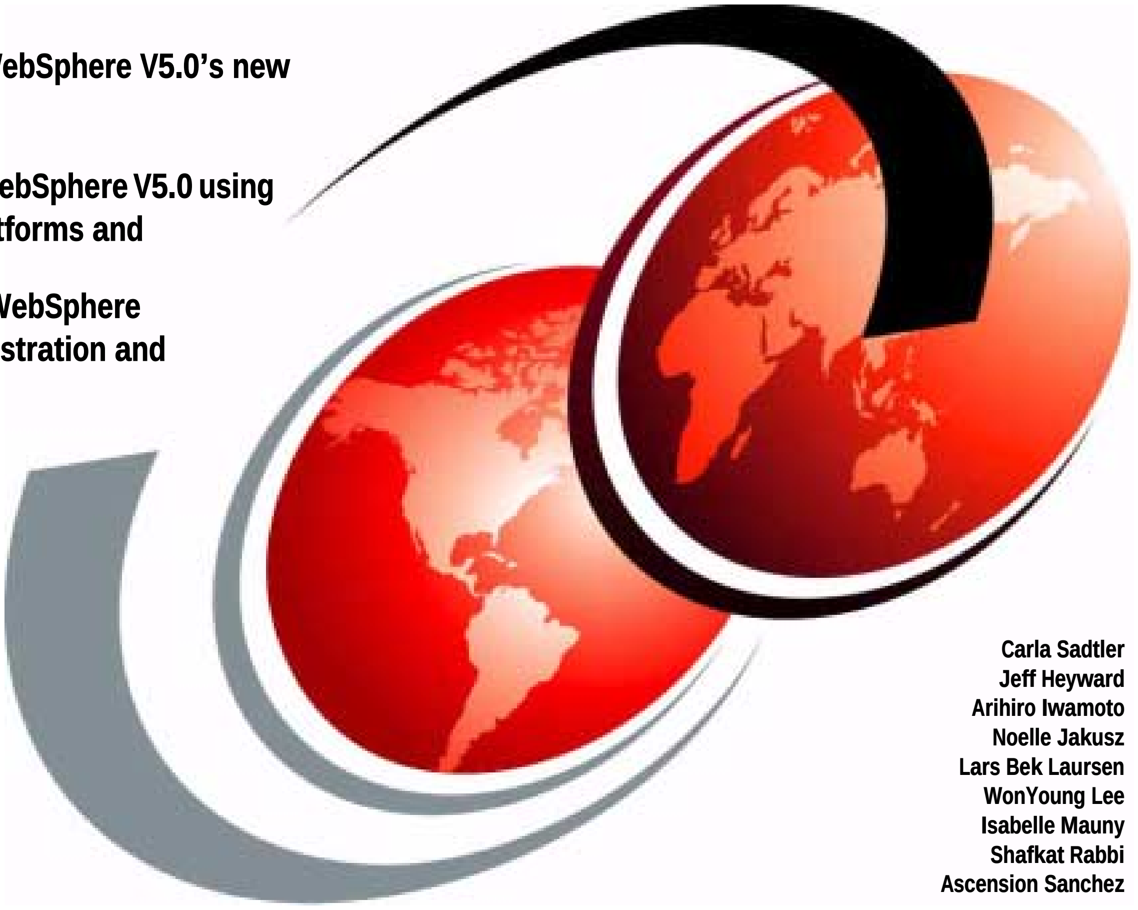
WebSphere® software

Draft Document for Review November 14, 2002 10:25 am

SG24-6195-00

IBM WebSphere Application Server Version 5.0 Handbook

WebSphere Handbook Series



Exploring WebSphere V5.0's new features

Installing WebSphere V5.0 using popular platforms and

Mastering WebSphere V5.0 administration and

Carla Sadtler
Jeff Heyward
Arihiro Iwamoto
Noelle Jakusz
Lars Bek Laursen
WonYoung Lee
Isabelle Mauny
Shafkat Rabbi
Ascension Sanchez

Redbooks

ibm.com/redbooks

International Technical Support Organization

WebSphere Application Server Version 5.0 Handbook

December 2002

SG24-6195-00

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

First Edition (December 2002)

This edition applies to Version ???, Release ???, Modification ??? of ???insert-product-name??? (product number ????-???).

Note: This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. We recommend that you consult the product documentation or follow-on versions of this redbook for more current information.

Contents

Notices

Trademarks

Preface

This redbook positions the new xxx against the yyy...

This redbook will help you install, tailor and configure the new...

This redbook gives a broad understanding of a new architecture...

This redbook will help you design/create a solution to migrate...

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.

Carla Sadtler is a ??title??????? at the International Technical Support Organization, Raleigh Center. He/she writes extensively and teaches IBM classes worldwide on all areas of ???????. Before joining the ITSO ???? years ago, ??name???? worked in ???department, country??? as ???????

Jeff Heward is a ????title??? in ???country???. He/she has ?? years of experience in ???? field. He/she holds a degree in ???? from ????. His/her areas of expertise include ?????? He/she has written extensively on ???????.

Isabelle Mauny is a ????title??? in ???country??? He/she has ?? years of experience in ???? field. He/she has worked at IBM for ??? years. His/her areas of expertise include ?????? He/she has written extensively on ???????.

Ascension Sanchez is a ????title??? in ???country???. He/she has ?? years of experience in ???? field. He/she holds a degree in ???? from ????. His/her areas of expertise include ?????? He/she has written extensively on ???????.

Noelle Jakusz is a ????title??? in ???country???. He/she has ?? years of experience in ???? field. He/she holds a degree in ???? from ????. His/her areas of expertise include ?????? He/she has written extensively on ???????.

Arihiro Iwamoto is a ????title??? in ???country???. He/she has ?? years of experience in ???? field. He/she holds a degree in ???? from ????. His/her areas of expertise include ?????? He/she has written extensively on ???????.

Shafkat Rabbi is a ????title??? in ???country???. He/she has ?? years of experience in ???? field. He/she holds a degree in ???? from ????. His/her areas of expertise include ?????? He/she has written extensively on ???????.

is a ????title??? in ???country???. He/she has ?? years of experience in ???? field. He/she holds a degree in ???? from ????. His/her areas of expertise include ?????? He/she has written extensively on ???????.

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

??????????
International Technical Support Organization, Raleigh Center

??????????
IBM ??????????

Department 7RK, WebSphere Technology and Training
IBM

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an Internet note to:

  `redbook@us.ibm.com`

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-6195-00
for WebSphere Application Server Version 5.0 Handbook
as created or updated on November 14, 2002.

December 2002, First Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

▸
▸

Changed information

▸
▸

Part 1

Getting started

Note to Author: Optionally, describe the book part here. If you are not using Part files, you need to restart the page numbering in the first chapter file of your book:

- In FrameMaker 5.5: **Open .book > select first chapter> File > Set Up File > Page Numbering: > Restart at 1 > Set**
  - Now delete the three Part files (p01, p02 and p03) from your book: **Open .book > select a file > File > Rearrange Files > Delete > Done**
- In FrameMaker 6.0: **Open .book > select first chapter> Format > Document > Numbering > Page "tab" > select First Page # radio button and set Page # to 1 > Set**
  - Now delete the three Part files (p01, p02 and p03) from your book: **Open .book > select a part file > Edit > Delete File from Book**

In this part we introduce/provide/describe/discuss...

1

Introduction to WebSphere Application Server V5.0

WebSphere Application Server V5.0 is a J2EE 1.3(Java 2 Enterprise Edition) compliant runtime environment. The J2EE 1.3 specification details the following:

- APIs that developers must use to build applications
- Runtime services that must be provided by an application server
- The communication protocols that the servers and applications running on those servers must communicate on.

These requirements allow applications that have been built to this specification and design to reap benefits such as, scalability, security, standardization, and performance.

WebSphere has implemented J2EE services and administration tools so that administrators can configure these services to take on the benefits of the J2EE architecture. This redbook will identify the J2EE services and how to configure them. It will also address some additional services (beyond J2EE 1.3) provided by the WebSphere Application Server V5.0.

In this book, we provide a detailed exploration of the WebSphere Application Server Version 5, Network Deployment runtime environment. We will provide a overview of WAS Express and WAS Enterprise. Readers interested in a more detailed explanation may also want to look at the following redbooks:

- Express Redbook (Link)
- Enterprise Redbook (Link)
- Web Services
- Security

Author Comment: Need details for the redbooks to be placed as references in the introduction

1.1 Introduction to WebSphere Application Server

IBM WebSphere Application Server, Version 5 delivers flexible configuration and deployment options to meet needs for stand-alone, multiserver distributed and highly dynamic decentralized distributed environments. As your e-business requirements change, you can migrate smoothly to the greater functionality and higher qualities of service offered by other configurations.

IBM WebSphere Application Severs have several major themes that drive its new functionality and services:

- J2EE 1.3 Compliance
- Web Services
- JMX Administration Model

1.1.1 J2EE 1.3 Compliance

J2EE compliance has been one of IBM's main motivations for building WebSphere V5.0. It was necessary for IBM to comply with the latest J2EE standards. IBM wants to support the latest trends and industry standards as they become available and the new specifications have significant improvements that IBM can use to increase the capabilities of the WebSphere product.

The J2EE 1.3 specification has provided new development specifications. Servlet 2.3, JSP 1.2, and EJB2.0 provide added API libraries for distributed application processing as well as integrated messaging support. More details on these specifications can be found in the J2EE Technologies Section

Author Comment: Not sure where the J2EE Section will be or what it will be called, but need a reference here.

1.1.2 Web Services

Another motivation for the basis of WebSphere is the improvement of IBM's provision for Web Services. WebSphere has improved its SOAP support in this release and also ships with the Private UDDI. The private UDDI allows customers to become the service provider without having to comply with any additional requirements by the platform. They can use the Private UDDI to publish their services directly, rather than have a third-party broker provide that service. IBM has also included a Web Services Gateway which will allow customers to integrate with other Web Services in heterogeneous Web Services environments. More details on the enhancements for Web Services can be found under the What's New in WebSphere section.

Link to Web Services redbook

1.1.3 JMX Administration Model

IBM has chosen to redesign the administrative model in WebSphere to adopt a standard resource management interface. JMX (Java Management Extensions) is a new framework that has been added to the Java language. JMX allows you to wrap all of your resources (hardware and software) in Java and expose them in a distributed environment. JMX also provides a mapping framework for integrating existing management protocols, such as SNMP, into JMX's own management structures.

Some of the benefits of using a JMX administrative model are:

- Improved availability
- Reduction of interdependencies among process
- Increased usability of resources
- Adoption of a standardized framework for resource management

More details on JMX can be found in the Section JMX.

1.2 WebSphere Application Server V5.0 packaging

The matrix described in Table 1-1 on page 6 provides a listing of the options available to WebSphere 5.0 customers and the features provided in each product set.

Author Comment: Do we need to cover development packaging in this redbook? it is now covered in the WebSphere Studio Section, though not sure it belongs there.

Table 1-1 Matrix of packages and features.

	IBM WebSphere Application Server Express	**IBM WebSphere Application Server, Base**	**IBM WebSphere Application Server, Network Deployment**	**IBM WebSphere Application Server, Enterprise**
Application Server	Light-weight version	yes	yes	yes
IBM HTTP Server	Embedded	yes	yes	yes
Application Client	no	yes	yes	yes
Application Server Toolkit	no	yes	yes	yes
DataDirect Technologies JDBC Drivers for WAS	no	yes	yes	yes
Edge Components	no	no	yes	yes
Deployment Manager	no	no	yes	yes
DB2 Universal Database Edition V7.2	no, Cloudscape is provided for samples	no	yes	yes
IBM Directory V4.1	no	no	yes	yes

	IBM WebSphere Application Server Express	IBM WebSphere Application Server, Base	IBM WebSphere Application Server, Network Deployment	IBM WebSphere Application Server, Enterprise
Enterprise Extensions (PME)	no	no	no	yes
WebSphere MQ V5.3	no	no	no	yes

1.3 Express, Base, Network Deployment, and Enterprise features

This section describes the features that have been listed in Table 1-1. The features have been separated by package and will be listed so that the first package provides the base and each additional package will only include the features specific to that package.

1.3.1 IBM WebSphere Application Server Express

Author Comment: This is the only place we will cover scalability, security, web services, etc. Need to talk about them pretty thoroughly here and then point to the upcoming books.

IBM WebSphere Application Server Express provides a lightweight version of the Application Server, a CloudScape database for development use, and WebSphere Studio Site Developer. This product targets web and web services providers.

1.3.2 IBM WebSphere Application Server, Base

The base installation provides the following components:

- Application Server

  This is the primary component in WebSphere. The base application server includes the code for the V5.0 Application Server which is in full compliance with J2EE 1.3. The application server provides containers that specialize in executing specific java applications. Containers provide different Declarative and runtime services. Declarative services in J2EE are provided through the use of deployment descriptors. Deployment descriptors allow the declaration of services outside the scope of our application components. A declarative service is a service that is provided by the container on the behalf of the component developer.

A typical J2EE server will provide a container for each application component type: web component container, enterprise bean container, application client container, and applet container.

– The Web Container

  The Web container runs an embedded HTTP server for exchanging requests and response over HTTP or HTTPS. This exchange occurs between the embedded HTTP Server and either a Web Server plug-in or a web browser. Web browser clients connect to dynamic content through the HTTP Server and the HTTP Serve plug-in. The Plug-in will map the HTTP requests to the associated resource running in the web container.

– The EJB Container

  The EJB container is an execution environment for EJB components. EJBs live in the container and the container provides services for the component. The EJB Container often lives in an application server, which provides an execution environment for it and other containers.

  See the J2EE Chapter for more details.

– The Application Client Container

  A container installed on the client machine for referencing Web applications and EJB Resources

– The Applet Container

  A web browser used to host Java applets

Other container services include:

– Life cycle MAnagement of application components
– Resource pooling
– Populating of JNDI Namespace based on the deployment names associated with EJB components
– Populating the JNDI Namespace with objects necessary for utilizing container service APIs
– Clustering

WebSphere Application Server also includes code for the Node Agent, which will be dormant if the server is used in a single server environment. The base server also includes support for Web Services and provides an enhanced version of SOAP.

- HTTP Server

  The web server used by WebSphere is IBM HTTP Server V1.3.22. The package also include a preview of IBM HTTP Server 2.0. The web server provided the capability for supporting communication over HTTP and HTTPS.

- Application Server Toolkit

  The application server toolkit includes a number of Eclipse-based tooling products. It is a minimal footprint version of development tooling that corresponds to the base tooling provided with WebSphere Studio Application Developer. The toolkit includes a debugger, a tool for application profiling, a log analyzer, and a workbench. It is installed separately from the base.

- Application Client

  The IBM WebSphere Application Server ships with several different types of application clients. An application client is the interface an application will use to communicate to services provided by applications running on WebSphere.

  – The *ActiveX application client* model, uses the Java Native Interface (JNI) architecture to programmatically access the java virtual machine (JVM). Therefore the JVM exists in the same process space as the ActiveX application (Visual Basic, VBScript, or ASP) and remains attached to the process until that process terminates.

  – In the *Applet application client* model, a Java applet embeds in a Hyper Text Markup Language (HTML) document residing on a remote client machine from the WebSphere Application Server. With this type of client, the user accesses an enterprise bean in the WebSphere Application Server through the Java applet in the HTML document.

