration Information

3. Click **Next** and the Install window asks you to confirm that pre-migration should start, as shown in Figure 25-4.

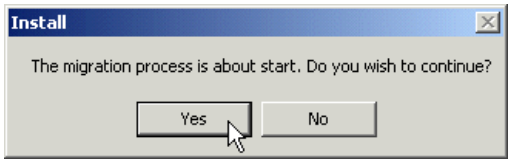


Figure 25-4 Pre-migration start

4. Click **Yes** to start exporting the current administrative configuration.

   Files and directories are saved to the backup directory, d:\was3_backup, and the existing configuration is exported using XMLConfig. Under the covers, the WASPreUpgrade batch file is executed.

   WASPreUpgrade displays the status while it is running, as shown in Figure 25-5 on page 1038. It also saves the logging information into the WASPreUpgrade.log file, which can be viewed with a text editor.

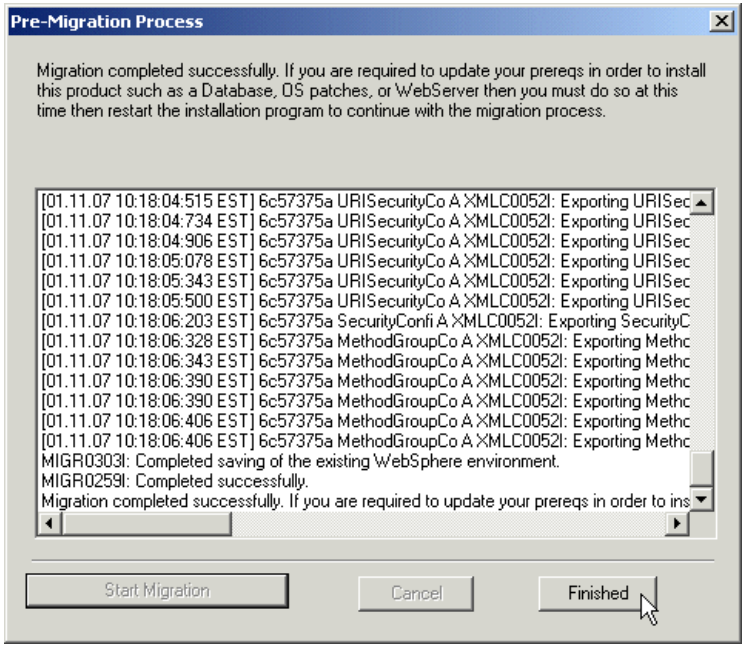


Figure 25-5 Pre-Migration Process

5. Check that the pre-migration process completed successfully by viewing the log file D:\was_migration\logs\waspremigration.log.

   If the pre-migration fails during installation, click **Cancel** and correct the logged errors. Then re-run this phase of migration or run WASPreUpgrade manually from the Migration Temporary Directory.

   For example:

   ```
   WASPreUpgrade.bat D:\was3_backup D:\WebSphere\AppServer itsohost
   -nameServiceHost itsohost -nameServicePort 900 >
   D:\was_migration\premigrate_install.log
   ```

   See 25.5.1, “WASPreUpgrade” on page 1045 for more details.

Note for UNIX users: If the migration log file indicates problems with migration and you click **Finish**, you cannot rerun this phase of the migration until you delete the file /tmp/WAS_Migration_temp.properties.

25.3.3 Migrate prerequisites

Once the pre-migration has completed, migrate prerequisites from your current WebSphere installation to supported levels as described in 25.2, “Migrating product prerequisites” on page 1032.

For our Windows NT, WebSphere V3.5 test environment all prerequisites were at the required levels, so none had to be migrated at this stage.

25.3.4 Installation

To proceed with the installation:

1. Stop the WebSphere V3.x administrative and application servers.
2. Restart the product installation program (setup.exe on Windows, install.sh on UNIX) to continue with the migration.

   The installation program will detect the following problems:

   – Prerequisites that have not been upgraded appropriately (runs prereq.properties from the installation media).

   – A running application server, if the new installation directory is the same as the current one. The product must be stopped in order for the installation program to overlay the files.

   It will then take you through the standard installation process, as shown in Figure 25-6 on page 1040.

Figure 25-6 Installation start

3. We specified the following installation options, as shown in Figure 25-7:

- The destination directory specified during the pre-migration phase. Unlike the standard installation process, it can't be modified during the combined installation and migration.
- All components except the IBM HTTP Server, since we use the existing IBM HTTP Server installation.
- Administrative database name of was40.
- The IBM HTTP Server plug-in for the existing IBM HTTP Server.

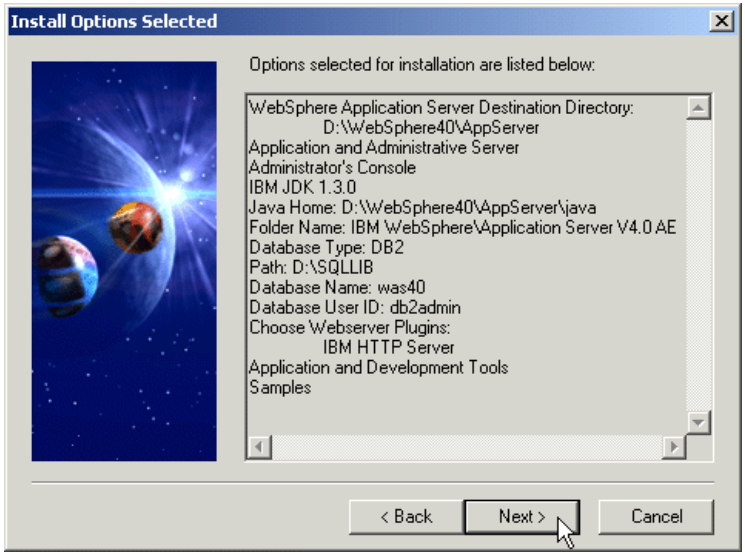


Figure 25-7 Install Options Selected window

25.3.5 Post-migration

After installation, you are prompted to begin post-installation migration, as shown in Figure 25-8 on page 1042.

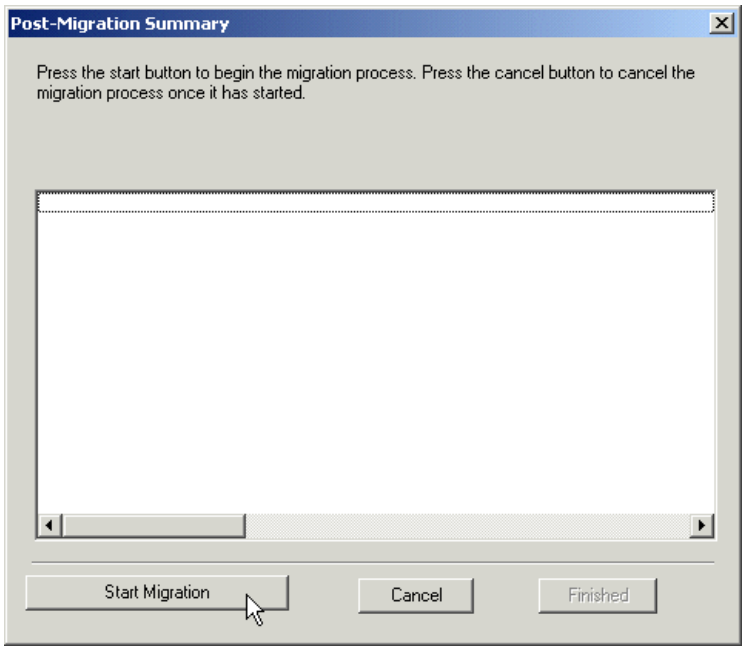


Figure 25-8 Post-migration summary start

1. Click **Start Migration.**

   The installation program takes the previously saved configuration file d:\was3_backup\websphere_3x_backup.xml and converts it into a valid WebSphere V4.0 XML configuration file d:\was3_backup\websphere_4x_restore.xml. This file is then used by XMLConfig to restore the previous WebSphere environment into the repository.

   Note: With one minor modification, you can use the websphere_3x_backup.xml file with a WebSphere V3.x XMLConfig tool to restore the previous configuration. The version of XMLConfig used by WASPreUpgrade encodes passwords during export. Before you import the configuration back into the WebSphere V3.x installation, these passwords must be reset to their correct, unencoded values.

Important: The websphere_3x_backup.xml file, found in the backup directory, cannot be used directly in the WebSphere V4.0 environment. The XML data files exported by the WebSphere V3.x XMLConfig tool cannot be processed by the WebSphere V4.0 tool.

2. When the post-migration phase finishes, the installation program displays the migration log file. Check <WAS_HOME>\logs\WASPostUpgrade.log file to see what is imported.

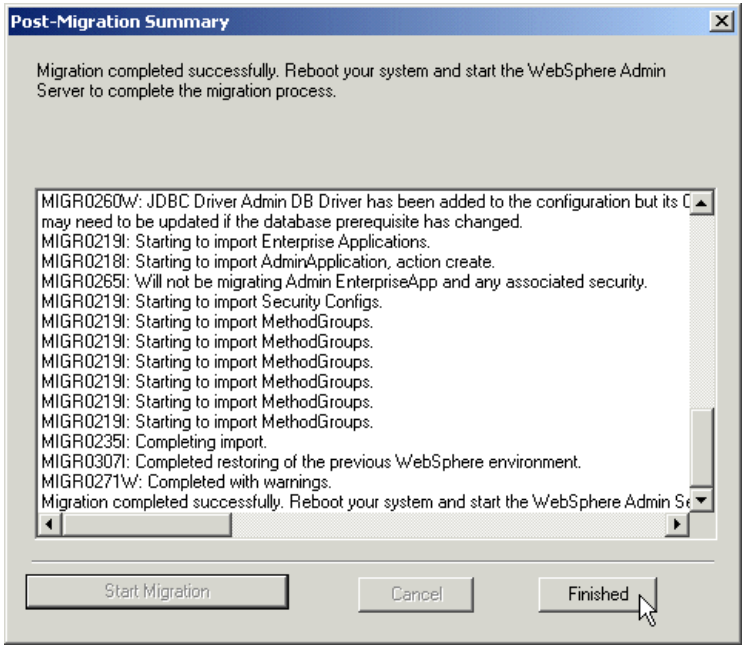


Figure 25-9 Post-migration summary finished

3. If this phase finishes successfully, reboot your system and start the WebSphere administrative server.

The post-migration phase runs WASPostUpgrade.bat under the covers and as such it could be run manually to restore the configuration in the new installation (from <WAS_HOME>\bin).

25.4 Manual migration of administrative configurations

In the previous section we have seen how the automated migration, which is part of the product installation, automatically handles the migration of administrative configurations by calling WASPreUpgrade and WASPostUpgrade under the covers. These migration tools can also be called from the command line.

The following is a summary of what the automated product migration process does:

1. It backs up the current administrative configuration and user data files.

2. It installs the new version of WebSphere Application Server.
3. It restores the administrative configuration in the new installation.

The next sections outline the steps required to complete the migration manually. We are not aware of any scenarios where manual migration might be necessary, but this feature is available for people who want to do things themselves.

25.4.1 Backing up the current configuration

You can save the current configuration by using either of the following:

1. Using the WebSphere V4.0 migration tools (WASPreUpgrade).

   This technique exports the existing repository and saves the necessary files. This method is recommended, because logic is provided to save the correct files in the structure required by the other migration tools.

2. Using the WebSphere V3.x XMLConfig tool.

   This export technique works with the following limitations:

   – The passwords are not encoded. Data import is still handled correctly by the migration tools later, but you should be aware of the security exposure of having passwords in the file unencoded.

   – Security data may not be saved (WebSphere versions prior to 3.02.2).

   – Substitution keys may be used.This can be resolved by using the `-substitute` keyword when using the WASPostUpgrade tool later.

   See 23.4, “Introducing XMLConfig” on page 923 or the InfoCenter for more information on XMLConfig.

Manually back up the following files and directories:

- <WAS_HOME>\hosts
- <WAS_HOME>\servlets
- <WAS_HOME>\classes
- <WAS_HOME>\deployableEJBs
- <WAS_HOME>\deployedEJBs
- <WAS_HOME>\properties
- admin.config

25.4.2 Restoring the administrative configuration

To install and restore the administrative configuration:

1. Uninstall WebSphere Application Server V3.x.
2. Install WebSphere Application Server V4.0.
3. Restore the previous configuration to the new installation by using WebSphere V4.0 migration tools (WASPostUpgrade).

   This tool uses the information created by the WASPreUpgrade tool to restore the previous WebSphere V3.x configuration to a WebSphere V4.0 installation.

Important: The websphere_3x_backup.xml file, found in the backup directory, cannot be used directly in the WebSphere V4.0 environment. The XML data files exported by the WebSphere V3.x XMLConfig tool cannot be processed by the WebSphere V4.0 tool:

- You cannot run XMLConfig -import <websphere_3x_backup.xml>.
- You can only run WasPostUpgrade -import <websphere_3x_backup_xml>.

25.5 Migration tools

The following sections describe the syntax of the WASPreUpgrade and WASPostUpgrade tools.

25.5.1 WASPreUpgrade

This tool is used to save the system configuration of an existing WebSphere V3.02 or V3.5 environment.

It saves to a specified backup directory all the files from the following directories in the existing WebSphere V3.x environment:

- <WAS_HOME>\hosts
- <WAS_HOME>\servlets
- <WAS_HOME>\classes
- <WAS_HOME>\deployableEJBs
- <WAS_HOME>\deployedEJBs
- <WAS_HOME>\properties

It also saves selected files from the WebSphere V3.x bin directory (admin.config and setupCmdLine.bat).

Later, the backup directory can be used with the WASPostUpgrade tool to restore the previously saved environment into a WebSphere V4.0 installation.

WASPreUpgrade also exports the existing application server configuration from the repository. If you are migrating from an Advanced Edition Installation, this step requires that the administration server of the existing environment be running.

WASPreUpgrade parameters

The parameters of the command are as follows:

```
WASPreUpgrade <backupDirectoryName> <currentWebSphereDirectory>
<administrationNodeName> [-nameServiceHost <host name> [ -nameServicePort <port
number>]] [-traceString <trace spec> [-traceFile <file name.]]}
```

The first three parameters of the command are required positional parameters. All the others are optional.

- `<backupDirectoryName>`. This is the name of the directory to store the saved configuration and files. The directory will be created if it does not already exist.
- `<currentWebSphereDirectory>`. This is the directory name of the currently installed WebSphere V3.02 or V3.5 product (<WAS_HOME>). This directory can be either a WebSphere Standard or Advanced Edition installation.
- `<administrationNodeName>`. This is the administration node name of the currently installed WebSphere product. XMLConfig will be called using this parameter. The WebSphere administration server will need to be running for this command to execute successfully.
- `[-nameServiceHost <host name> [-nameServicePort <port number>]]`. These parameters can be used to override the default host name and port number used by XMLConfig.
- `[-traceString <trace spec> [-traceFile <file name>]]`. The -traceString parameter value is **"**`*=all=enabled`**"** and must be specified with quotes to be processed correctly.

An example of usage of WASPreUpgrade follows:

```
WASPreUpgrade.bat d:\was3_backup d:\WebSphere\AppServer itsohost
-nameServicePort 900 -nameServiceHost itsohost >
d:\was_migration\premigrate_install.log 2>&1
```

25.5.2 WASPostUpgrade

WASPostUpgrade is used to restore the administrative configuration from a previous WebSphere V3.02 or V3.5 environment. The administrative configuration could have been backed up manually or by using the WASPreUpgrade tool, as seen in 25.4.1, “Backing up the current configuration” on page 1044. The result of this command is an updated WebSphere V4.0 administrative configuration. The administrative server must be running for this command to run successfully.

WASPostUpgrade parameters

The parameters of the command are as follows:

```
WASPostUpgrade <backupDirectoryName> [-import <xml data file>] [-adminNodeName
<primary node name>] [-nameServiceHost <host name> [ -nameServicePort <port
number>]] [-traceString <trace spec> [-traceFile <file name>]] [-substitute
<“key1=value1[;key2=value2;[...]]”>]
```

The first parameter and the node name are required. The others are optional.