  – The *J2EE application client* is a Java application program that accesses enterprise beans, JDBC databases, and Java Message Service message queues. The J2EE application client program runs on client machines. This program follows the same Java programming model as other Java programs; however, the J2EE application client depends on the application client run time to configure its execution environment, and uses the Java Naming and Directory Interface (JNDI) name space to access resources.

  – The *Pluggable* and *thin application clients* provide a lightweight Java client programming model. These clients are best suited in situations where a Java client application exists but the application needs enhancements to use enterprise beans, or where the client application requires a thinner, more lightweight environment than the one offered by the J2EE application client. The difference between the thin application client and the pluggable application client is that the thin application client includes a Java virtual machine (JVM), and the pluggable application client requires a JVM be provided by the user.

- DataDirect Technologies JDBC Drivers for WAS

  Because J2EE-certified JDBC drivers are essential in any successful J2EE application model, DataDirect Technologies (formerly the DataDirect business unit of MERANT) has dedicated their research and development team to creating and supporting JDBC drivers based on only the industry's highest quality and reliability standards.

  WebSphere V5.0 now supports the following JDBC Drivers:

  – JDBC drivers provided by a supported database

  – DataDirect type 3 (SequeLink)

  – DataDirect type 4 (ConnectJDBC)

1.3.3 IBM WebSphere Application Server, Network Deployment

Network Deployment includes all of the features of the base installation as well as the following:

- Deployment Manager

  The Deployment manager is a process provided by the application server to communicate with independent node agents (installed with the base WAS), running in each node in the cell as lightweight (partial J2EE environment) JVM processes. Node agents coordinate such events as configuration synchronization. The Deployment Manager manages all the nodes in a distributed topology.

- Edge Components

An improved version of Edge Components is currently being shipped with

Author Comment: Refer to Scalability book for more details.

WebSphere 5.0. These Edge Components provide a workload management solution, that is fully integrated with the WebSphere platform. As part of the workload management solution, a Workload Management Controller can be implemented to provide end-to-end monitoring and weighting of WebSphere Servelt and Enterprise Java Bean containers. This will enable dynamic changes in the workload distribution as the load on the back-end servers change. This elimates having to rely on pre-defined static metrics. The load balancing attributes provided through the Edge Components are exposed through Java

Management Extensions (JMX). This allows an administrator to use exposed methods to start and stop the controller, set timeouts, set network communication parameters, set log levels, as well as high availability parameters. The edge components in this package are:

- Workload Management Controller
- Switch Consultants

  The network dispatcher provides switch consultants for CISCO and Nortel switches and a Site Selector which provides a scaling network dispatcher. Consultant code can be used to optimize server performance within a CISCO to Nortel infrastructure. Consultants generate server weighting metrics and distribute them to Cisco CSS 11000 switches or Nortel Alteon 180 series of switches for optimal server selection, load balancing and fault tolerance.
- Edge-of-network caching technology

  Caching is used to reduce network congestion by storing frequently accessed content so information only has to be retrieved once.Information can be saved, or cached, based on when the information should be updated, expiration date, or how large the cache is. By using caching one can decrase download times resulting in better quality of service for customers and reduced load on back-end servers.
- Transactional Quality of Service (TQoS)

  Through integrated edge technology, TQoS ensures that all customers recieve the service they deserve as the business grows beyond its traditional boundries to accept new customers, partners and suppliers.
- Content Distribution Clients and Servers (CDC, CDS)

  Through Edge-base content distribution technology, companies can deploy website content, to include web pages, fragments of pages (such as stock quotes and individual catalog prices), and application components, to caches and other servers in the network. With CDC and CDS you can automate the placement, distribution, and syhcnonization of web content.

► DB2 Universal Database Enterprise Edition V7.2

DB2 had been included in the application server package to provide databases session persistence, as well as EJB persistence management. Since DB2 is no longer required to provide the administrative repository, DB2's main focus is to provide a persistence option out of the box.

Author Comment: Is the above true for Solaris?

- IBM Directory V4.1

  IBM Directory Server provides tight integration with IBM operating systems, middleware, identity-management and security products and complies with the LDAP (Light-weight Directory Access Protocol) versions 2 and 3.

  LDAP directories are a core component of e-business infrastructures and are being integrated with many operating systems. A LDAP directory is a repository of data. The data held in an LDAP directory includes information about identity, security, applications, systems and network management. LDAP is a standard directory schema that provides a key integration point for all network data.

- UDDI Registry

  UDDI stands for Universal Description, Discovery, and Integration. UDDI is a specification that helps to locate services provided through a service broker. It is an essential element for brokers to use to run a registry of services.It enables the enterprise to run its own Web Services broker within the company or provide brokering services to the outside world.

- Web Services Gateway

  The gateway allows Web Services to be exposed so that they can be invoked outside the firewall in a managed and secure environment.

1.3.4 IBM WebSphere Application Server Enterprise

- Programming Extensions (PME)

  WebSphere Extensions are a set of tools that extend the basic capabilities of the Websphere Programming model and runtime. The extensions provide high level objects and frameworks for incorporating monitoring, scheduling, messaging and workflow. The following is a brief description of each extension:

  – Dynamic Access Intent: The purpose of this extension is to specify application specific runtime information for Enterprise Java Beans. This runtime information can be supplied per component, per application, accessed dynamically at runtime and configured at deployment time using the Application Assembly tool. Dynamic access intent improves performance by allowing the fine tuning of runtime behavior in the EJB container.

  – Activity Sessions

    Activity Sessions complement the JTA transaction service in WebSphere. It represents a UOW (Unit of Work) without the baggage of a global transaction.It uses a context propagation mechanism which is provided by the standard OMG activity service to provide a distributed context for

WebSphere. ActivitySessions allow you to map transactional requests to the scope of a HTTPSession.

- Dynamic Query Capabilities

  Dynamic Query Service (DQS) is an extension to the EJB QL specification allows queries to be created and executed at runtime. Using this services queries do not have to be predefined in the deployment descriptor.

- Container Managed Messaging

  CMM is built on top of the Message Driven Bean implementation in the base version of WebSphere. It provides the capability of integrating asynchronous messaging with Enterprise Java Beans. This enhanced messaging allows for detailed separation of business logic from messaging logic. CMM beans provide a level of abstraction for the messaging logic. This is accomplished through the use of receiver beans and sender beans. These components control all messaging logic so that EJBs can focus on executing the business logic for the request, rather than be consumed with the transmission of that request.

- WebSphere Workflow

  Workflow enables the automation and facilitation of business interactions within and outside a company. WebSPhere supports workflow management with the following services:

  - Enterprise Access Builder (EAB) micro-flows
  - MQ Adapter Service, adapter flows (MQAO)
  - MQ Workflows (MQWF)
  - Lotus Workflows (LWF)

  Websphere also has the ability to manage workflows through the following tools:

  - Flow Editor
  - WorkList Manager
  - Staff Editor

- Asynchronous Beans

  Asynchronous beans provide the ability to add subsystem monitoring through the use of event listeners. Alarms scheduling can also be implemented through the use of asynchronous beans enabling the scheduling of a piece of code to execute at a specified time. These beans also enable object pooling which can help performance by reducing garbage collection.

- Scheduler

The scheduler is a service that enables the implementation of time-based business functionality. It can be used to execute tasks at specified times or at intervals. It also provides an extension to Asynchronous Beans alarms. The scheduling service is provided by IBM WebSphere Application Server, which provided integrated management of services, integration into a clustered configuration, and WAS will also handle all service persistency.

– Startup Beans

WebSphere provides stateful session EJBs to handle the execution of application specific code at application startup or before application shutdown. Each EAR or EJB modules can have several startup beans. Startup beans are supported in a clustered environment, as well as startup prioritization.

- WebSphere MQ V5.3

Author Comment: Need information on MQ as provided with WAS Enterprise

1.4 WebSphere Development Packaging

WebSphere also has packages that are specifically for development use. The packages are geared towards software providers. The packages include:

- IBM WebSphere Application Server for Developers
- IBM WebSphere Application Server Enterprise for Developers
- IBM WebSphere Application Server for Developers (Trial Use Only)- Web only

Table 1-2 on page 14 lists the features available with each development package.

Table 1-2

	IBM WebSphere Application Server for Developers	**IBM WebSphere Application Server Enterprise for Developers**	**IBM WebSphere Application Server for Developers (Trial Use Only)- Web only**
Application Server	yes	yes	yes
Application Client	yes	yes	yes
Application Server Toolkit	yes	yes	yes

	IBM WebSphere Application Server for Developers	**IBM WebSphere Application Server Enterprise for Developers**	**IBM WebSphere Application Server for Developers (Trial Use Only)- Web only**
DataDirect Technologies JDBC Drivers for WAS	yes	yes	no
DB2 Personal Developer’s EditionV7.2	yes	yes	yes
IBM Directory V4.1	no	yes	no
WebSphere MQ V5.3	no	yes	no

1.5 WebSphere Application Server V5.0 support for OS and database platforms

1.5.1 Distributed platform operating system support

The following is a list of the operating systems and their respsective levels supported under IBM WebSphere Application Server v5.0:

- Windows 2000, Advanced Server SP2
- Windows NT, SP 6a
- AIX, 4.3.3, 5.1
- Linux/Intel
  - RedHat 7.2
  - SuSE 7.2
  - SuSE SLES 7
  - MontaVista Prof. 2.1
- Linux/390
  - TurboLinux Server 6.5
  - SuSE SLES 7
- Solaris 8 and up

- OS400, 5.1, 5.2

1.5.2 Distributed platform database support

- The following databases are supported under WIN2000, NT, AIX, Linux/Intel, Linux/390, and Solaris:
  - CloudScape 5.0.3 (for Samples shipped with the product)
  - DB2 7.2 FP6
  - DB2 for 390 6.1
  - Oracle Enterprise Edition 8i Rel. 3 and 9i
- SQL Server Enterprise 7.0 SP2 is only supported on the NT platform.
- SQL Server Enterprise 2000 is only supported on the WIN2000 platform.
- Sybase 12.0 is supported on NT, AIX and Solaris.
- Informix 7.31 and 9.3 is supported on WIN2000, NT, AIX, and Solaris.
- DB2 400 is supported on the OS400 platform.

2

What’s new in WebSphere V5.0

This chapter provides a brief description of the improvements and additions to IBM WebSphere Application Server V5.0 from previous releases. There are many changes to Systems Management in WebSphere 5.0.Some of the changes include the elimination of the administrative repository, a move to a web based administrative application, and support for the new services provide under J2EE 1.3. The following changes or enhancements will be discussed in this chapter:

- What’s new in J2EE 1.3
- New Packaging for WebSphere editions
- Performance enhancements
- Security Enhancements
- Administration tools
- Enhanced Problem Determination Tooling

2.1 What’s new in J2EE 1.3

Author Comment: Will make a reference to the content in the J2EE section of the book for each of these topics.

New in J2EE 1.3:

- PAckaging and Deployment: Installed RAR
- EJB 2.0
  - Local Interfaces
  - New CMP
  - Home Methods
  - Message Driven Beans
  - HAndle Delegates
- Servlet 2.3
  - Servlet Filtering
  - Application Life-cycle Listeners and events
  - Enhanced Internationization
  - Overall API CHanges
- JSP 1.2
  - XML Views for JSPs
  - New Tag Support for Iteration
  - Tag Support for Application Life cycle events
  - Tag library Life cycle improvements

2.2 New packaging for WebSphere Editions

The new packaging for WebSphere was created to [WHAT?]. This allows customers with different levels of business requirements to leverage the same technology provided under J2EE 1.3. The following is a list of the different packages of WebSphere Application Server:

2.2.1 IBM WebSphere Application Sever, Express

Author Comment: Reference to Express Redbook

2.2.2 IBM WebSphere Application Server, Base

- Support for JDK 1.3.1
- J2EE 1.3, Servlet 2.3, JSP 1.2, EJB 2.0
- Enhanced Web services including WSF, WS-Security, and a technology preview of JSR109
- Integrated Java Messaging Service
- Enhanced problem determination features including FFDC (First Failure Data Capture)
- Enhanced security features:
  - JAAS
  - CSIv2 interoperability
  - Java2 security
  - Support for third party security providers
- Enhanced support for HTTP session state fail-over
- Browser-based Systems Administration

2.2.3 IBM WebSphere Application Server Network Deployment

In addition to the base features:

- Distributed systems management, security, and directory support
- Workload management, clustering and fail over support
- Dynamic network caching
- UDDI repository and Web Service gateway support
- XML file based configuration

2.2.4 IBM WebSphere Application Server Enterprise

- Container managed messaging
- Dynamic EJB query service
- Shared work areas

- Service Choreography
- Asynchronous beans
- Dynamic query service
- Application profiling

2.3 Workload Management Enhancements

Workload Management is a key feature in providing:

- Scalability

  Scalability allows us to configure WebSphere to serve more users. As our need to accept new customers, create new partnerships and branch to more suppliers grows so does our need to scale to larger architectures to handle these new requests. WebSphere WLM enables the applications running under WebSphere to be scaled to any number of machines at any time, increasing the amount of requests the applications can serve.
- Load Balancing

  This feature allows an application to share the workload fairly across machines participating in the enterprise. Most of the enhanced features concern the ability to balance the workload appropriately.
- Availability.

  Another basic need is to have the server be able to function even in the case of a system failure. Workload Management enables the identification of systems that will take over system processing in the case of a server failure.

WebSphere introduces the concept of cells, nodes and clusters to enable workload management. There is a one to many relationship between clusters and cells.

Note: The idea of a cluster and cell replaces the Model/Clone and Server Group as the WLM domain in WebSphere v5.0.

A feature that is new with WebSphere V5.0 is Server Weighted Round Robin Routing. A routing table will be provided for the HTTP plug-in and the ORB in the EJB Client for each cluster. The administrator can set a weight value for each cluster, the routing table will then be decremented on each new request. Once the routing table reaches zero it will no longer take any new requests. This is only over ridden for the following reasons:

- Transaction and Session Affinity

- In Process (handled by the ORB)
- Prefer Local

When all servers reach zero the routing table(s) are reset. As a best practice it is recommended to use low weight values. This will help to avoid load variations on the server/cluster.

Author Comment: More details here, why is this a best practice? Or should we just refer the reader to the Redbook.

Server Weighted Round Robin Routing replaces the random and round robin routing for EJB WLM, and is added to the existing routing algorithms (round robin and random) for HTTP WLM.

There are also primary and backup server lists that can be provided to the HTTP plug-in that can be created, which can improve HTTP Session Fail over Routing as well as primary/backup server clusters that can be specified for EJB WLM. This provides the ability for IIOP Requests to fail over to a cluster in the same or in a a different cell.

2.4 Security Enhancements

WebSphere v5.0 extends is security support by enhancing its authentication options to include Kerbos tokens. These tokens provide strong authentication security for client/server applications. If you would prefer to use a to use an alternate security authentication or authorization security implementation, WebSphere 5.0 provides Security Programming Interfaces (SPIs) for Integration into those third party solutions. There is an SPI policy file associated with the server. It can be located at <WAS install directory>\DeploymentManager\config\cells\<cell>\nodes\<node name>r

Note: In previous versions of WebSphere we had a sas.server.props property file that we could use to configure security options provided by the Security Associations Services in WebSphere. This properties file has been replaced by secutiry.xml at the server and cell levels. The SAS service has been replaced by CSIv2.