- `<backupDirectoryName>`. This is the name of the directory that has the saved configuration and files. It was created by the WASPreUpgrade tool.
- `[-import <xml data file>]`. This is an optional parameter that can be used to specify an XML data file that was created using either a WebSphere V3.02 or V3.5 version of XMLConfig. The default XML configuration file (websphere_3x_backup.xml) in the backupDirectoryName will be used if this parameter is not specified.
- `[-adminNodeName <primary node name>]`. This is the administration node name of the currently installed WebSphere product.
- `[-nameServiceHost <host name> [-nameServicePort <port number>]]`. These parameters can be used to override the default host name and port number used by XMLConfig.
- `[-traceString <trace spec> [-traceFile <file name>]]`. These parameters are used to gather trace information. The -traceString parameter value is `“*=all=enabled”` and must be specified with quotes to be processed correctly.
- `[-substitute “key1=value1[;key2=value2;[...]]”]`. It is used for substitution of security values in the XML data file. In the input XML file, each key should appear as key for substitution.

25.5.3 Transitioning to WebSphere V4.0

After migrating to WebSphere V4.0 and starting the administrative console you will see a different topology view from that in WebSphere V3.x. Refer to 13.1, “Introducing the WebSphere Administrative Console” on page 442.

The J2EE programming model specifies an architecture for how applications are created and deployed. Because applications in WebSphere V3.x were not architected in the same manner, the migration process re-creates these applications. All migrated Web resources and enterprise beans are created in J2EE applications. All enterprise applications defined in the WebSphere V3.x installation are mapped into J2EE applications with the same name and deployed in the default server. All other Web resources and enterprise beans that are mapped but not included in an enterprise application are mapped into a default J2EE application called “DefaultApplication”.

Servlets and Web applications are mapped to J2EE Web Application Archive (WAR) files. Enterprise beans are deployed as EJB 1.1 beans in J2EE JAR files using EJBDeploy. These resources are combined in a J2EE Enterprise Application Archive (EAR) file and deployed in the WebSphere V4.0 configuration. Security is mapped to J2EE security and included with the EAR. Models and clones are converted to server groups and clones and their EJBs and Web applications are included in the application conversion.

Samples are not migrated. They have been updated specifically for J2EE in WebSphere V4.0. The new samples should be used instead of the ones previously provided with WebSphere V3.x.

It is recommended that you carefully analyze the WASPostUpgrade.log file, because detailed information specific to each bean deployed is saved in the log.

Note: The OSE protocol has been removed in WebSphere V4.0, and you should switch to the HTTP transport to allow the Web server communicate with WebSphere Application Server V4.0. HTTP transport is configured as part of the Web Container Service and GenPluginCfg.bat can be used to edit the default plug-in configuration file, plugin-cfg.xml. See Chapter 14, “Configuring the Web server interface” on page 481.

25.6 Migrating your application code

Though migration creates J2EE applications and deploys them, the migration tools do not update application code to supported specification levels. So there will be cases where the developer will need to modify the application to make application code compliant with the J2EE specification. After code changes are completed, the application must be redeployed on WebSphere V4.0.

For more information about J2EE, see Chapter 3, “The Java 2 platform” on page 29, or the following Web site:

http://java.sun.com/j2ee

Detailed examination is beyond the scope of this book,but the following is a summary of the potential migration areas:

- **Enterprise Beans**. The EJB specification level for WebSphere V4.0 has changed from that of WebSphere V3.x. In WebSphere V4.0, the EJB specification level is EJB 1.1, not EJB 1.0, and the prerequisite is JDK 1.3.

  EJBs may not require many coding changes, but will need to be repackaged with J2EE deployment descriptors. If you have an EJB 1.0 jar file, you will be able to convert it to an EJB 1.1 jar file (containing the correct J2EE deployment descriptors) using the Application Assembly Tool (AAT). However, this process will not convert your code. There are some features of the 1.0 programming model that are tolerated in WebSphere V4.0, but may not be supported in future releases of WebSphere Application Server.

  Any code using WebSphere Application Server data sources, including BMP entity beans and session beans that access databases, will need to be modified. For further details see the section below on JDBC and IBM database connection support APIs.

- **Servlets**. Servlets will require migration if they are not at the supported specification level 2.2 or they rely on deprecated or removed IBM servlet extensions. Servlet 2.2 support was introduced in WebSphere V3.5.2. The WebSphere V3.5.2 and higher InfoCenters detail the changes needed to update servlets to 2.2. Refer to the Java Servlet API 2.2 specification for complete information concerning new and deprecated APIs. See:

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

  One of the most significant changes between Servlet 2.1 and 2.2 is the scope of the HTTP session object. In WebSphere V4.0, HTTP sessions cannot be shared between Web applications. This may require coding changes to applications that use shared HTTP sessions. See Chapter 15, “Configuring session management” on page 513 for more information.

- **JSPs**. JSP 0.91 is not supported in WebSphere V4.0. Applications with JavaServer Pages (JSPs) written to the 0.91 specification will need to be

updated to the JSP 1.1 specification level. Since JSP 1.1 is a strict superset of JSP 1.0, JSP 1.0 code should run unmodified, but will not take advantage of any of the improvements in 1.1.

Figure 25-10 shows the migration path for Servlets, JSPs and EJBs:

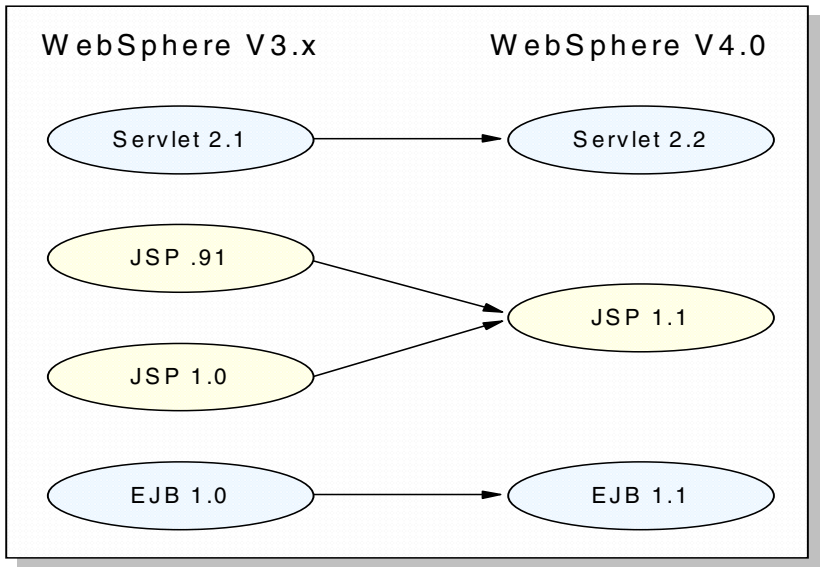


Figure 25-10 Servlet, JSP and EJB migration

- **XML**. XML4J 2.0.15 is replaced by XML4J 3.1. If your XML applications use XML for Java API Version 2.0.x or earlier, you must migrate them to API Version 3.1, or the equivalent open-source version.
- **JDBC and IBM database connection support APIs**. Connection pooling (provided through data source objects) was introduced in WebSphere V3.0x. Applications that use V3.0x connection pooling need to be changed slightly and recompiled.

  If existing applications are still using the connection manager model from WebSphere V2.0x, you must update the application code to use the current connection pooling model. The shift in models corresponds to a change in supported JDBC specification levels. Do not forget to switch to supported JDBC 2.0 drivers.
- **Items newly deprecated in WebSphere V4.0**

  `ConnectionPreemptedException`, introduced in WebSphere V3.0x, no longer exists. `StaleConnectionException` has replaced `ConnectionPreemptedException` in all cases.

The packages `com.ibm.db2.jdbc.app.stdext.javax.sql` and `com.ibm.ejs.dbm.jdbcext` have been deprecated.

Replace the following import statement:

```
import com.ibm.db2.jdbc.app.stdext.java.sql.*;
```

with the statement:

```
import javax.sql.*;
```

Applications using the `com.ibm.ejs.dbm.jdbcext` package will still be allowed to retrieve a data source, but new data sources cannot be created or bound into JNDI by using this interface. All new data sources must be created by using `com.ibm.websphere.advanced.cm.factory.DataSourceFactory`.

The following methods in `com.ibm.websphere.advanced.cm.factory.DataSourceFactory` have been deprecated: `createJTADataSource()` and `createJDBCDataSource()`. These methods have been replaced with the `getDataSource(java.util.Properties)` method.

The class `com.ibm.ejs.cm.portability.StaleConnectionException` has been deprecated. Applications currently using this class will still function, but it is recommended that new applications be written using `com.ibm.websphere.ce.cm.StaleConnectionException`.

- **User profiles**. WebSphere V3.0 and V3.5 provided a User Profile Manager XML object that allowed you to store and retrieve information about each user. In WebSphere V4.0, the User Profile Manager is packaged separately as an EJB. UserProfileManager configuration is not migrated.
- **HttpSession**. Certain access methods have been deprecated in WebSphere V4.0. These deprecated APIs still work in WebSphere V4.0, but they may be removed in a future version of the product. Changes are summarized in Table 25-1:

Table 25-1 Access methods deprecated

Wherever you use	Use instead
getValue()	getAttribute()
getValueName()	getAttributeNames()
removeValue()	removeAttribute()
setValue()	setAttribute()

In accordance with the Servlet 2.2 specification, HttpSession objects must be scoped within a single Web application context; they may not be shared between contexts. This means that a session can no longer span Web applications. Objects added to a session by a servlet or JSP in one Web application cannot be accessed from another Web application. The same session ID may be shared (because the same cookie is in use) but each Web application will have a unique session associated with the session ID.

- **Session configuration**

  Relative to session security, the default Session Manager setting for Integrate Security is now false. This is different from the default setting in some earlier releases.

  Refer to Chapter 15, “Configuring session management” on page 513, or to the InfoCenter for more information on session configuration.

- **Security**. Unlike previous versions, WebSphere V4.0 does not protect URIs served by an external Web server. One way of protecting these is by using the WebSeal product. WebSphere V4.0 continues to protect URIs (including URIs for HTML files) that are served by the application server.

- **Transactions**. WebSphere V3.x ran with a 1.1.x level of JDK. WebSphere V3.x included packages written by IBM to provide transaction support features usually provided by JDK 1.3. Now that WebSphere V4.0 runs with JDK 1.3, applications should no longer import the proprietary IBM packages, but instead import the standard Java 1.3 packages that provide the required functionality.

  a. In Java source files, find the import statement:

     ```
     import com.ibm.db2.jdbc.app.jta.javax.transaction.*
     ```

  b. Change the import statement to:

     ```
     import javax.transaction.*
     ```

  c. Recompile the Java files using JDK 1.3.

  Other transaction considerations for WebSphere V4.0:

  – One database connection cannot be used across multiple user transactions. If an application component obtains a connection to a database, then begins a transaction, the connection is closed automatically when the transaction ends. The connection must be obtained again before beginning another transaction.

  – Transactions that began by using UserTransaction now use the isolation level specified when the enterprise bean is deployed.

    In WebSphere V3.02, the transaction isolation level defaulted to:

    • REPEATABLE_READ for DB2

    - SERIALIZABLE for Oracle
  - The timeout units for transaction inactivity are in milliseconds.
  - If multiple datasource connections are involved in the same transactions, then JTA must be enabled on those datasources. JTA must be enabled for two-phase commit actions.

- **XML configuration**. The XML Configuration Management Tool (XMLConfig) was introduced in WebSphere V3.02. Many of the interfaces have changed in WebSphere V4.0. Refer to 23.4, “Introducing XMLConfig” on page 923 for further details.

  Changes from WebSphere V3.0x for programmatic access:

  - The XMLConfig constructor now throws NamingException and InvalidArgumentException.
  - The XMLConfig tool now supports variable substitution and variable Hashtable setter.

- **WebSphere Control Program.** The WebSphere Control Program was introduced in WebSphere V3.5. Many of the interfaces have changed in WebSphere V4.0. Refer to 23.1, “Introducing WebSphere Control Program” on page 882 for further details.

25.6.1 Features removed from WebSphere V4.0

- **wlmjar utility**. The wlmjar utility is not needed. WebSphere V4.0 supports workload management in a more transparent way than V3.x. The wlmjar utility and the associated Java class are deprecated. There is no need to use these, since all EJBs in WebSphere V4.0 can take advantage of the WLM support that is built into the ORB.
- **OSE remote**. Support for the OSE protocol has been removed in WebSphere V4.0. You should switch to the HTTP plug-in to allow the Web server communicate with WebSphere Application Server V4.0. HTTP transport is configured as part of the Web Container Service and GenPluginCfg.bat can be used to edit the defaultplug-inn configuration file, plugin-cfg.xml. See Chapter 14, “Configuring the Web server interface” on page 481 for more information.
- **Servlet redirector**. This feature has been removed now that the Web server plug-in can connect to the application servers via HTTP/HTTPS.
- **Native install**. Native install is no longer an option. This option used to be available on WebSphere V3.5.x and allowed using tools such as SMIT (AIX) to install the product. However, the Java installer in silent mode (that is, with a response file) can be used if you need to install without using a GUI, such as X-Windows on UNIX systems.

- **AE browser-based administration**. This was provided as tech preview in WebSphere V3.5.x.

25.7 Redeploy applications

There are two steps involved in creating J2EE applications:

1. Copying the appropriate files into the archive (classes, JSPs, HTML, image files, and so on).
2. Creating deployment descriptor files for the modules and applications.

In WebSphere V4.0, the AAT supports both steps by enabling users to copy files with the appropriate relative paths into the archive, as well as by providing a GUI method for defining deployment descriptors.

Developers can also set environment-specific binding information through the AAT. These bindings are used as defaults when the application is installed through the administrative console. For more information on packaging and deploying applications see Chapter 18, “Packaging an application” on page 639 and Chapter 19, “Deploying an application” on page 687.

25.8 Staging suggestions

In order to ease the effort of migrating, the following staging suggestions are recommended before migrating to WebSphere V4.0:

1. Use WebSphere V3.5.2, minimum, for development and test.
2. Move to JSP 1.1 and Servlet 2.2.

   JSP 1.1 and Servlet 2.2 specifications are supported in WebSphere V3.5.2 and later, and it would be a good idea to convert your applications to use these features on your existing installation.
3. Eliminate usage of removed APIs.

   The use of removed APIs such as the Connection Manager API (com.ibm.servlet.connmgr.*) should be eliminated.

Part 6

Appendixes

A

Back up and restore your WebSphere environment

In this appendix we describe the steps you can use to:

- Back up your WebSphere environment
- Restore your WebSphere environment from a previously created backup

Back up your WebSphere environment

You can use the following steps to back up your WebSphere environment completely with the entire configuration and all files. The steps contain commands for Windows platforms. For UNIX the commands are similar.

1. Create directories for all backup information:

```
mkdir D:\WASbackup
mkdir D:\WASbackup\bin
mkdir D:\WASbackup\properties
mkdir D:\WASbackup\installedApps
mkdir D:\WASbackup\installableApps
mkdir D:\WASbackup\etc
```

2. Back up WebSphere configuration as XML.

   Make sure the administrative server is running. Make a backup of your current WebSphere configuration use XMLConfig. The configuration will be saved in a XML file. Use the following command:

```
XMLConfig -adminNodeName itsohost -export D:\WASbackup\wasexport.xml
```

   Note: Using XMLConfig to restore the configuration redeploys all the enterprise applications. That means all changes directly made to the installed enterprise application are lost. To avoid this, export the enterprise application to an installable EAR file. Replace the original EAR file with the exported version.