In addition, there are three files, at the cell level, that we can use to configure security.

- security.xml

- admin-authz.xml
- naming-authz.xml

2.4.1 Common Secure Interoperatability (CSIv2)

CSIv2 has replaces the SAS security features that were provided in previous versions of WebSphere. With the enhanced features of CSIv2 you can:

- Choose between SSL and TCP/IP
- Enable SSL client Authentication
- Provide an optional BasicAuth client login
- Configure identity assertion
- Choose between stateful and stateless.

CSIv2 is an OMG specification that implements the CORBA Security Service. It is now a required part of the J2EE 1.3 platform. By using this security implementation you can enable multi-vendor Interoperatability for your security implementation. This is a benefit because most companies have investments in security servers and now those servers can communication with the security service in WebSphere. The following list describes the concepts involved in implementing CSIv2 security.

- Interoperatability Object Reference (IOR) Registration
- Configuring Listener Ports
- Setting client request policy
- Configuring client request flow

Author Comment: Reference to Security Red book... for more details.

2.4.2 Java Authentication and Authorization Service (JAAS)

In J2EE 1.3 the is of JAAS becomes required as the API for authentication. This Authorization model extends the Java 2 security model by providing subject-based authorization and authenticating identities.

JAAS provides several components and interfaces to standardize its authentication, they are as follows:

- Login Configuration
- LoginModule

- LoginContext

The Login Configuration specifies the login module(s) to be used in an application. The LoginContext provides a programmatic interface to JAAS that programmers can communicate through method calls at runtime.

You can configure JAAS services through the administrative console in WebSphere. It is provided under JAAS configuration in the Security Center.

2.4.3 Web Services Security

WebSphere 5.0 provides a technical preview to security services specific Web Service implementations. Web Services Security provides security handlers for the client and server to interface with WebSphere security services.

The security handler intercepts SOAP messages and modifies them based on the target service security requirement supplied through configuration files in WebSphere. The server will validate each request, based on the security configuration. However, since this is a technical preview, there is only support for request messages. The server will sign those messages with digital signatures and no encryption is supported. This service is fully integrated in to WebSphere security, So in order for WebSphere to get the security information the authentication information will be part of the SOAP message.

2.5 New Administration tools

The Administrative console is now web-based and can be viewed using a web browser. WebSphere will also provide command line tools, as it has done with previous versions, but this time those tools have been standardized. The tool known as wscp has been replaced by a JMX alternative WSAdmin. XMLConfig Tool has been removed since the Admin repository has been removed. There is no longer any need to have a tool that converts relational admin data to XML, now that the administrative data is stored in XML documents.

2.6 Web services

What is Web Services? It is a technology for building self-contained, self-describing modular applications that can be published, located, and invoked across the Web. Think of Web Services as Legos® with logic, It has interlocking blocks that create open distributed systems. A service is basically a URL with XML-in and XML-out.

2.6.1 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.

2.6.2 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 2-1 on page 24:

- 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 2-1 Web services components and operations

2.6.3 Web services components

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.

2.6.4 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.

2.6.5 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 2-1 shows a sample SOAP request, and Example 2-2 on page 27 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 2-1 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 2-2 SOAP response

```
HTTP/1.1 200 OK
Server: IBM_HTTP_SERVER/1.3.x Apache/1.3.x (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.

2.6.6 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.

2.6.7 WebSphere 5.0 Support for Web Services

Compared to WebSphere 4.0 the support has been enhanced in this version. The following table represents the version comparison between version4.0 and version 5.0 of WebSphere. These versions will be important to know because code written to one version of the specification will not necessarily be compatible with the runtime environment that uses another specification.

Table 2-1 Web services support

WAS Version	**WAS 5.0**	**WAS 4.0**
SOAP	2.2	2.2

WAS Version	WAS 5.0	WAS 4.0
SOAP Client Integration	2.2	none
UDDI	2.0	1.0.4

The main enhancements in this version is the use of a private UUDI registry and an enhanced Web Services Gateway. They are described in the following sections:

UDDI Private vs. Public Registries

The public registry is a UDDI business registry (UBR), you can publish your services to a UBR, and anyone with access to that UBR will be able to utilize those services.There is no validity check done on the web service. This kind of registry is provided to publicly distribute and utilize existing web services.

A private registry is new to WebSphere 5.0. This provides the ability for a company to create a registry that has access control policies defined by an administrator. Rather than using a public UBR, a company can create their own registry by installing and running a separate UDDI registry product, provided in the WebSphere package. This allows a company to have complete control over the administration and content of a web service.

Web Services Gateway

This middle ware component provides a framework for invoking web services between internet and intranet environments. The gateway provides the ability to enable the use of web services across administrative boundaries. The main benefit to the gateway is that it allows interoperatability between web services deployed on different vendor platforms. The gateway will abstract the vendor details and publish and serve web service requests based on the standard protocols and transports.

Some other benefits to using a web service gateway is that you can leverage your existing infrastructure by creating web services as a means of communicating between systems in you enterprise. The gateway is deployed as a J2EE application and can be managed by the WebSphere Administrator.

2.7 Performance enhancements

WebSphere v5.0 provides tooling that enables intelligent end-to-end application optimization. This means that administrators can now handle volume and performance tuning dynamically. This service is provided by WebSphere through the use of the following enhancements:

2.7.1 Java Management Extensions (JMX)

WebSphere's support for JMX provides Java components that will log and record statistics on usage and resources. These objects are then exposed to third party JMX-compliant applications used for monitoring. Because this information is now provided through an industry standard interface. Administrators can use best-of-breed tooling that is tightly integrated into their enterprise for managing performance data and the monitoring process.

2.7.2 Performance Monitor Interface

WebSphere provides a Performance Monitor Interface (PMI) that allows you to capture and manage a wide range of performance metrics. This interface allows you to not only capture metrics defined in the PMI, but WebSphere and application specific metrics as well.

2.7.3 Tivoli Performance Viewer

Another tool that can be used to enhance performance is the Tivoli Performance Viewer. This tool provides the ability to configure smart auto-tuning parameters that will automatically make recommendations to tune critical WebSphere parameters for maximized performance

Note: The Tivoli Performance Viewer was previously known as the Resource Analyzer.

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?

Author Comment: THINGS TO ADD:
- Requirement for a container to support both 1.2/1.3 applications
- Insist in new stuff : EJB 2.0 mainly....
- Removed Classloader section, which was a duplicate of deploying chapter.

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.

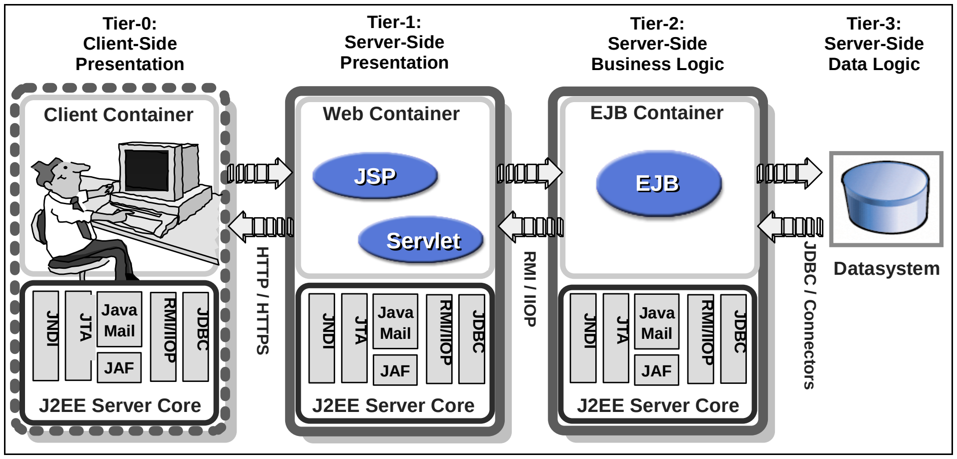


Figure 3-1 J2EE server model

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

	API	WebSphere V3.5.2+	WebSphere V4.0
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 44 for our discussion on containers.

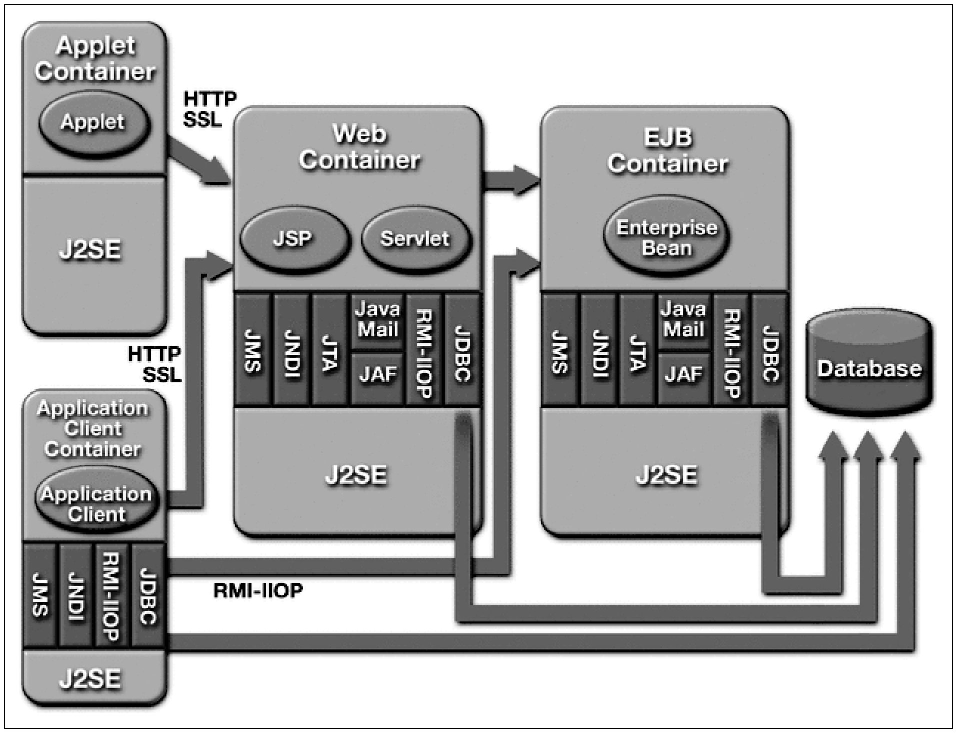


Figure 3-2 J2EE components, containers, and services

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.

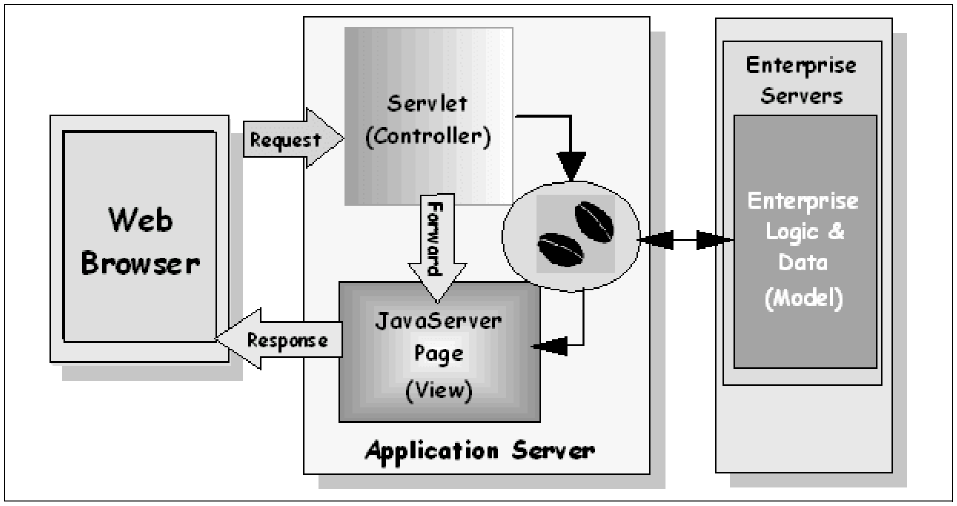


Figure 3-3 The Model-View-Controller design model

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.

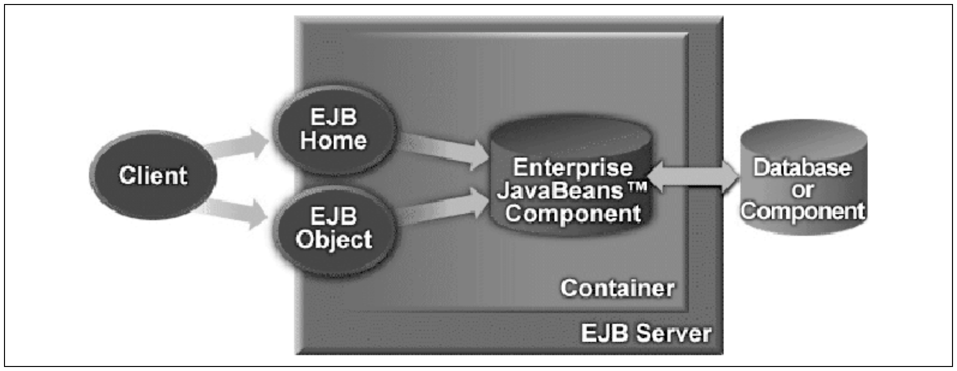


Figure 3-4 Enterprise Java Beans Architecture

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 37. 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 38.

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 50.

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.

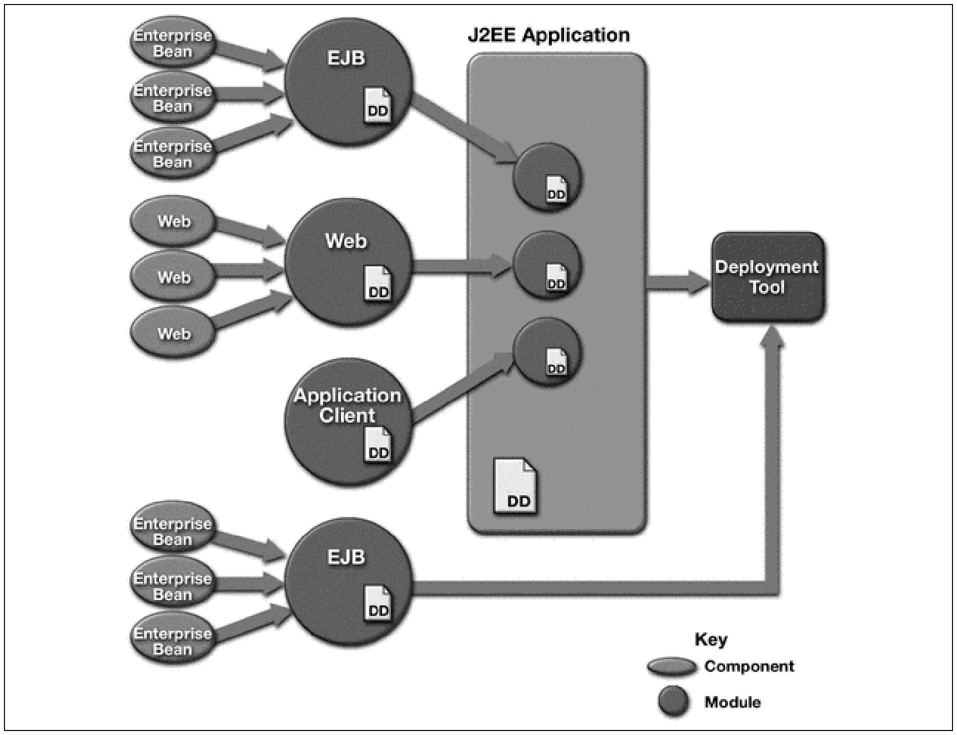


Figure 3-5 J2EE packaging and deployment overview

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 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

Author Comment: Insert conclusion + transition to next chapter.

4

WebSphere architecture overview

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

- IBM WebSphere Application Server runtime architecture and components.
- IBM WebSphere Application Server Network Deployment runtime architecture and components.
- Server clusters.
- Managed servers / processes.

4.1 Introduction

IBM WebSphere Application Server is IBM’s implementation of J2EE (Java 2 Enterprise Edition) platform, conforming to the Java 2 Platform Enterprise Edition (J2EE) Specification, v1.3.

In this section we discuss the runtime architecture and components of two different configurations of this product:

- IBM WebSphere Application Server.
- IBM WebSphere Application Server Network Deployment.

Note: Subsequent chapters of this redbook will specifically address the IBM WebSphere Application Server Network Deployment configuration.

4.2 IBM WebSphere Application Server

In this section we describe the runtime architecture and components of IBM WebSphere Application Server (base).

A base configuration includes only the application server process. There is no node agent or Deployment Manager involved in this configuration (required by the Network Deployment configuration described in Section 4.3, “IBM WebSphere Application Server Network Deployment” on page 65).

No coordination between application server processes is supported in the base configuration, with each application server instance having to be separately administered.

4.2.1 Component diagram

The IBM WebSphere Application Server (base) runtime architecture consists of the following components, as shown in Figure 4-1.

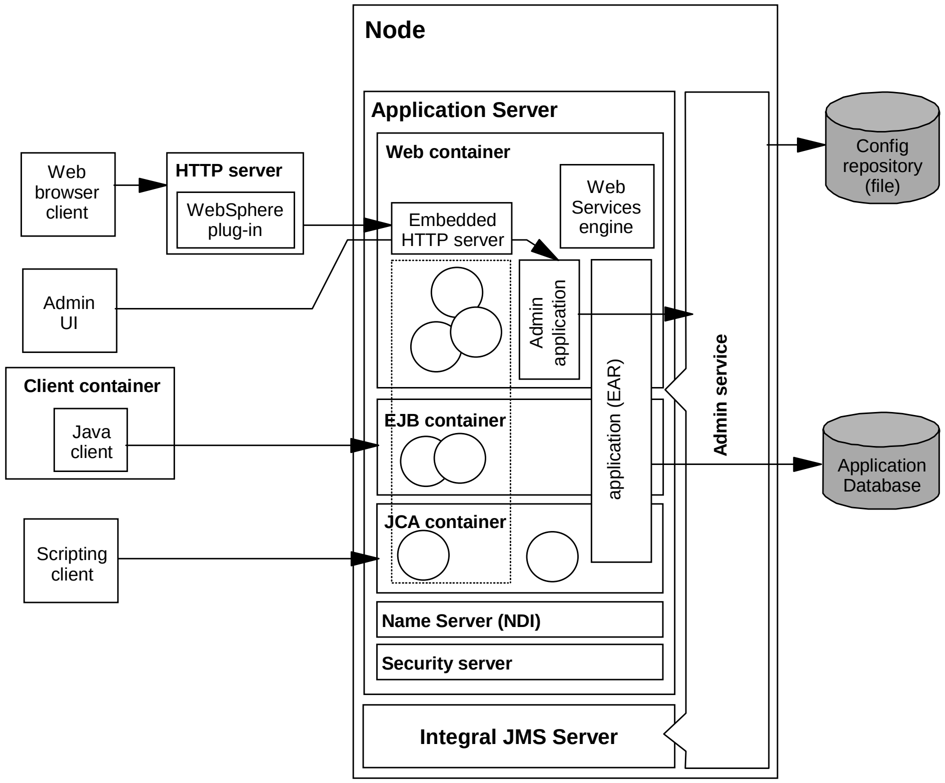


Figure 4-1 IBM WebSphere Application Server components

4.2.2 Node

Logical grouping of WebSphere managed server processes, sharing common configuration and operational control.

In the base configuration, each application server is responsible for its own configuration in the configuration repository.

Note:

- A standalone (base) node does not have a node agent server defined.
- A node is identified by a logical name for configuration purposes. In the case of the Network Deployment configuration, no two nodes in the same cell can have the same name.

4.2.3 Configuration repository

Repository that holds copies of all of the individual component configuration XML documents.

In WebSphere 5.0, all configuration information is stored in XML files. The database used for the administrative repository in earlier versions of WebSphere Application Server, is not needed in WebSphere 5.0.

The application server's admin service takes care of the configuration and makes sure it is consistent during the runtime.

To find out more about the WebSphere 5.0 configuration repository, see Chapter 12, “System Management” on page 405.

4.2.4 Application server

The application server is the primary component of WebSphere. It runs in a Java Virtual Machine (JVM), providing the runtime environment for the application's code. The Application server provides containers that specialize in enabling the execution of specific Java application components. There are three containers in the application servers:

- Web container
- EJB container
- J2C container

Web browser clients connect to the Web container through the HTTP server and HTTP server plug-in to access the dynamic content. Standalone Java applications, either J2EE application clients or thin Java clients, connect to the EJB container to access EJBs and invoke methods through RMI/IIOP.

Web container components can use EJB resources within the application logic. Application servers can access a shared database for storing data.

The Application server provides other services besides the containers:

- Object Request broker (ORB)
- Name service (JNDI)
- Security service (JAAS and Java 2 security)
- Admin service (JMX)
- Trace service
- Performance Monitoring Interface (PMI)
- Transaction management

- Messaging interfaces (JMS)
- E-mail interfaces (JavaMail)
- Database connection (JDBC) and connection pooling

4.2.5 HTTP Server and plug-in

The WebSphere Application Server works with an HTTP server (Web server), to handle requests for dynamic content, such as servlets, from Web applications. The HTTP server and application server communicate using the WebSphere HTTP plug-in specific to a HTTP server.

The HTTP plug-in is available for popular Web servers, including:

- IBM HTTP Server
- Apache Web Server
- Microsoft Internet Information Server (IIS)
- Netscape iPlanet

Note: We use the terms HTTP server and Web server interchangeably thoughout this book.

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, but can also be configured to use secure HTTPS, if required.

To find out more about the WebSphere Web server interface, see Chapter 16, “Configuring the Web server interface” on page 549.

4.2.6 Embedded HTTP server

A key feature of IBM WebSphere Application Server 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 16, “Configuring the Web server interface” on page 549 for further details on using WebSphere without an external Web server.

4.2.7 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. It is a configuration, rather than a "live object", which is why it can be created, but not started or stopped.

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.

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.

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 HTTP 404 error is returned to the browser.

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

IBM WebSphere Application Server provides two default virtual hosts:

- **default_host**

  Used for accessing the default applications.

  For example:

  ```
  http://localhost:80/servlet/snoop
  http://localhost:9080/servlet/snoop
  ```

- **admin_host**

  Specifically configured for accessing the WebSphere 5.0 web-based administration tool. Other applications are not accessible through this virtual host.

  For example:

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

See Chapter 16, “Configuring the Web server interface” on page 549 for more information on virtual hosts.

4.2.8 Web container

The WebSphere 5.0 Web container processes servlets, JSP files and other types of server-side includes. Prior to 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.

Web container configuration provides information about the application server component that handles 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(s) on which the Web container listens for incoming HTTP(S) requests.

The WebSphere admin clients can be used to edit the configurations of Web containers. Each application server runtime has one logical Web container, which can be modified, but not created or removed.

The web container runs an embedded HTTP server for handling HTTP(S) requests from external HTTP server plug-ins or web browsers.

4.2.9 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, the 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 host more than one EJB JAR file.

4.2.10 JCA container

The Java Connector Architecture (JCA) container is a component provided by WebSphere Application Server, into which JCA resource adapters from EIS vendors can be plugged-in, configured and used by JCA compliant applications.

For information on jow to configure WebSphere support for JCA resource adapters, see Chapter 18, “Configuring WebSphere resources” on page 623. For information on JCA containers, see the J2EE Connector specification at:

`http://java.sun.com/j2ee/connector/`

4.2.11 Client application container

The client application container is a separately installed component on the client's machine. It allows the client to run applications in an EJB-compatible J2EE environment.

There is a command-line executable (**`launchClient`**) that is used to launch the client application along with its client container runtime.

Note: Instead of the single global namespace used by WebSphere 4.0, WebSphere 5.0 uses a distributed namespace as described in Chapter 6, “Naming” on page 105. This affects J2EE clients as each client runs in its own Java Virtual machine (JVM). As a consequence, each client must be specifically configured to bind to the namespace root of a chosen Application Server, node agent or Deployment Manager.

4.2.12 Admin UI

The WebSphere 5.0 administration interface (Admin UI) runs in a web browser client connecting to the admin application hosted by the Application server. Users assigned to different admin roles can manage the Application server and certain components and services using this interface.

In the base configuration, the admin application is installed and run on the Application server. In the Network Deployment configuration, it is installed and run on the Deployment Manager only (by default).

In the base configuration, the Admin UI always connects to the Application server and can manage only that application server.

See Chapter 13, “WebSphere administration basics” on page 449 for a detailed description of the web administration application and WebSphere 5.0 administration tasks.

4.2.13 Admin application

The WebSphere 5.0 web-based administration interface is installed as a standard J2EE 1.3 compliant web application. The application is packaged in a WebSphere specific J2EE 1.3 Enterprise Application Archive (EAR).

See Chapter 13, “WebSphere administration basics” on page 449 for a detailed description of the web administration application.

4.2.14 Admin Service

The Admin service runs within each WebSphere 5.0 server JVM. In the base configuration, the Admin service runs in the Application server. In the Network Deployment configuration, each of the following WebSphere 5.0 servers host an Admin service:

- Deployment Manager
- Node agent
- Application server
- JMS server

The Admin service provides the necessary functions to create/update/remove configuration data for the application server and its components. The configuration is stored in a repository; the config repository is basically a set of XML files that configures the application server. The files are stored in the application server's file system.

Note:

1. Application servers are attached to nodes and nodes can be part of a cell in the network deployment environment. In this environment the deployment manager is responsible for managing all the application servers in the cell, which means that the administrator has access to multiple application servers under one user interface through the Deployment Manager.
2. The Admin service running in a particular server is only responsible for that server.

Admin services has a course-grain admin security control and filtering functionality, providing different levels of administration to certain users or groups using the following admin roles:

- Administrator
- Monitor

- Configurator
- Operator

4.2.15 Scripting client

The scripting client **`wsadmin`** provides extra flexibility over the web-based administration application, allowing the administration of applications from a simple the command-line, with no graphical interface.

Using the scripting client not only makes administration quicker, but helps automate the administration of multiple application servers and nodes using scripts.

The scripting client uses the Bean Scripting Framework (BSF), which allows a variety of scripting languages to be used for configuration and control of WebSphere 5.0.

Important:

1. The version of **`wsadmin`** provided with WebSphere 5.0 only provides support for the JACL scripting language.
2. WebSphere 5.0 does not include the WebSphere 4.0 **`WSCP`** and **`XMLConfig`** administration tools. The functionality of both tools (and much more) has been merged into the single **`wsadmin`** scripting tool.
3. Scripts written to use **`WSCP`** will need to be rewritten to work with **`wsadmin`**.

See Chapter 23, “Command line administration and scripting” on page 971 for a detailed description of the scripting client and examples of its use.

4.2.16 Integral JMS server

The integral JMS (embedded messaging) server is a complete JMS server that runs in the application server. It supports point-to-point and publish/subscribe styles of messaging and is integrated with the transaction management service.

The integral JMS server is used for:

- Support of Message-Driven Beans.
- Messaging within a WebSphere cell.

In the base configuration, the JMS server runs in the same JVM as the application server. In the Network Deployment configuration, the JMS Server is separated from the application server and run in a separate dedicated JVM.

See Chapter 7, “Asynchronous messaging” on page 147 for a detailed description of embedded messaging and the different configurations supported by the integral JMS Server.

4.2.17 Application

Applications are custom designed and developed applications that are hosted and run by the application server. An application is packaged into an Enterprise Application Archive (EAR) that is deployed to one or more application servers.

A J2EE 1.3 application contains any or all of the following modules shown in Table 4-1.

Table 4-1 J2EE 1.3 application modules

Module	Filename	Contents
Web module	<module>.war	Servlets, JSP files, and related code artifacts.
EJB module	<module>.jar	Enterprise beans and related code artifacts.
Application client module	<module>.jar	Application client code.
Resource adapter module	<module>.rar	Library implementation code that your application uses to connect to enterprise information systems (EIS).

See Chapter 19, “Packaging an application” on page 691 and Chapter 20, “Deploying an application” on page 835 for details on the packaging and deployment of applications respectively to WebSphere 5.0.

4.2.18 Application database

Data storage is an essential part of many applications. The application database runs on a database server in an enterprise system where multiple application servers can share the same database.

4.2.19 Session database

In a multi-server environment, session information can be stored in a central session database for session persistence. The multiple application servers hosting a particular application need to share this database information in order to maintain session states for the stateful components.

An alternative approach is to use the memory-to-memory session replication functionality provided by WebSphere 5.0. See Chapter 17, “Configuring session management” on page 551 for further details.

4.2.20 Name server

Each application server JVM hosts a name service that provides a Java Naming and Directory Interface (JNDI) namespace. The service is to register all EJBs and J2EE resources (JMS, J2C, JDBC, URL, JavaMail) hosted by the application server.

Note: The WebSphere 5.0 JNDI implementation is built on top of a Common Object Request Broker Architecture (CORBA) naming service (CosNaming).

See Chapter 6, “Naming” on page 105 for further information on the WebSphere 5.0 naming architecture and components.

4.2.21 Security server

Each application server JVM hosts a security service that uses the security settings held in the configuration repository to provide authentication and authorization functionality.

Author Comment: Reference required here to the WAS 5 Security redbook.

4.2.22 Web services engine

The Web services engine does not really stand as a separate component. The application server implements numerous APIs for additional services. Web services is provided as a set of APIs in cooperation with the J2EE applications.

WebSphere's Web services engine is based on AXIS, and implements the following specifications:

- SOAP (Simple Object Access Protocol)

  A protocol that defines the messaging between Objects. It is based on the XML and XML Schema specification.
- WSDL (Web Services Description Language)

  Describes the services that can be located and used by applications.
- UDDI (Universal Description, Discovery and Integration)

  Enables an application to find services on the network published by service brokers.

- WSIF (Web Services Invocation Framework)

  A tool that provides a standard API for invoking services described in WSDL, no matter how or where the services are provided. The architecture allows new bindings to be added at run time.

4.3 IBM WebSphere Application Server Network Deployment

In this section we describe the runtime architecture and components of IBM WebSphere Application Server Network Deployment configuration. The discussion is limited to those components of IBM WebSphere Application Server Network Deployment that are not part of the base IBM WebSphere Application Server configuration. For a description of components common to both configurations, see Section 4.2, “IBM WebSphere Application Server” on page 54.

A Network Deployment configuration includes support for multiple nodes, each with a node agent process and several application servers, all coordinated within an administrative cell by the Deployment Manager process. Clusters of load balanced application servers can be configured within a Network Deployment cell.

The configuration and application binaries of all components in the cell are managed by the Deployment Manager and synchronized out to local copies on each of the nodes.

4.3.1 Component diagram

The IBM WebSphere Application Server Network Deployment runtime architecture consists of the following components, as shown in Figure 4-2.

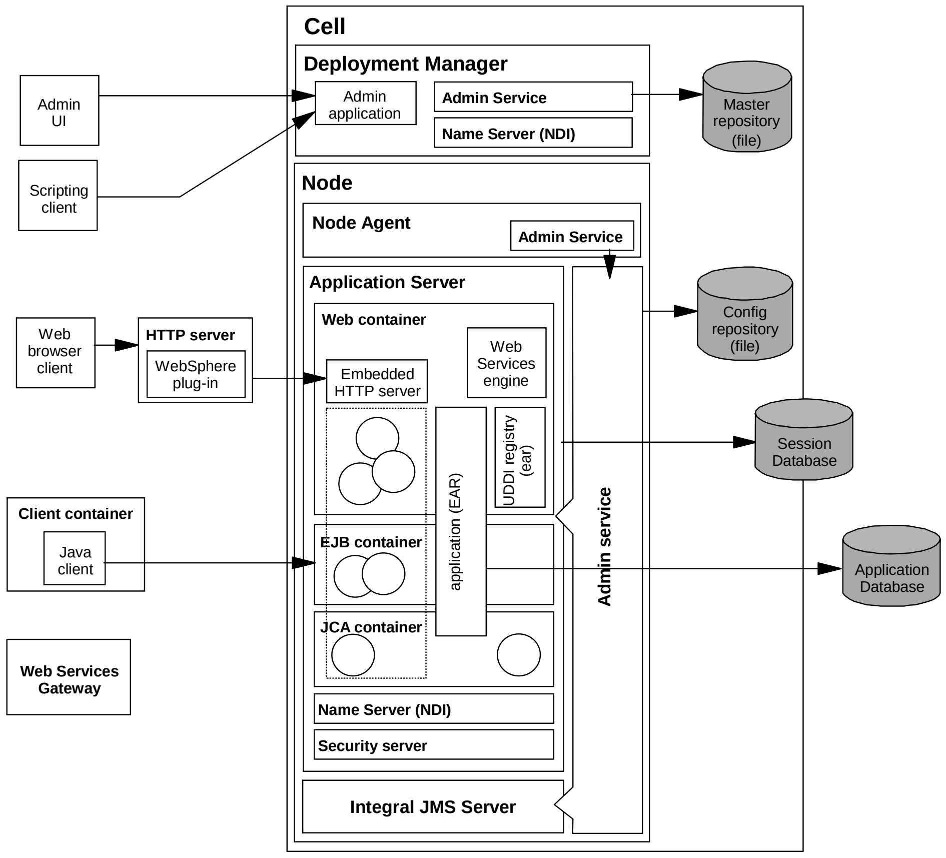


Figure 4-2 IBM WebSphere Application Server Network Deployment components

4.3.2 Cell

A WebSphere 5.0 cell is a grouping of nodes into a single administrative domain.

The configuration and application binaries of all nodes in the cell are centralized into a cell master configuration repository. This centralized repository is managed by the Deployment Manager process and synchronized out to local copies held on each of the nodes.

Note: A cell is identified by a logical name for configuration purposes.

For further information on the WebSphere 5.0 cell, see Chapter 12, “System Management” on page 405.

4.3.3 Deployment Manager

The Deployment Manager process:

- Provides a single, central point of administrative control for all elements of the entire WebSphere distributed cell. Administrative tools that need to access any managed resource in a WebSphere 5.0 cell usually connect to the Deployment Manager as the central point of control.
- Provides a single point to access all configuration information and control for a cell. It is responsible for the content of the repositories (configuration and application binaries) on each of the nodes.
- Aggregates and communicates with the node agent process resident on each node of the.
- Hosts the web-based administrative console.
- Controls clusters and balancing the work load of application servers across the nodes.