3. Stop the administrative server. Use the WebSphere Administrative Console and stop the node, or use WSCP:

```
wscp> Node stop /Node:itsohost/
```

4. Copy configuration, properties and key files:

```
copy D:\WebSphere\AppServer\bin\admin.config D:\WASbackup\bin
copy D:\WebSphere\AppServer\properties\*.prop* D:\WASbackup\properties
xcopy D:\WebSphere\AppServer\etc D:\WASbackup\etc
```

5. Back up WebSphere administrative repository.

   The WebSphere administrative repository contains only the WebSphere configuration data and by default no application data. For DB2, use the following steps to back up the `WAS` database:

   a. Log in with DB2 SYSADM privileges and start a DB2 command window:

```
db2cmd
```

   b. Make sure all applications are disconnected from the database:

```
db2 list active databases
```

If this command returns a list of active databases including `WAS`, then first disconnect all applications.

c. Make the backup of your `WAS` database.

```
db2 backup database WAS to D:\WASbackup
```

Note: Please make sure you also make a backup of database(s) containing your application data.

d. For example, if `SAMPLE` is the application database, then use the following command:

```
db2 backup database SAMPLE to D:\WASbackup
```

6. Make copies of your application code (EAR, JAR, WAR, etc.). Make sure you copy the installed and the installable versions, too. Follow the steps to make copies of your application code located in the default directories:

```
xcopy /e D:\WebSphere\AppServer\installedApps D:\WASbackup\installedApps
xcopy /e D:\WebSphere\AppServer\installableApps
D:\WASbackup\installableApps
```

If you have installed connectors, then copy these, too:

```
xcopy /e D:\WebSphere\AppServer\installedConnectors
D:\WASbackup\installedConnectors
```

Restore your WebSphere environment

To restore the previously saved WebSphere configuration in a freshly installed WebSphere Application Server V4.0 environment, first make sure the following preconditions are met:

- Node name of the host must be the same.
- Version number of WebSphere including fixpak levels must be the same.
- The <WAS_HOME> directory (for example, D:\WebSphere\AppServer on Windows) must be the same.
- Database names need to be the same for the administrative repository and application data.

Use the following steps to restore your new WebSphere environment to the state of the previous backup:

1. Copy back all the configuration, properties, and key files.

```
copy D:\WASbackup\bin D:\WebSphere\AppServer\bin\admin.config
copy D:\WASbackup\properties D:\WebSphere\AppServer\properties
xcopy D:\WASbackup\etc D:\WebSphere\AppServer\etc
```

2. Copy back all the installed and installable application code and connectors:

```
xcopy /e D:\WASbackup\installedApps D:\WebSphere\AppServer\installedApps
xcopy /e D:\WASbackup\installableApps
D:\WebSphere\AppServer\installableApps
xcopy /e D:\WASbackup\installedConnectors
D:\WebSphere\AppServer\installedConnectors
```

3. To restore your configuration do one of the following:
   - Restore configuration from XML file:
     i. Edit the <WAS_HOME>\bin\admin.config file. Set the following properties:

```
install.initial.config=false
com.ibm.ejs.sm.adminServer.createTables=true
```

     ii. Make sure the administrative repository database server is running.
     iii. Check that the administrative repository database is created. For DB2:

```
db2 list database directory
```

     If it is not created, use the createdb.bat script in the <WAS_HOME>\bin directory to create it.
     iv. Start the administrative server and it should create the administrative table structure:

```
net start "IBM WS AdminServer 4.0"
```

     v. Use XMLConfig to import the XML file:

```
XMLConfig -adminNodeName itsohost -import D:\WASbackup\wasexport.xml
```

     vi. Stop the administrative server using WSCP:

```
wscp> Node stop /Node:itsohost/
```

   - Restoring the configuration from DB2 backup:
     i. Make sure that DB2 is running and administrative server is stopped.
     ii. Edit the <WAS_HOME>\bin\admin.config file. Set the following properties:

```
install.initial.config=false
com.ibm.ejs.sm.adminServer.createTables=false
```

     iii. Log in with DB2 SYSADM privileges and start a DB2 command window:

```
db2cmd
```

     iv. Restore your DB2 backup of `WAS` database:

```
db2 restore database WAS from D:\WASbackup
```

4. Restore your application data database(s), if necessary.

5. Start the administrative server.

```
net start "IBM WS AdminServer 4.0"
```

B

The admin.config file definitions

This chapter contains the definitions of the admin.config file. The admin.config file contains many administrative server properties you can set. In addition, you might want to pass the administrative server some generic Java command-line arguments.

Directives in the <WAS_HOME>/bin/admin.config file are similar to their counterparts on the Java command line for the administrative server. The command line argument name is appended to a standard package name for the administrative server. For example the command-line argument -lsdPort becomes com.ibm.ejs.sm.adminServer.lsdPort in the admin.config file.

Table B-1 on page 1064, Table B-2 on page 1065, Table B-3 on page 1067, Table B-4 on page 1068, Table B-5 on page 1068, Table B-6 on page 1068, Table B-7 on page 1069, and Table B-8 on page 1069 show the list of parameters that you can specify in the admin.config file. Note that you should only update this file if necessary. Before you update it, we recommend that you make a backup to recover from unexpected results of modification.

com.ibm.CORBA package

Table B-1 Parameters in the admin.config file: com.ibm.CORBA package

Property	Default value	Description
ConfigURL		The path to the file specifying ORB configuration values, such as: file\:/<WAS_HOME>/properties/sas.server.props
ServerSocketQueueDepth	50	The ServerSocket maximum queue depth. Specify an integer greater than 50. Values less than 50 or invalid integer values will cause a warning to be logged to the message file and the default value of 50 to be used. Setting this property to a value beyond operating system-imposed limitations will likely not have any effect.

com.ibm.ejs.sm.adminServer package

Table B-2 Parameters in the admin.config file: com.ibm.ejs.sm.adminServer package

Property	Default value	Description
adminDomain		The logical name specified for the WebSphere administrative domain.
adminDomain.bootstrap		The description is unavailable at this time.
agentMode	false	Whether this administrative server should run as a full administrative server (false), or in administrative server agent mode (true).
bootstrapHost		Fully qualified host name of the machine containing the administrative server.
bootstrapPort	900	Port number for the administrative server. The administrative console connects to this port.
classpath		Location of WebSphere Application Server libraries and files. Never edit this except for WebSphere patches or when updating the classpath to add JDBC driver to the administrative database.
connectionPoolsize	5	The number of concurrent database connections to allow from the administrative server to the administrative database.
connectionProperties		Use this property to pass arguments into setConnectionProperties() for the WebSphere administrative server Java process. Separate multiple connectionProperties with semicolons: name=value;name=value; ...
createTables	Initially true, then set to false the first time you run the administrative server	Set to true to create the administrative database tables the next time you start the administrative server (typically, the first time you start the administrative server after product installation). In WebSphere V3.5, this setting is called dbInitialized.
dbdataSourceClassName		The data source corresponding to the administrative database. For example: COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource
dbdisable2Phase		Set to true to disable two-phase commit on administrative database transactions.

Property	Default value	Description
dbinformixLockModeWait		A data source property required for certain database brands. What was called dbUrl in WebSphere V3.5 is replaced by specific data source properties for each database brand.
dbpassword		The password for the administrative database.
dbSchema	EJSADMIN	The database schema for the administrative database.
dbUrl		A data source property required for certain database brands. What was called dbUrl in WebSphere V3.5 is replaced by specific data source properties for each database brand.
dbuser		The user ID for the administrative database.
diagThreadPort	-1 = next available port	The port on which DrAdmin listens. The default is to use the next available port. This property allows you to control the port and is mainly used in DMZ configurations.
disableAutoServerStart	false	Disable automatic starting of the application server. Setting this to true will start the node without starting any application server(s).
disablePMI	true	When set to true the performance monitoring classes are disabled, to avoid collecting performance data when it is not needed. This disablePMI setting was called disableEPM in WebSphere V3.5.x.
earFile		The full path to a file containing configuration data for starting the administrative server, and product in general.
jarFile		Paths to the JAR files required by the administrative server, such as: product_installation_root/lib/repository.jar; product_installation_root/lib/tasks.jar
logFile		The log files for recording administrative server events, such as: <WAS_HOME>/tranlog/was_tranlog1,<WAS_HOME>/tranlog/was_tranlog2
lsdHost		The host name of the administrative server.