Using the Deployment Manager, horizontal scaling, vertical scaling and distributed applications are all easy to administer and manage, since application servers are managed by nodes, and one or more nodes is managed by a cell.

For a detailed description of how the Deployment Manager and Node Agents cooperate to provide distributed administration, configuration and management in WebSphere 5.0, see Chapter 12, “System Management” on page 405.

4.3.4 Master configuration repository

The master configuration repository contains all of the cell’s configuration data. All updates to this repository are performed by the Deployment Manager process.

The configuration repository at each node is a synchronized subset of the master repository. The node repositories are read-only for application server access, since only the Deployment Manager can initiate their update, by pushing configuration changes out from the cell master configuration repository.

For further details on the WebSphere 5.0 master configuration repository, see Chapter 12, “System Management” on page 405.

4.3.5 Node agent

The node agent is an administrative process and is not involved in application serving functions. It hosts important administrative functions such as:

- File transfer services

- Configuration synchronization
- Performance monitoring

The node agent aggregates and controls all of the WebSphere managed processes on its node, by communicating with:

- Deployment Manager - to coordinate configuration synchronization and to perform management operations on behalf of the Deployment Manager.
- Application Servers and JMS Servers - to manage (start/stop) each server and to update its configuration and application binaries as required.

Note: Only one node agent is defined (and run) on each node.

For further information on the WebSphere 5.0 node agent, see Chapter 12, “System Management” on page 405.

4.3.6 UDDI registry

UDDI stands for Universal Description, Discovery and Integration, which is a specification that helps to locate services at the service broker. Basically it is an essential element for Web services brokering to run a registry. It enables the enterprise to run its own Web services broker within the company or provide brokering services to the outside world.

Note:

- The WebSphere 5.0 UDDI registry is installed as a J2EE 1.3 compliant web application. It is packaged in a J2EE 1.3 Enterprise Application Archive (EAR).
- The registry needs to be installed by the Administrator into each Application Server thatvrequires its services.

For detailed information on the UDDI registry provided with WebSphere 5.0, see the WebSphere 5.0 online InfoCenter at:

`http://www-3.ibm.com/software/webservers/appserv/infocenter.html`

4.3.7 Web services gateway

The Web services gateway is a middleware component that bridges the gap between Internet and Intranet environments during Web service invocations. It is used to manage:

- Web services.

- Channels that carry requests to and responses from the web services.
- Filters that act upon the web services.
- References to UDDI Registries in which web services can be registered.

The gateway builds upon the Web services Definition Language (WSDL) and the Web Services Invocation Framework (WSIF) for deployment and invocation.

A Web service is deployed to the Web Services Gateway by deploying a WSDL file which describes how the Web Services Gateway should access it. The WSDL file can be deployed to a UDDI Registry or to a URL. Requests passing through the Web Services Gateway can be sent to a Java class, an Enterprise Java Bean (EJB), a SOAP server or a SOAP/JMS server, including another gateway.

A request to the Web Services Gateway arrives through a channel, is translated into an internal form, then passed through any filters that are registered for the requested service, and finally sent on to the web service implementation. Responses follow the same path in reverse.

For further information, see the WebSphere 5.0 online InfoCenter at:

`http://www-3.ibm.com/software/webservers/appserv/infocenter.html`

4.3.8 WebSphere Edge components

The Edge components provide a mechanism for efficiently and economically hosting Web-accessible content and provide Internet access.

The software usually runs on machines close (in a network configuration sense) to the boundary between an enterprise intranet and the Internet. The Caching Proxy and Load Balancer are Edge components that help Administrators to provide better service to internal and external users who access documents stored on server machines at the enterprise.

The Load Balancer component consists of a set of subcomponents that create servers to direct network traffic flow, reduce congestion bottlenecks and balance the load on various other services and systems. You install Load Balancer subcomponents between the Internet and your back-end servers. Load Balancer servers can act as either entry points to the Internet, point-of-presence machines, or internal network servers. They provide site selection, workload management, session affinity, and transparent failover. This load balancing is transparent to users and other applications.

Note: The WebSphere Edge components are shipped on a separate CD with the WebSphere V5 product. The CD is called the DMZ CD.

For further information, see the Edge components online InfoCenter at:

`http://www-3.ibm.com/software/webservers/appserv/ecinfocenter.html`

4.3.9 Cluster

A WebSphere 5.0 cluster is a logical collection of application server processes, with the sole purpose of providing workload balancing.

Application servers that belong to a cluster are "members" of that cluster and must all have idential application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any other configuration data.

For example, one cluster member might be running on a large multi-processor server while another member of that same cluster might be running on a small laptop. The server configuration settings for each of these two cluster members are very different, except in the area of application components assigned to them. In that area of configuration, they are identical.

The members of a cluster can be located on a single node (vertical cluster), across multiple nodes (horizontal cluster) or a combination of the two.

> **Note:** WebSphere 5.0 does not provide an equivalent of the ServerGroup-Clone functionality available in WebSphere 4.0. A WebSphere 5.0 server cluster is a grouping of *separately configured application servers* that all happen to host the same application.

See Chapter 15, “Server Configuration” on page 525 for information on configuring WebSphere 5.0 server clusters.

4.4 Managed servers / processes

All operating system processes that are components of the WebSphere product are called managed servers / processes. This means that the servers all participate in the administrative domain. WebSphere JMX support is embedded in all managed processes and these processes are available to receive administration commands and to output administration information about the state of the managed resources within the processes.

IBM WebSphere Application Server Network Deployment provides the following managed servers / processes:

- Deployment Manager (Network Deployment only)

Provides a single point to access all configuration information and control for a cell. The DeploymentManager aggregates and communicates with all of the NodeAgent processes on each node in the system.

- Node Agent (Network Deployment only)

  Aggregates and controls all of the WebSphere managed processes on its node. One NodeAgent per node only.

- Application Server (Base and Network Deployment)

  Managed server that hosts J2EE applications.

- JMS Server (Network Deployment only)

  Managed server that hosts the embedded messaging service for a node.One JMSServer per node only.

5

Topology selection

This chapter describes various topologies which IBM WebSphere Application Server Network Deployment supports. In addition, options regarding the use of application server clusters, multiple cells, vertical and horizontal scaling, as well as multiple tiers (separating the Web containers and EJB containers) are covered.

Author Comment: Review all mention of WLM & security to see if we can't provide a little more info. Everything is very generic, yet we don't cover it anywhere else.

Author Comment: Review to see if we need to define the nodes (from patterns). Also make sure the concept of cluster is clear .. each server runs the same app. Plus terms that we use interchangeable (Web server = HTTP server, load balancer node = ip sprayer).

5.1 Topology selection criteria

A variety of factors come into play when considering the appropriate topology, or configuration, for a WebSphere deployment. The selection criteria typically includes a review of your requirements in the following areas:

- Security
- Performance
- Throughput
- Scalability
- Availability
- Maintainability
- Session management

This section will review these factors, their availability and support provided by IBM WebSphere Application Server Network Deployment.

5.1.1 Security

Security concerns usually require a physical separation of the Web server from the application server processes, typically across one or more firewalls. A common configuration would be the use of two firewalls to create a demilitarized zone (DMZ). Information in the DMZ has some protection based on protocol filtering between the Internet and the DMZ. A Web server intercepts the requests and forwards them on through the next firewall. The sensitive portions of the application and business data reside behind the second firewall which filters based on IP addresses or domains.

Noelle: The following note seems out of place here. It is much more detailed and specific than anything else here. .. Maybe this is more of a packaging/deploying note. Can you beef up this section a little?

Reminder: Enterprise beans in each application server of a cluster have separate identities. Therefore, every bean must be explicitly protected by configuring its resource security in the enterprise application configuration.

5.1.2 Performance

Performance involves minimizing the response time for a given transaction load. Although a number of factors relating to application design can affect performance, one or both of the following techniques are commonly used:

- Vertical scaling

  Involves creating additional application server processes on a single physical machine, providing for software/application server failover as well as load balancing across multiple JVM (application server processes). Vertical Scaling allows an administrator to profile an existing application server for bottlenecks in performance, and potentially use additional application servers, on the same machine, to get around these performance issues.

- Horizontal scaling

  Involves creating additional application server processes on multiple physical machines to take advantage of the additional processing power available on each machine. This provides hardware failover support and allows an administrator to spread the cost of an implementation across multiple physical machines.

5.1.3 Throughput

Throughput, while related to performance, involves the creation of a number of application server instances to service the same requests. IBM WebSphere Application Server Network Deployment provides clusters, which logically group a number of application servers. The application servers are added through vertical and/or horizontal scaling.

Note: There is no longer the concept of server groups and clones, as in WebSphere Application Server V4.0. Now each application server cooperates in a *cluster,* as a *cluster member*.

5.1.4 Scalability

Multiple machines can be configured to add processing power, improve security, maximize availability, and balance workloads. The components that provide functions for configuring scalability include:

- WebSphere application server clusters

  Allows the creation of a logical group of servers that all host and run the same application(s). Members of a cluster can be located on the same machine (vertical scaling) and/or across multiple machines (horizontal scaling).

The use of application server clusters can improve the performance of a server, simplify its administration, and enable the use of workload management; however, an administrator will need to assess whether the cost of implementing a clustered configuration (network traffic, performance), outweighs the benefits of a clustered environment. For more details see *IBM WebSphere V5.0 Applications: Ensuring High Performance and Scalability, SG24-6198.*

- Workload management (WLM)

  Incoming processing requests from clients are transparently distributed among the clustered application servers. WLM enables both load balancing and failover, improving the reliability and scalability of WebSphere applications.

Note: IBM WebSphere Application Server Network Deployment provides three types of workload management (WLM):

- Web server WLM

  Uses the Network Dispatcher feature from Edge Components to distribute HTTP requests across Web servers.
- Plug-in WLM

  Distributes servlet requests across Web containers.
- EJS WLM

  Distributes EJB requests across EJB containers.