Property	Default value	Description
lsdPort		Port number for the administrative server Location Service Daemon.
managedServerClassPath		Used by the application server.
nannyPort		The port on which the nanny process listens.
processPriority		Possible values are: ► 24 for Solaris ► 28 for AIX
seriousEventLogSize	1000	The number of entries to record in the serious events log. Allowing the log to grow too large can affect performance negatively.
qualifyHomeName	true	This value can be true or false.
traceString		The trace specification for the administrative server. See 24.6.4, “Tracing administrative server startup” on page 974 for further details.
traceOutput		The trace output file for the administrative server. See 24.6.4, “Tracing administrative server startup” on page 974 for further details.
wlm		Set to true to enable WLM of AdminServers. See 17.6, “Administrative server WLM” on page 625 for further details.

com.ibm.ejs.sm.util.process.Nanny package

Table B-3 Parameters in the admin.config file: com.ibm.ejs.sm.util.process.Nanny package

Property	Default value	Description
adminServerJvmArgs		Java command-line arguments for the nanny process.
path		Classpath for the nanny process, such as: <WAS_HOME>\bin;<WAS_HOME>\SQLLIB\bin;...
errtraceFile		The standard error trace file for the nanny process, such as: <WAS_HOME>\logs\adminserver_stderr.log
traceFile		The trace file for the nanny process, such as: <WAS_HOME>/logs/nanny.trace

Property	Default value	Description
maxtries	3	How many time the nanny process should attempt to start the administrative server before giving up.

com.ibm.itp package

Table B-4 Parameters in the admin.config file: com.ibm.itp package

Property	Default value	Description
location		The description is unavailable at this time.

com.ibm.websphere package

Table B-5 Parameters in the admin.config file: com.ibm.websphere package

Property	Default value	Description
preconfigured--CustomServices		Path to a file containing definitions for Custom Services that are to be preconfigured on every application server defined in the domain.

com.ibm.ws.jdk package

Table B-6 Parameters in the admin.config file: com.ibm.ws.jdk package

Property	Default value	Description
path		The path to the root directory of the JDK installed with the product, such as: <WAS_HOME>/jdk.

install.initial package

Table B-7 Parameters in the admin.config file: install.initial package

Property	Default value	Description
config	false	If set to true, the system attempts to create the default resources (such as the default application server) the next time you start the product administrative server. For WebSphere V4.0, you need to set both this setting and com.ibm.ejs.adminServer.createTables to true if you have dropped your WebSphere administrative database tables and need to create them again.
config.file		The full path to the property file defining the default configuration, such as: <WAS_HOME>/properties/initial_setup.config

server package

Table B-8 Parameters in the admin.config file: server package

Property	Default value	Description
root		Main path of WebSphere, that is <WAS_HOME>.

C

The plugin-cfg.xml file definitions

The configuration file for the HTTP transport plug-ins is a single XML file. It is installed in the WebSphere <WAS_HOME>/config directory by default and is normally named plugin-cfg.xml.

The location of the config file is configurable with all the Web servers supported except for Microsoft IIS and Lotus Domino Server. For these servers, the plugin-cfg.xml should be located as follows:

- Lotus Domino Server
  - On Windows NT/2000, the location of the config file is specified in the registry. The variable that is checked is HKEY_LOCAL_MACHINE/SOFTWARE/IBM/WebSphere Application Server/4.0/Plugin Config.
  - On UNIX platforms, the environment variable WAS_HOME is checked. If it is set, then /config/plugin-cfg.xml is appended to the value and that file must exist. If it is not set, it must exist at the default location.
    - On AIX this location is /usr/WebSphere/AppServer/config/plugin-cfg.xml.
    - On Solaris and HP-UX this location is /opt/WebSphere/AppServer/config/plugin-cfg.xml.

- Microsoft IIS
  - On Windows NT/2000, the location of the config file is specified in the registry. The variable that is checked is HKEY_LOCAL_MACHINE/SOFTWARE/IBM/WebSphere Application Server/4.0/Plugin Config.

A description and explanation of all elements and attributes applicable to the HTTP Transport plug-ins follows.

Elements and attributes

In this section we look at the elements and attributes used in the plugin-cfg.xml file. See Example C-1 on page 1072 for a sample plugin-cfg.xml file listing.

Example: C-1 Sample plugin-cfg.xml file

```
<Config>
   <Log LogLevel="Error" Name="D:\WebSphere\AppServer\logs\native.log" />
   <VirtualHostGroup Name="default_host">
      <VirtualHost Name="*:80" />
   </VirtualHostGroup>
   <ServerGroup Name="Default Server">
      <Server CloneID="suhqpp91" Name="Default Server">
         <Transport Hostname="itsohost" Port="9080" Protocol="http" />
      </Server>
   </ServerGroup>
   <UriGroup Name="perfServletApp/perfservlet_URIs">
      <Uri Name="/wasPerfTool" />
   </UriGroup>
   <Route ServerGroup="Default Server"
      UriGroup="itsohost/sampleApp/default_app_URIs"
      VirtualHostGroup="default_host" />
</Config>
```

You can get an idea of how the elements and attributes used in the plugin-cfg.xml file are related from the Rational Rose UML model in Figure C-1 on page 1073. Each Route element in plugin-cfg.xml is defined by a ServerGroup, a VirtualHostGroup, and a UriGroup. If a requested URL is found to match to a VirtualHost and a Uri assigned to the Route, the HTTP plug-in will dispatch the request to the next server in its ServerGroup.

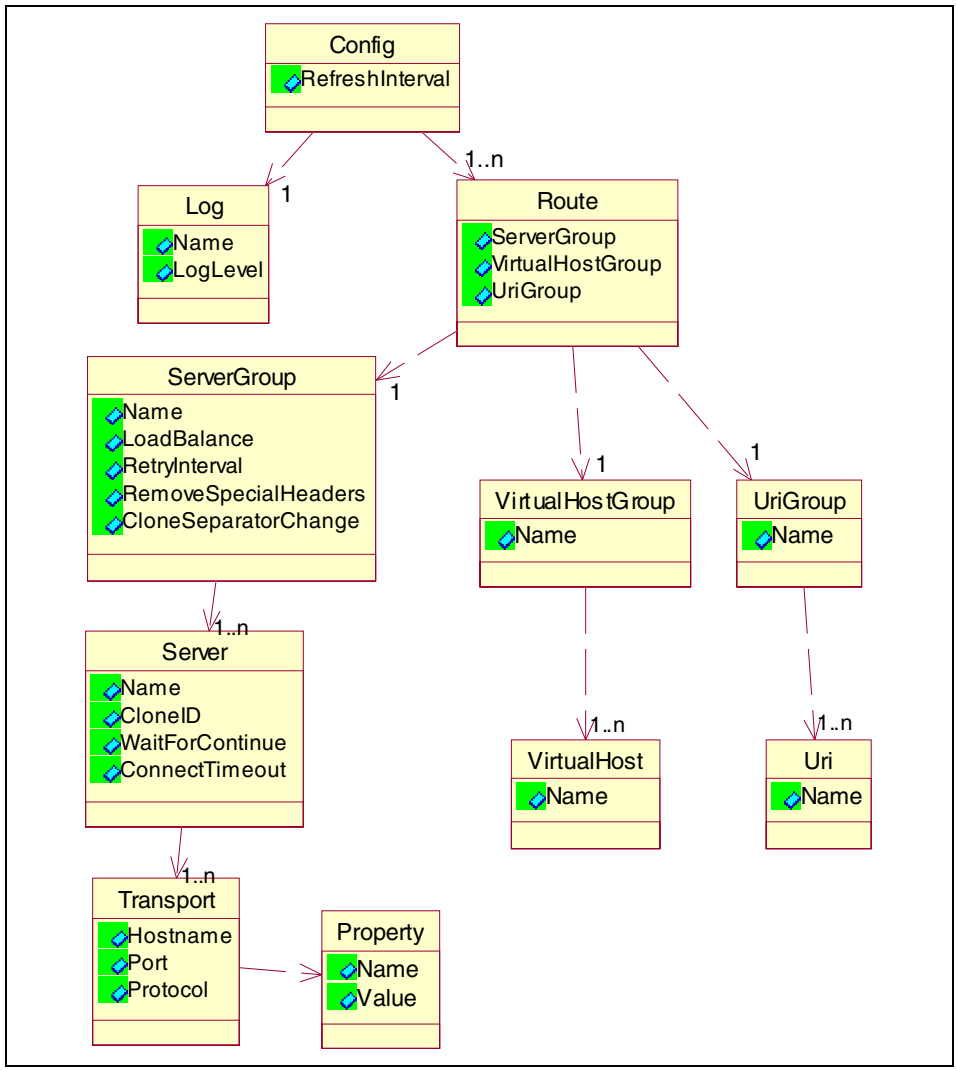


Figure C-1 plugin-cfg.xml element logical model

Config element

Config is the main element of the plug-in configuration. All other elements are contained within this element.