- IP sprayer

  Transparently redirects incoming HTTP requests from Web clients to a group of Web servers. Although the clients behave as if they are communicating directly with a given Web server, the IP sprayer is actually intercepting all requests and distributing them among all the available Web servers in the group. IP sprayers (such as WebSphere Application Server Edge or Cisco Local Director) can provide scalability, load balancing, and failover for Web servers.

Author Comment: Web server WLM (see gray box) is the same as IP sprayer above .. no?

5.1.5 Availability

To avoid a single point of failure and maximize system availability, the topology must have some degree of process redundancy. High availability topologies typically involve horizontal scaling across multiple machines. Vertical scaling can improve availability by creating multiple processes, but the machine itself becomes a point of failure.

In addition, an IP sprayer can direct client HTTP requests to available Web servers, bypassing any that are offline. Another server can back up the IP sprayer, which eliminates it as a single point of failure.

Improved availability is one of the key benefits of scaling up to multiple machines. IBM WebSphere Application Server Network Deployment provides tools that let you manage availability by distributing critical functionality across multiple machines. Applications hosted on multiple machines generally have less down time and are able to service client requests more consistently.

The following commonly used techniques can be combined to take advantage of the best features of each and create a highly available system.

Hardware and process redundancy

Eliminate the single points of failure in a system by including hardware and process redundancy:

- Use horizontal scaling to distribute application servers across multiple physical machines. If a hardware or process failure occurs, cluster application servers are still available to handle client requests. Web servers and IP sprayers can also benefit from horizontal scaling.
- Use backup servers for databases, Web servers, IP sprayers, and other important resources. This ensures that they remain available if a hardware or process failure occurs.
- Deploy an application in multiple cells. If an entire cell goes offline, the other cells are still available to handle client requests.

Process isolation

Provide process isolation so that failing servers do not negatively impact the remaining healthy servers of a configuration. The following configurations provide some degree of process isolation:

- Deploy the Web server onto a different machine from the application servers. This ensures that problems with the application servers do not affect the Web server, and vice versa.
- Use horizontal scaling, which physically separates application server processes onto different machines.

- Deploy an application in multiple cells. Problems are confined to one cell while the other cells remain available.

Load balancing

Use load-balancing techniques to make sure that individual servers are not overwhelmed with client requests. These techniques include:

- Use of an IP sprayer to distribute requests to the Web servers in the configuration.
- Directing requests from high-traffic URLs to more powerful servers.

Author Comment: mention the edge component, feature, that does this to each line here.

Failover support

The application must continue to process client requests when servers are stopped or restarted. Ways to provide failover support include:

- Use horizontal scaling with workload management to take advantage of its failover support.
- Use the HTTP transport to distribute client requests among application servers.

Hardware-based high availability

In most cases there is very little to be gained by configuring IBM WebSphere Application Server Network Deployment to work in conjunction with a hardware-based high availability (HA) product, e.g.. HACMP, Sun Cluster or MC/ServiceGuard.

The only case where a hardware-based HA solution would provide value is where WebSphere serves as the coordinator of a distributed (two-phase commit) transaction. In all other cases, the cluster capabilities inherent to IBM WebSphere Application Server Network Deployment should be sufficient to provide high availability of the WebSphere runtime.

5.1.6 Maintainability

The topology affects the ease with which system hardware and software can be updated. For instance, using multiple WebSphere cells or horizontal scaling can make a system easier to maintain because individual machines can be taken offline without interrupting other machines running the application.

Sometimes maintainability conflicts with other topology considerations. For example, limiting the number of application server instances makes the application easier to maintain but can have a negative effect on throughput, availability, and performance.

When deciding how much vertical and/or horizontal scaling is to be used in a topology, consideration needs to be made for hardware upgrades, e.g.. adding or upgrading CPUs and memory.

Tip: An alternative topology for maintainability involves the use of multiple cells (administrative domains). .. point to a reference here ..

5.1.7 Session management

Unless only a single application server is used, or the application is completely stateless, maintaining session state between HTTP client requests will also be a factor in determining the chosen topology.

IBM WebSphere Application Server Network Deployment provides two different functions for the sharing of sessions between multiple application server processes:

- Database persistence

  Session data is persisted to a database shared by the application servers.

- Memory-to-memory replication

  Provides replication of session data between the memories of different application server JVMs. A JMS publish/subscribe mechanism, called WebSphere Internal Messaging, is used to provide assured session replication between the JVM processes, by leveraging the internal JMS provider provided with WebSphere Application Server.

  For additional information see Chapter 17, “Configuring session management” on page 551..

  Note: This mechanism is new to IBM WebSphere Application Server Network Deployment 5.0. Prior versions of WebSphere only provided persistence of the session to a database.

The choice of whether to use memory-to-memory session replication or database persistence for sessions can be made by weighing up the following advantages and disadvantages of the memory-to-memory approach:

- Advantages compared to database persistence

   a. A separate database product is not required.

   b. Memory-to-memory replication is faster, by virtue of the high performance messaging implementation used by WebSphere 5.0.

- Disadvantages compared to database persistence

   a. Only works in the Network Deployment (and above) environment. The functionality is not provided with the base WebSphere 5.0 product, since it depends upon a shared internal JMS server messaging infrastructure.

      For example, if an application’s session must be persisted to a database in a base WebSphere 5.0 environment (standalone application server), then database persistence must be used.

In addition, the configuration of an IP sprayer such as the IBM Network Dispatcher, included as part of the WebSphere Application Server Edge, needs to be considered when session state is a factor.

5.1.8 Topology selection criteria summary

The preceding factors are not mutually exclusive. They can be combined in many different ways, as shown by the examples in “Topologies” on page 84.

Table 5-1 provides a summary of the topology selection criteria available with IBM WebSphere Application Server Network Deployment.

Table 5-1 Topology selection summary

Topology	**Security**	**Performance**	**Throughput**	**Maintainability**	**Availability**	**Session**
Vertical scaling		Limited benefit	Limited to resources on a single machine	Easiest to maintain	Process isolation	Required
Horizontal scaling		Best in general	Best in general	Code migration to multiple nodes	Process and hardware redundancy	Required
HTTP separate	Allow for firewalls and DMZs	Usually better than local	Usually better than local			
Three tiers	Most options for firewalls	Typically slower than single JVM API	Clustering can improve throughput			

Topology	Security	Performance	Throughput	Maintainability	Availability	Session
One cell				Ease of maintenance		
Multiple cells				Harder to maintain than single cell	Process, hardware and software redundancy	

5.2 WebSphere component coexistence

IBM WebSphere Application Server Network Deployment supports a number of scenarios for the coexistence of IBM WebSphere Application Server installations on a single machine. This support provides new options to consider when selecting a topology:

- Multiple IBM WebSphere Application Server instances.
- Multiple server instances using a single installation.
- Coexistence of Base and Network Deployment Manager.
- Single vs. multiple Web servers.

5.2.1 Multiple WebSphere instances

IBM WebSphere Application Server supports two types of runtime instances on a single machine:

- **Application server**: Multiple instances from a single installation of WebSphere Application Server.
- **Deployment Manager**: Multiple instances from a single installation of WebSphere Application Server Network Deployment.

Note: As indicated in Table 5-2, multiple *installations* of specific software on a single machine is currently not supported. However, a single installation can be configured to run multiple runtime instances on a single machine.

Table 5-2 Multiple installation scenarios

Installed product	**with WAS 5.0 base**	**with WAS 5.0 ND**
WebSphere 5.0 Base	Currently this is not supported	Supported There are no issues. There are no port conflicts in the default configuration settings for both products. The WebSphere development community commonly uses this environment.
WebSphere 5.0 ND	Supported	Currently this is not supported

Author Comment: make sure it is clear that WAS ND means only the deployment manager. I think ND as a product will be packaged with the base server as well and this could get confusing.

5.2.2 Multiple server instances using a single installation

The following configurations are supported for running multiple runtime instances using a single installation:

- Creating and running multiple IBM WebSphere Application Server instances from a single WebSphere Application

  This installation requires the manual creation of copies of the configuration folder to be placed on each server.

- Creating and running multiple network deployment manager instances from a single WebSphere Application Server Network Deployment installation.

Although multiple deployment managers can be configured for a single WAS 5.0 ND installation, these deployment managers cannot provide failover or clustering support for each other. In essence this configuration allows the deployment managers of multiple (and unconnected) cells to be configured and run on a single machine.

Author Comment: Doesn’t this cause a problem with ports? Need to mention how to deal with this. Is this in the installation chapter?

5.2.3 Coexistence of Base and Network Deployment

IBM WebSphere Application Server supports the installation of Network Deployment software (Deployment Manager) on the same machine as a base IBM WebSphere Application Server installation.

This option allows the Deployment Manager and a node to both be concurrently installed and run on a single machine.

Advantages

- No need for a dedicated machine to host the cell Deployment Manager and its master cell repository.
- Ability to reuse existing backup facilities provided on the node machine.

Disadvantages

- Need to manually move the Deployment Manager and master configuration repository to another machine if node machine must be rebuilt.

  Note: IBM WebSphere Application Server Network Deployment does not provide any system management tools for automating this task.

- Possible contention for machine resources (if the node’s application servers are heavily used) may affect the performance of cell system management functions.
- Less flexibility in tailoring server machines to address specific tasks.

5.2.4 Single vs. multiple Web servers

In addition to multiple instances, WebSphere Application Server also provides Web server options when there are coexisting application servers on a single machine:

- Single Web server for coexisting, multi-version application servers on one machine.
- Single Web server for multiple instances of WebSphere 5.0.
- Separate Web servers for each application server instance, when running multiple instances of WebSphere 5.0.

Author Comment: Doesn’t this cause a problem with port 80? Need to mention how to deal with this. Is this in the installation chapter?

5.3 Topologies

IBM WebSphere Application Server Network Deployment supports a number of common topologies:

- Vertical scaling
- HTTP server separation
- Reverse proxy
- Multi-tiered
- Horizontal scaling with clusters
- Horizontal scaling with IP sprayer
- Single vs. multiple WebSphere cells
- Multiple applications on a node