Config has the following attributes:

- RefreshInterval

RefreshInterval is the length of time in seconds that the plug-in should check the config file to see if updates or changes have occurred. The plug-in stats the file and checks to see if modifications have occurred since the last time the plug-in configuration was loaded.

Default value:

– 60 seconds

Expected value:

– Integer

Notes:

In a development environment where changes are usually going to be made often to the plug-in config file, a lower setting than the default is recommended. In production, a higher value than the default is a good idea, since updates to the configuration will not occur as often.

If the plug-in reload fails for some reason, a message is written to the plug-in log file and the previous configuration is used until the plug-in config reload is successful. If you are not seeing the changes you made to your plug-in configuration for some reason, make sure you check the plug-in log file for clues to what the problem is.

Log element

The log element allows you to specify information about what and where plug-in related messages should be logged.

Log has the following attributes:

- Name

  Name is the full path to the file that you want the plug-in to write messages to.

  Default value:

  – None

  Expected value:

  – Fully qualified path to log file

  Note: If the file does not exist then it will be created. If the file already exists then it will be opened in append mode and the previous plug-in log messages will remain.

- LogLevel

  LogLevel is the level of detail that you want written to the log file. Trace allows you to see the steps in the request process in detail. Warn and Error means that only information about abnormal request processing will be logged.

  Default value:

  – Error

  Expected values:

  – Trace, Warn, or Error

  Note: Be very careful about setting the level to Trace. A lot of error messages are logged at this level, which can cause the disk space to fill up very quickly. A Trace setting should never be used in a normally functioning environment, because it is a serious performance hit.

VirtualHostGroup element

The VirtualHostGroup element allows you to group VirtualHost definitions together that are configured to handle similar types of requests.

VirtualHostGroup has the following attribute:

- Name

  The name is an arbitrary string used to specify this group of virtual hosts.

  Default value:

  – None

  Expected value:

  – A character string identifier

VirtualHost element

The VirtualHost element defines a host name/port combination that the plug-in should look for to determine if a request should be handled by the application server or not.

VirtualHost has the following attribute:

- Name

  The plug-in can be configured to route requests to the application server based on the incoming HTTP host header and port for the request. Name allows you to specify what those combinations are.

Expected value:

– A host name or IP address and port combination separated by a colon.

Default value:

– None

Note: Wildcarding is available for this attribute. The only acceptable solutions are either a * for the host name, a * for the port, or a * for both. A * for both means that any request will match this rule. If no port is specified in the definition, the default HTTP port of 80 is assumed.

ServerGroup element

The ServerGroup element allows you to specify a group of application servers that are configured to handle the same types of requests.

ServerGroup has the following attributes:

- Name

  An arbitrary string that can be used to identify this group of servers.

  Default value:

  – None

  Expected value:

  – A character string identifier

- LoadBalance

  The type of load balancing that should be used for this server group.

  Default value:

  – Round Robin

  Expected values:

  – Round Robin or Random

  Note: The round robin implementation has a random starting point. This means that the first server picked will be done so randomly and then round robin will be used from that point forward. This is so that in multiple process-based Web servers, all of the processes don't start up by sending the first request to the same application server.

- RetryInterval

  The length of time that should elapse from the time that a server is marked down to the time that the plug-in will retry a connection.

Default value:

– 60 seconds

Expected value:

– Integer

- RemoveSpecialHeaders

  The plug-in adds special headers to the request before it is forwarded to the application server. These headers store information about the request that will need to be used by the application. By default, the plug-in will remove these headers from incoming requests before adding the headers it is supposed to add. This attribute should only be set to false if you truly understand what you are doing. It can be a security exposure to not remove these headers from incoming requests.

  Default value:

  – True

  Expected values:

  – True or false

- CloneSeparatorChange (WebSphere V4.0.2)

  The plug-in expects the colon (:) character by default to separate the CloneIDs when using session affinity. Several pervasive devices were not pleased with the colon, so by setting this attribute to true the plug-in will expect the separator to be the plus (+) character. In order to successfully use this you must set the appropriate property in the application server JVM so that the session manager uses the + character instead of the : character.

  Default value:

  – False

  Expected values:

  – True or false

Server element

Server has the following attributes:

- Name

  The arbitrary string used to identify this server.

  Default value:

  – None

Expected value:

– A character string identifier

- CloneID

  Used in conjunction with Session Affinity. When this attribute is set, the plug-in will check the incoming cookie header or URL for jsessionid. If the jsessionid is found, then the plug-in will look for a CloneID or CloneIDs. If CloneIDs are found and a match is made to this attribute, then the request will be sent to this server rather than being load balanced across the server group.

  Default value:

  – None

  Expected value:

  – A character string

  Note: If you are not using Session Affinity, then it is best to remove these CloneIDs from the configuration because there is added request processing in the plug-in when these are set. If CloneIDs are not in the plug-in, then it is assumed that session affinity is not on and the request is load balanced across the server group.

- WaitForContinue

  This attribute allows you to specify whether or not to use the HTTP 1.1 100 Continue support before sending the request content to the application server.

  Default value:

  – False

  Expected values:

  – True or false

  Note: This function is necessary when configuring the plug-in to work with certain types of proxy firewalls. The plug-in by default does not wait for the 100 Continue response from the application server before sending the request content because it is a performance hit.

- ConnectTimeout (WebSphere V4.0.2)

  When this value is set the plug-in will perform a non-blocking connect when making a connection to the application server. If the connection has not been successful in the specified number of seconds, the plug-in will mark the server down and attempt to send the connection to one of the other servers in the server group.

Default value:

– 0

Expected value:

– Integer

Note: By default the plug-in will perform a blocking connect, which means the plug-in will sit in the connect until the system default timeout expires or the connection is successful.

Transport element

The Transport definition requires the information needed by the plug-in in order to successfully send the request and read the response from the application server.

Transport has the following attributes:

- Hostname

  The host name or IP address that the application server is running on.

  Default value:

  – None

  Expected values:

  – Hostname or IP address

- Port

  The port that the application server is listening on.

  Default value:

  – None

  Expected value:

  – Integer

- Protocol

  The protocol that should be used for the communication between the application server and the plug-in. The default is for the communication to be non-secure. However, it is possible to further enhance security by encrypting this communication.

  Default value:

  – HTTP

Expected values:

– HTTP or HTTPS

Property element

Property element is a generic name value pair that can be used to supply information to other elements. Currently the Transport element is the only element that uses properties.

Property has the following attributes:

- Name

  The property name that is being defined. Supported names recognized by the transport are keyring, stashfile, and password.

  Default value:

  – None

  Expected value:

  – A character string

- Value

  The property value that is being defined.

  Default value:

  – None

  Expected value:

  – A character string

UriGroup element

The UriGroup element is a group of URIs that will be able to be handled by the same application server. The route will compare the incoming URI with the URIs in the group to determine if the application server will handle the request.

UriGroup has the following attributes:

- Name

  An arbitrary string used to identify this group of URIs.

  Default value:

  – None

Expected value:

– A character string identifier

Uri element

The Uri element is used to define URIs that will be handled by the application server.

Uri has the following attribute:

- Name

  The actual string that should be compared to the incoming request URI to determine if WebSphere should handle the request.