- Combined

Note: For each of the common topologies, a decision needs to be made regarding the placement of the Deployment Manager and master cell repository. The Deployment Manager can be located either on a dedicated machine, or on the same machine as one of its nodes. The advantages and disadvantages of each approach are outlined in Section 5.2.3, “Coexistence of Base and Network Deployment” on page 83.

5.3.1 Vertical scaling

Vertical scaling refers to configuring multiple application servers on a single machine, usually by creating a cluster of associated application servers all hosting the same application(s). This configuration is depicted in Figure 5-1.

Author Comment: Should consider adding the persistent db to the pic.

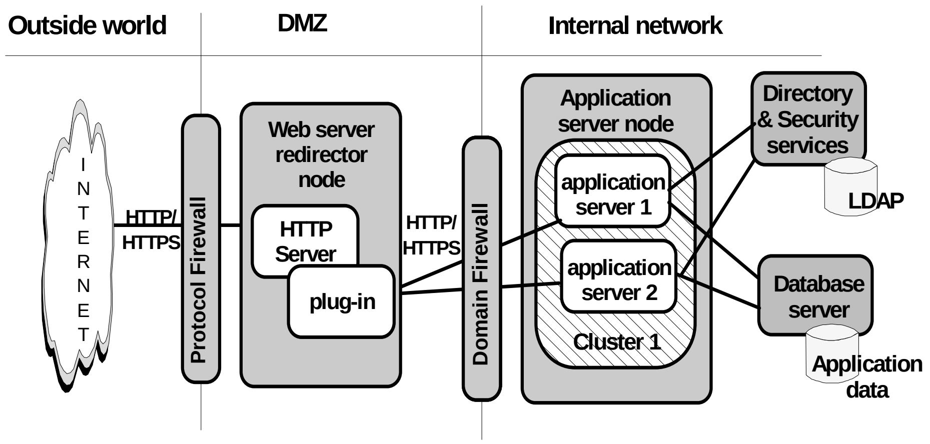


Figure 5-1 Vertical scaling with clusters

The figure represents a simple vertical scaling example, with multiple cluster members on the application server node. Vertical scaling can also be implemented on more than one machine in a configuration. Vertical scaling can be combined with other topologies to boost performance and throughput.

Advantages

Vertical scaling has the following advantages:

- More efficient use of machine processing power

  An instance of an application server runs in a single Java Virtual Machine (JVM) process. However, the inherent concurrency limitations of a JVM process prevent it from fully utilizing the processing power of a machine. Creating additional JVM processes provides multiple thread pools, each corresponding to the JVM associated with each application server process. This avoids concurrency limitations and lets the application server use the full processing power of the machine.

- Load balancing

  Vertical scaling topologies can make use of the WebSphere workload management facility.

Author Comment: Need a little more info here about what we can use in the way of WLM since this isn't covered in this book.

- Process failover

  Vertical scaling can provide failover support among application servers of a cluster. If one application server instance goes offline, the other instances on the machine continue to process client requests.

Disadvantages

Single machine vertical scaling topologies have the drawback of introducing the host machine as a single point of failure in the system. Vertical scaling on multiple machines avoids the single point of failure.

5.3.2 HTTP server separation

HTTP server separation topologies physically separate the HTTP (Web) server from the application servers, placing the Web server on a different machine in the configuration.

Note: This topology is often used with firewalls to create a secure demilitarized zone (DMZ) surrounding the Web server.

WebSphere Application Server can use a HTTP plug-in to route requests from the Web server to application servers on remote machines. This configuration is depicted in Figure 5-2.

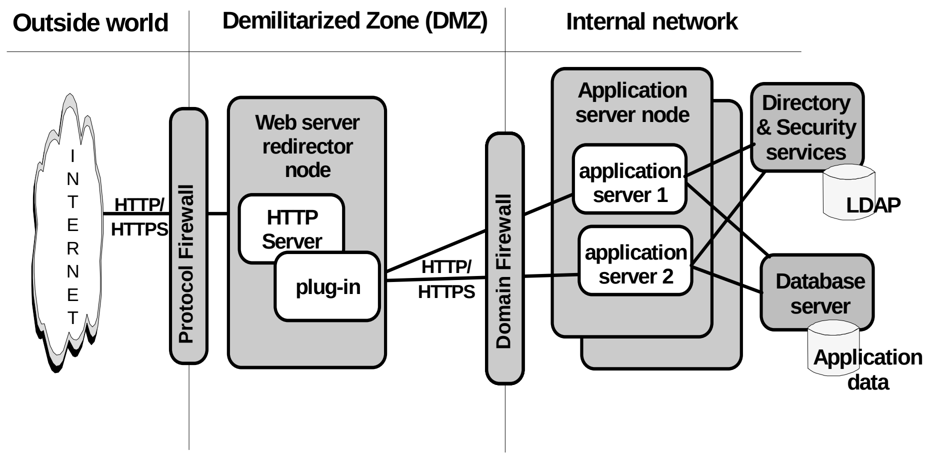


Figure 5-2 HTTP transport remote web server topology

The Web server redirector node hosts the Web server and receives HTTP requests from clients. An HTTP plug-in forwards the requests to the application server nodes by using the HTTP or HTTPS protocol. The business data is hosted on the database server node.

Variations on this configuration include vertical scaling of the application servers. Additional application server nodes can be added to the configuration to implement horizontal scaling.

Advantages

HTTP server separation has the following advantages:

- Supports load balancing and failover, eliminating single points of failure.

  A point of failure exists when one process or machine depends on another process or machine. A single point of failure is undesirable because if the point fails, the whole system will become unavailable. When comparing DMZ solutions, a single point of failure refers to a single point of failure between the Web server and application server. Various failover configurations can minimize downtime and possibly even prevent a failure. However, these configurations usually require additional hardware and administrative resources.

- Avoids data access from DMZ.

  A DMZ configuration protects application logic and data by creating a demilitarized zone between the public Web site and the servers and databases where this valuable information is stored. Desirable DMZ topologies do not have servers that directly access databases from the DMZ.

- Supports WebSphere security in that the location of the Web server is not relevant to the security services provided by WebSphere.

  WebSphere security protects applications and their components by enforcing authorization and authentication policies. Configuration options compatible with WebSphere security are desirable because they do not necessitate alternative security solutions.

- Supports Network Address Translation (NAT) firewalls.

  A firewall product that runs NAT receives packets for one IP address, and translates the headers of the packet to send the packet to a second IP address. In environments with firewalls employing NAT, avoid configurations involving complex protocols in which IP addresses are embedded in the body of the IP packet, such as Java Remote Method Invocation (RMI) or Internet Inter-Orb Protocol (IIOP). These IP addresses are not translated, making the packet useless.

- Supports Secure Sockets Layer (SSL) encryption for communications between the Web server and the application server.

Configurations that support encryption of communication between the Web server and application server reduce the risk that attackers will be able to obtain secure information by "sniffing" packets sent between the Web server and application server. A performance penalty usually accompanies such encryption.

- Performance bottlenecks may be reduced.
- Administration is simplified. The HTTP plug-in uses a single, easy-to-read XML configuration file.

  Some solutions require little or no maintenance after you establish them, while others require periodic administrative steps, such as stopping a server and starting it again after modifying resources that affect the configuration.

Disadvantages

HTTP server separation has the following disadvantages:

- It requires manual configuration and administration of the HTTP server.
- There is no protocol shift for inbound and outbound traffic across a firewall. The Web server sends HTTP requests to application servers behind firewalls and an HTTP port in the firewall must be open to let the requests through.

  Configurations that require switching to another protocol (such as IIOP), and opening firewall ports corresponding to the protocol, are often more complex to set up, and the protocol switching overhead can impact performance. The following approaches can be used to address this :

  – Configure the HTTP plug-in to use HTTPS. This will provide a high security connection between the HTTP server and the application server.

    This connection can be configured so that the HTTP 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 5-4 contains a summary of HTTP server separation topology characteristics:

Table 5-3 Characteristics of HTTP server separation topology

Characteristic	**Comment**
SSL support	Yes
Workload management (WLM)	Yes
Network address translation (NAT)	Yes

Characteristic	Comment
Performance	High
Administration of configuration	Manual
Avoids data access from DMZ	Yes
Avoids DMZ protocol switch	No
Avoids single point of failure	Yes
Compatible with WebSphere security	Yes
Minimum required number of firewalls holes	1 per application server, plus 1 if WebSphere 5.0 security is used by the web server.

5.3.3 Reverse proxy

Reverse proxy (or IP-forwarding) topologies use a reverse proxy server, such as the one in WebSphere Application Server Edge, to receive incoming HTTP requests and forward them to a Web server. The Web server in turn forwards the requests to the application servers that do the actual processing. The reverse proxy returns requests to the client, hiding the originating Web server. This configuration is depicted in Figure 5-3.

Note:

- The reverse proxy approach for separating the HTTP server and the application server is generally not suggested for use with WebSphere 5.0. The HTTP plug-in behaves much like a reverse HTTP proxy, without its disadvantages. See Section 5.3.3, “Reverse proxy” on page 89 for details.
- Use of a reverse proxy has benefits if placed before the HTTP server, as shown in Figure 5-3.

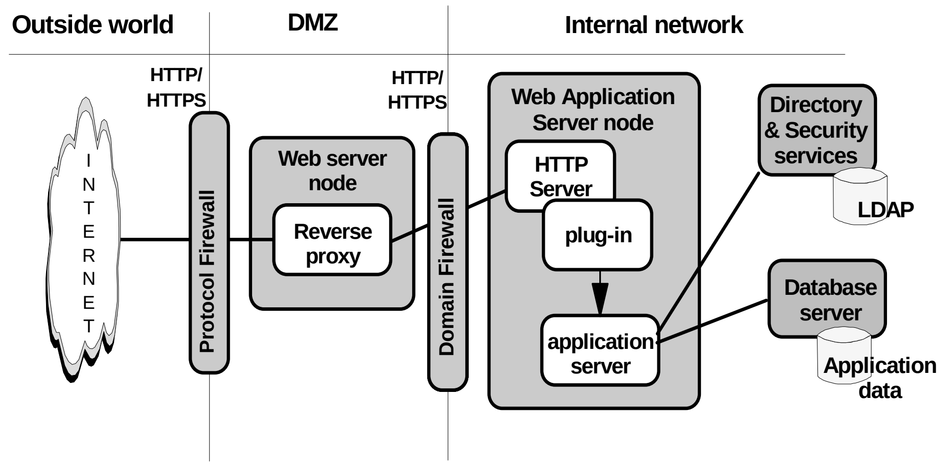


Figure 5-3 Reverse proxy Web server topology

In this example, a reverse proxy resides in a demilitarized zone (DMZ) between the outer and inner firewalls. It listens on an HTTP port (typically port 80) for HTTP requests. The reverse proxy then forwards those requests to an HTTP server that resides on the same machine as application server. After the requests are fulfilled, they are returned through the reverse proxy to the client, hiding the originating Web server.

Reverse proxy servers are typically used in DMZ configurations to provide additional security between the public Internet and the Web servers (and application servers) servicing requests. A reverse proxy product used with WebSphere Application Server must support Network Address Translation (NAT) and WebSphere security.

Reverse proxy configurations support high-performance DMZ solutions that require as few open ports in the firewall as possible. The reverse proxy capabilities of the Web server inside the DMZ require as few as one open port in the second firewall (potentially two if using SSL - port 443).

Advantages

Advantages of using a reverse proxy server in a DMZ configuration include:

- Supports WebSphere security and NAT firewalls.
- It is a well-known and tested configuration, resulting in less customer confusion than other DMZ configurations.
- Reliable and its performance is relatively fast.

- Eliminates protocol switching by using the HTTP protocol for all forwarded requests.
- Does not affect the configuration and maintenance of a WebSphere application.
- Uses only one firewall port (HTTP) for requests and responses.

Disadvantages

Disadvantages of using a reverse proxy server in a DMZ configuration include:

- The presence of a reverse proxy server in a DMZ might not be suitable for some environments.
- Requires more hardware and software than similar topologies that do not include a reverse proxy server, making it more complicated to configure and maintain.
- Does not participate in WebSphere workload management.
- Cannot be used in environments where security policies prohibit the same port or protocol being used for inbound and outbound traffic across a firewall.

Table 5-4 contains a summary of reverse proxy topology characteristics:

Table 5-4 Characteristics of reverse proxy topology

Characteristic	**Comment**
SSL support	Yes
Workload management (WLM)	No
Network address translation (NAT)	Yes
Performance	High
Administration of configuration	Manual
Avoids data access from DMZ	Yes
Avoids DMZ protocol switch	Yes
Avoids single point of failure	No
Compatible with WebSphere security	Depends on Web server
Minimum required number of firewall holes	1

5.3.4 Multi-tiered

Partitioning the application server processes into servlet application servers and EJB application servers as depicted in Figure 5-4, can provide some advantages from a security perspective as well as some potential advantages from a performance perspective.

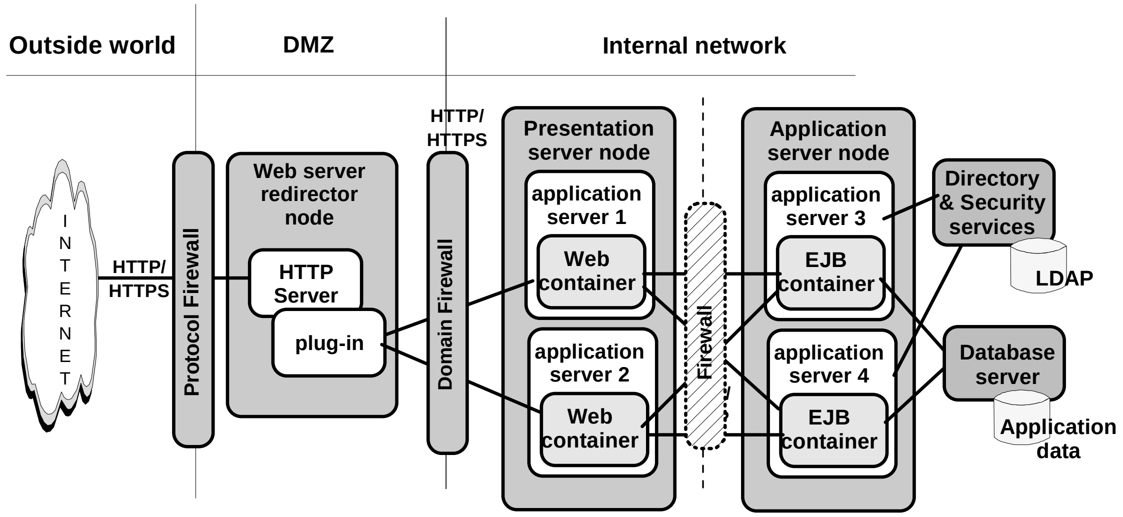


Figure 5-4 Multi-tiered topology

In the example in Figure 5-4, application server processes that run servlets reside on a presentation server node. This puts them closer (in a network sense) to the HTTP server, improving their response to client requests.

Application server processes that run enterprise beans reside on the application server node. This puts them in closer proximity to the application data, which is represented in an application by entity beans and stored on the database server.

Clustering the application servers help maximize resource use on each node. The example shows two cluster members on each node, but as many can be used as can be supported by each machine’s resources.

Author Comment: are there two clusters or one?

Advantages

Multi-tiered topologies have the following advantages:

- A cluster in a multi-tiered topology provides process redundancy and uses memory more efficiently than in similar topologies that host only single instances of application servers.
- Allows flexibility in the replication of application servers, for example, you could have two Web containers, but five EJB containers.
- Additional resources on the machines can improve application throughput and performance.
- If a firewall is introduced between each pair of tiers, an additional layer of security can be provided for entity beans and application data.

Disadvantages

Multi-tiered topologies have the following disadvantages:

- Eliminates local Java Virtual Machine (JVM) optimizations that occur when both the Web containers and EJB containers run in the same application server. It also introduces network latency. These factors tend to slow system performance.