  Default value:

  – None

  Expected value:

  – A character string

  Note: Simple wildcarding is available so that you can specify rules such as *.jsp or /servlet/* to be handled by WebSphere

Route element

Route has the following attributes:

- VirtualHostGroup

  The virtual host group that should be matched for this route. The incoming host header and server port are matched to determine if this request should be handled by the application server.

  Default value:

  – None

  Expected value:

  – The name of one of the defined VirtualHostGroups

  Note: It is possible to leave this out of the route definition. If it is not present, then every request will match during the virtual host match portion of route determination.

- UriGroup

  The URI group that should be matched for this route. The incoming URIs for the request are matched to the defined URIs in this group to determine if this request should be handled by the application server.

  Default value:

  – None

  Expected value:

  – The name of one of the defined UriGroups

  Note: It is possible to leave this out of the route definition. If it is not present, then every request will match during the URI match portion of route determination.

- ServerGroup

  The server group that should be used to handle this request.

  Default value:

  – None

  Expected value:

  – The name of one of the defined ServerGroups

  Note: The server group that should be used to handle this request. If both the URI and the virtual host matching is successful for this route, then the request will be sent to one of the servers defined within this server group.

D

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

```
ftp://www.redbooks.ibm.com/redbooks/SG246176
```

Alternatively, you can go to the IBM Redbooks Web site at:

```
ibm.com/redbooks
```

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24-6176-00.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
sg246176.zip	Zipped code samples

Folder name	*Description*
CustomRegistry	Code samples for Chapter 21
Webbank	Sample application for Chapter 18, Chapter 19, Chapter 21, and Chapter 23
WSCP	Code samples for Chapter 23
XMLConfig	Code samples for Chapter 23

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	10 MB
Operating System:	Windows, AIX, Solaris, Linux
Processor:	500MHz Pentium, RS/6000, Sparc
Memory:	384 MB RAM minimum

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

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 1087.

- *Database Performance on AIX in DB2 UDB and Oracle Environments*, SG24-5511
- *WebSphere Version 4 Application Development Handbook*, SG24-6134
- *WebSphere Scalability: WLM and Clustering using WebSphere Application Server Advanced*, SG24-6153
- *IBM WebSphere V4.0 Advanced Edition Scalability*, SG24-6192
- *Web Services Wizardry with WebSphere Studio Application Developer*, SG24-6292
- *IBM WebSphere Advanced Edition: Security*, SG24-6520
- *WebSphere Application Server V4 for Linux, Implementation and Deployment Guide*, REDP0405

Other resources

These publications are also relevant as further information sources:

- Jim Gray, et al. *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, 1993, ISBN 1558601902
- Subrahmanyam Allamaraju, et al. *Professional Java Server Programming*, Second Edition, Wrox Press, 2000, ISBN 1861004656

Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://www.alphaworks.ibm.com`
  IBM alphaWorks

- http://www.ibm.com/software/ts/cics
  IBM CICS
- http://www.ibm.com/software/data/db2/udb
  IBM DB2 Universal Database
- http://www.ibm.com/developer
  IBM developerWorks
- http://www.ibm.com/software/webservers/httpservers
  IBM HTTP Server
- http://www.ibm.com/software/webservers/appserv
  IBM WebSphere Application Server
- http://www.ibm.com/websphere/developer
  IBM WebSphere Developer Domain
- http://www.ibm.com/software/webservers/appserv/infocenter.html
  IBM WebSphere InfoCenter
- http://www.ibm.com/software/ts/mqseries
  IBM WebSphere MQSeries
- http://www.ibm.com/websphere
  IBM WebSphere Software Platform
- http://www.info-zip.org
  Info-ZIP
- http://docs.iplanet.com
  iPlanet Documentation
- http://developer.iplanet.com/
  iPlanet for developers
- http://java.sun.com/j2ee
  Sun's Java 2 Platform, Enterprise Edition site
- http://java.sun.com/products/
  Sun's Java Technology Products & APIs site
- http://www.sun.com/solaris
  Sun Solaris
- http://sunsolve.sun.com/
  SunSolve
- http://marting.develop.com/certsrv
  Microsoft Certificate Services
- http://home.netscape.com/newsref/std/cookie_spec.html
  HTTP Cookies Specification

- http://www.ajubasolutions.com/
  Tcl Web site
- http://www.scriptics.com/
  Tcl Developers Exchange
- http://tcl.activestate.com/man/
  Tcl manual pages
- http://www.scriptics.com/software/java/
  Tcl Java integration (JACL)
- http://www.w3.org/TR/SOAP
  SOAP
- http://xml.apache.org/soap
  Apache SOAP
- http://www.uddi.org
  UDDI Organization
- http://oss.software.ibm.com/developerworks/projects/uddi4j/
  uddi4j Project
- http://www.w3.org/TR/wsdl
  WSDL
- http://xmethods.com
  XMethods

How to get IBM Redbooks

Search for additional Redbooks or redpieces, view, download, or order hardcopy from the Redbooks Web site:

ibm.com/redbooks

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

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.

Special notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of other companies:

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

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

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

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

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

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

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.

Abbreviations and acronyms

AAT	Application Assembly Tool
AE	Advanced Edition
AEd	Advanced Developer Edition
AEs	Advanced Single Server Edition
API	Application Programming Interface
BMP	Bean Managed Persistence
CA	Certificate Authority
CMP	Container Managed Persistence
DD	Deployment descriptor
DTD	Document type definition
DNS	Domain Name System
EIS	Enterprise information system
EJB	Enterprise JavaBeans
EAR	Enterprise Archive
HACMP	High-Availability Cluster Multiprocessing
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IBM	International Business Machines Corporation
IDE	Integrated Development Environment
IDL	Interface Definition Language
IMAP	Internet Message Access Protocol
ITSO	International Technical Support Organization
J2C	J2EE Connector
J2EE	Java 2 Platform, Enterprise Edition
JAF	JavaBeans Activation Framework
JAR	Java archive
JKS	Java keystore
JCA	J2EE Connector Architecture
JDBC	Java database connectivity
JDK	Java Development Kit
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
JSP	JavaServer Pages
JTA	Java Transaction API
JTS	Java Transaction Service
LDAP	Lightweight Directory Access Protocol
LSD	Location Service Daemon
LTPA	Lightweight Third Part Authentication
MIME	Multi-Purpose Internet Mail Extensions
MVC	model-view-controller
NIC	Network Interface Card
NAT	Network Address Translation
OLD	Object Level Debugger
OLT	Object Level Trace
ORB	Object Request Broker
OSE	Open Servlet Engine
PD	Problem Determination
PKI	Public-Key Infrastructure

POP3	Post Office Protocol
RAR	Resource Adapter Archive
RDBMS	relational database management system
RMI	Remote Method Invocation
SSL	Secure Sockets Layer
UDDI	Universal Discovery Description and Integration
URL	Uniform Resource Locator
WAR	Web archive
WAS	WebSphere Application Server
WLM	Workload Management
WSCP	WebSphere Control Program
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

X

Z

Redbooks

IBM WebSphere V4.0 Advanced Edition Handbook

(2.0” spine)
2.0” <-> 2.498”
1052 <-> 1314 pages

IBM WebSphere V4.0 Advanced Edition Handbook

Exploring WebSphere V4.0's new features, including J2EE support

Installing WebSphere V4.0 using popular platforms and topologies

Mastering WebSphere V4.0 administration and application deployment

This redbook explores in detail IBM's flagship application server offering, the IBM WebSphere Application Server V4.0, Advanced Edition. The topics covered include:

- Introduction to WebSphere Application Server V4.0 and J2EE enterprise applications.
- WebSphere V4.0 architecture and topology alternatives, Web services, and J2EE security.
- WebSphere V4.0 installation planning and installation steps on Windows, AIX, Solaris, and Linux platforms for commonly used topologies.
- WebSphere Administrative Console configuration tasks, including examples of how to package and deploy your J2EE enterprise applications.
- Advanced administrative tasks, including monitoring and tuning your environment, command-line administration, troubleshooting, and migration.

This redbook is a practical reference intended for system administrators, developers, and architects who need to implement the WebSphere V4.0 runtime environment, to package and deploy Web applications, and to perform ongoing management of the WebSphere environment.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6176-00 ISBN 0738423130