- The level of redundancy can make maintenance more complicated.

5.3.5 Horizontal scaling with clusters

Horizontal scaling exists when the members of an application server cluster are located across multiple physical machines. This lets a single WebSphere application span several machines, yet still present a single logical image. This configuration is depicted in Figure 5-5.

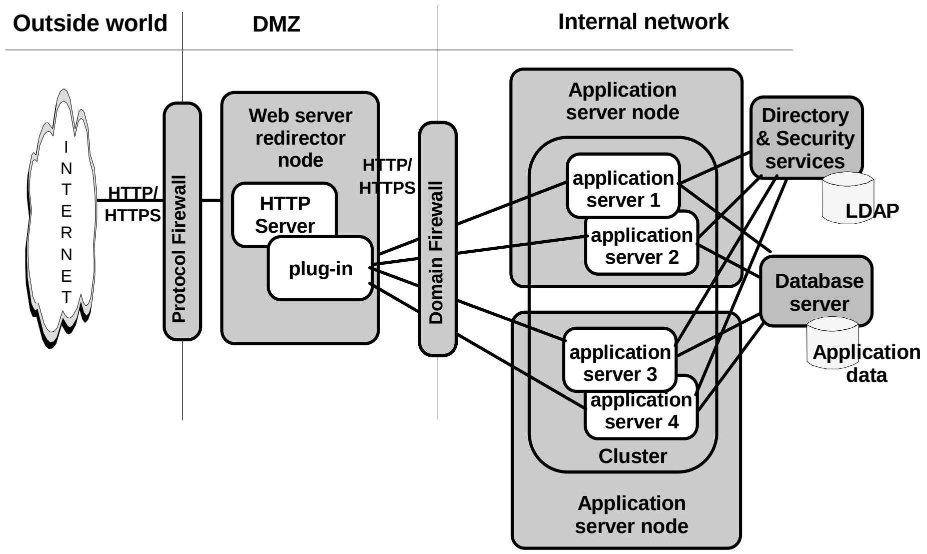


Figure 5-5 Horizontal scaling with clusters topology

The HTTP server distributes requests to cluster member application servers on the application server nodes.

The Network Dispatcher component of WebSphere Application Server Edge, which can distribute client HTTP requests, can be combined with clustering to reap the benefits of both types of horizontal scaling. See Section 5.3.6, “Horizontal scaling with IP sprayer” on page 95 for a description of this configuration.

Advantages

Horizontal scaling using clusters has the following advantages:

- Provides the increased throughput of vertical scaling topologies but also provides failover support. This topology allows handling of application server process failure and hardware failure without significant interruption to client service.
- Optimizes the distribution of client requests through mechanisms such as workload management or remote HTTP transport.

Disadvantages

Horizontal scaling using clusters has the following disadvantage:

- Increased maintenance effort.

5.3.6 Horizontal scaling with IP sprayer

Load-balancing products such as Network Dispatcher can be used to distribute HTTP requests among application server instances that are running on multiple physical machines.

The Network Dispatcher component of WebSphere Application Server Edge is an IP sprayer that performs intelligent load balancing among Web servers using server availability, capability, workload, and other criteria.

Author Comment: If clustering is a factor here we should discuss and show it.

Simple IP sprayer topology

Figure 5-6 depicts a simple horizontal scaling configuration that uses an IP sprayer on the load balancer node to distribute requests among application servers on multiple machines:

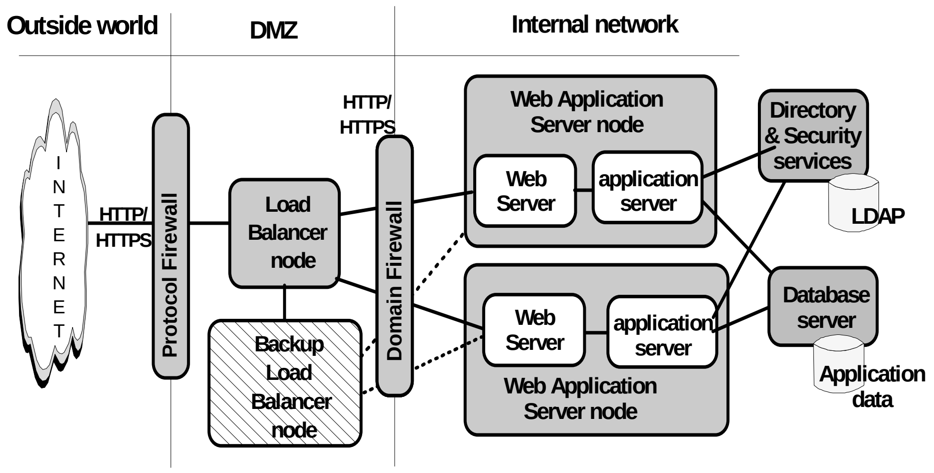


Figure 5-6 Simple IP sprayer-based horizontally scaled topology

A backup node for the load balancer node is normally configured in order to eliminate it as a single point of failure.

Note: In an IP sprayer (or similarly configured) topology, each Web server can be configured to perform workload management over all application servers in a specific cluster. In this configuration session affinity will be preserved, not matter which web server is passed the request by the IP sprayer.

Complex IP sprayer topology

Figure 5-7 depicts a topology where an IP sprayer distributes requests among several machines containing web servers and clustered application servers.

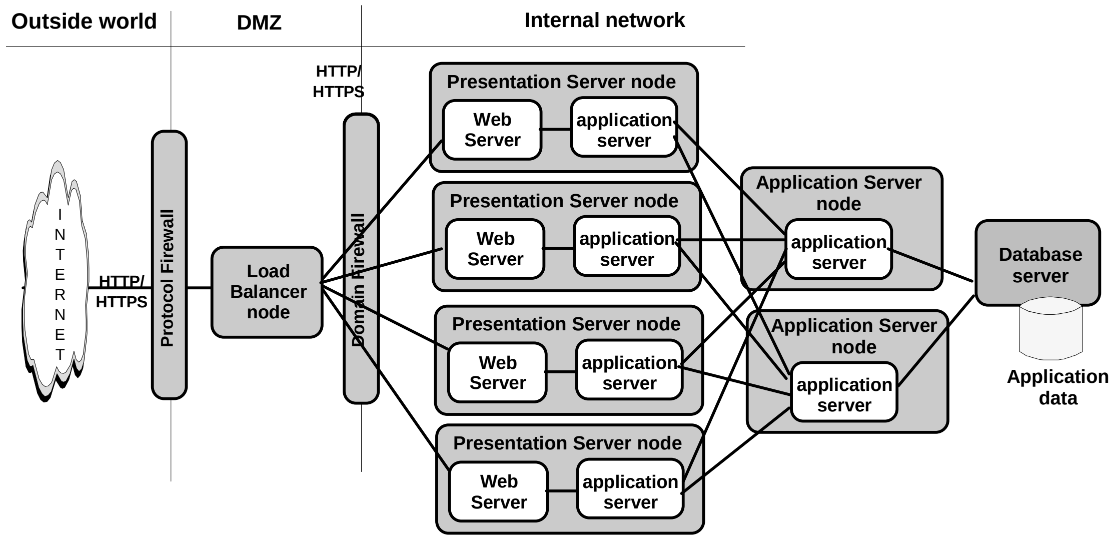


Figure 5-7 Complex IP sprayer horizontally scaled topology

The presentation server nodes host servlet-based applications. The application server nodes contain enterprise beans that access application data and execute business logic. This allows numerous less powerful machines to be employed in the first tier and fewer but more powerful machines in the second tier.

Advantages

Using an IP sprayer to distribute HTTP requests has the following advantages:

- Improved server performance by distributing incoming TCP/IP requests (in this case, HTTP requests) among a group of Web server machines.
- The use of multiple Web servers increases the number of connected users.
- Elimination of the Web server as a single point of failure. It can also be used in combination with WebSphere workload management to eliminate the application server as a single point of failure.
- Improved throughput by letting multiple servers and CPUs handle the client workload.

Disadvantages

Using an IP sprayer to distribute HTTP requests has the following disadvantage:

- Extra hardware and software is required for the IP sprayer servers.

5.3.7 Multiple WebSphere cells

Figure 5-8 depicts how an application can be implemented over multiple WebSphere Application Server Network Deployment cells:

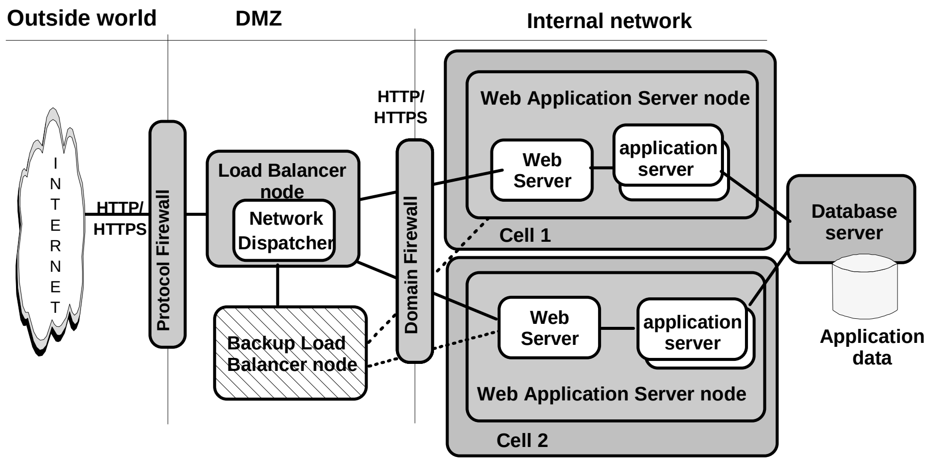


Figure 5-8 Multiple WebSphere cell topology

The example application runs simultaneously in two cells, each hosted on a different physical machines. Network Dispatcher is used as an IP sprayer to distribute incoming HTTP requests among the two cells, presenting a single image of the application to clients.

Each cell is administered independently as each cell has its own set of XML configuration files.

Tip: A different version of the application can be run in each cell. In addition, because the cells are isolated from one another, different versions of the WebSphere Application Server software can be used in each cell.

There are no hard limits on the number of nodes and application servers that can be used in a single cell. However, the choice of whether to use a single or multiple cells can be made by weighing the advantages and disadvantages.

Advantages

Using multiple cells has the following advantages:

- Isolation of hardware failure

  If one cell goes offline due to hardware problems, the others can still process client requests.

- Isolation of software failure

  Running an application in two or more cells isolates any problems that occur within a cell, while the other cells continue to handle client requests. This can be helpful in a variety of situations:

  – When rolling out a new application or a revision of an existing application. The new application or revision can be brought online in one cell and tested in a live situation while the other cells continue to handle client requests with the production version of the application.

  – When deploying a new version of the IBM WebSphere Application Server software. The new version can be brought into production and tested in a live situation without interrupting service.

  – When applying fixes or patches to the IBM WebSphere Application Server software. Each cell can be taken offline and upgraded without interrupting the application.

- If an unforeseen problem occurs with new software, using multiple cells can prevent an outage to an entire site. A rollback to a previous software version can also be accomplished more quickly. Hardware and software upgrades can be handled on a cell-by-cell basis during off-peak hours. Improved performance

  Running an application using multiple smaller cells may provide better performance than a single large cell because there is less inter-process communication in a smaller cell.

Disadvantages

Using multiple cells has the following disadvantages:

- Deployment is more complicated than for a single administrative cell. Using a distributed file system that provides a common file mount point can make this task easier.
- Multiple cells require more administration effort because each cell is administered independently. This problem can be reduced by using scripts to standardize and automate common administrative tasks.

5.3.8 Multiple clusters on a node

When deciding how to deploy an application, one decision to be made is whether to deploy a cluster of application servers across all machines, as depicted in Figure 5-9, or only on a single machine, as depicted in Figure 5-10.

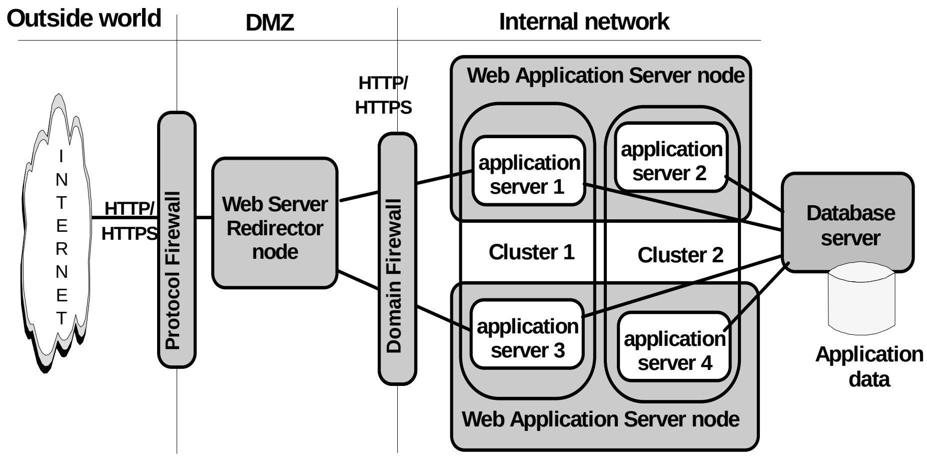


Figure 5-9 Multiple application server clusters on each machine

The topology in Figure 5-9 uses horizontal scaling. Each cluster of application servers is distributed throughout all of the machines in the system. In this example, a member of each cluster is hosted on each Web application server node. The Web server redirector node distributes client requests to the application servers on each node.

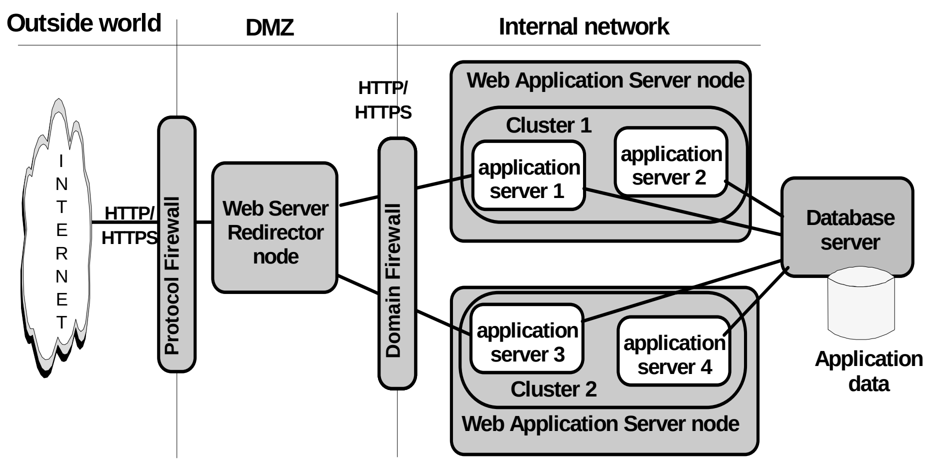


Figure 5-10 Single application server cluster on each machine

Jeff: check this figure. There was no figure here before so I am guessing at what it should look like. Are the applications the same on app server 1/2 and a different app on server 3/4?

The topology in Figure 5-10 uses vertical scaling. Each cluster of application servers is located on a single machine of the system.

Advantages

Hosting members of multiple application server clusters across one or more machines has the following advantages:

- Improved throughput

  Using an application server cluster enables the handling of more client requests simultaneously.

- Improved performance

  Hosting cluster members on multiple machines enables each member to make use of the machine's processing resources.

- Hardware failover.

  Hosting cluster members on multiple nodes isolates hardware failures and provides failover support. Client requests can be redirected to the application server members on other nodes if a node goes offline.

- Application software failover

  Hosting cluster members on multiple nodes also isolates application software failures and provides failover support if an application server stops running. Client requests can be redirected to cluster members on other nodes.

- Process isolation

  If one application server process fails, the cluster members on the other nodes are unaffected.

Disadvantages

Hosting members of multiple application server clusters across one or more machines has the following disadvantage:

- More complex maintenance

  Application servers must be maintained on multiple machines.

5.3.9 Combined topology

An example of a topology that combines the best elements of the other topologies discussed in this chapter is depicted in Figure 5-11:

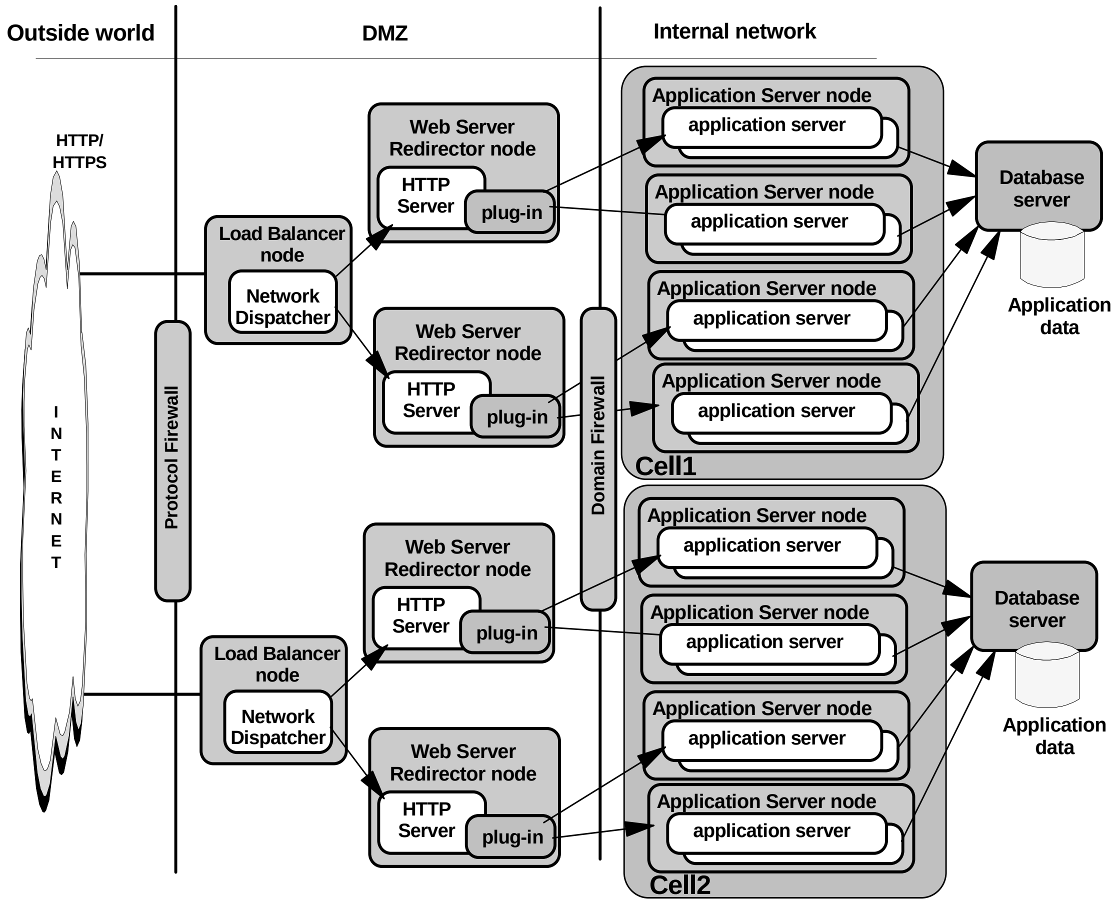


Figure 5-11 Combined topology

This topology combines elements of several different basic topologies:

- Two WebSphere cells.
- Two load balancer nodes.
- Two HTTP servers for each cell with the HTTP plug-in.
- Four application server machines for each cell .
- The use of application server clusters for both vertical and horizontal scaling. In the example topology, each machine hosts two cluster members; in practice, the number of cluster members is limited by the computing resources of each node.

- Two database servers for each cell. These servers host mirrored copies of the application database.

Advantages

This topology is designed to maximize throughput, availability, and performance. It incorporates the best practices of the other topologies discussed in this chapter:

- Having more than one load balancer, HTTP server, application server, and database server in each cell eliminates single points of failure.
- Multiple cells provide both hardware and software failure isolation, especially when upgrades of the application or the application server software are rolled out. (Hardware and software upgrades can be handled on a cell-by-cell basis during off-peak hours).
- Horizontal scaling is done by using both application server clusters and the IP sprayer to maximize availability and eliminate single points of process and hardware failure.
- Application performance is improved by using several techniques:
  - Hosting application servers on multiple physical machines to boost the available processing power.
  - Creating multiple smaller cells instead of a single large cell. There is less interprocess communication in a smaller cell, which allows more resources to be devoted to processing client requests.
  - Using clusters to vertically scale application servers on each node, which makes more efficient use of the resources of each machine.
- Applications with this topology can make use of several workload management techniques. In this example, workload management can be performed through one or more of the following:
  - Using the IBM WebSphere Application Server Network Deployment workload management (WLM) facility to distribute work among clustered application servers.
  - Using the load balancer to distribute client HTTP requests to each HTTP server.

  For example, an application can manage workloads at the HTTP server level with the load balancer and at the application server level with WebSphere workload management. Using multiple workload management techniques in an application provides finer control of load balancing.

  Regardless of which workload management techniques are used in the application, Network Deployment participates in workload management to provide failover support.

- In this topology, users notice an interruption only when an entire cell is lost. If this occurs, the active HTTP sessions are lost for half of the clients. The system can still process HTTP requests although its performance is degraded.

Disadvantages

The combined topology has the following disadvantage:

- Multiple cells require more administration effort, because each cell is administered independently. This problem can be reduced by using scripting to standardize and automate common administrative tasks.

5.4 Closing thoughts on topologies

Whatever topology is decided upon, a best practice is to partition the production acceptance environment exactly the same as the production environment. This avoids surprises when deploying an application into production.

Another consideration, when practical for an application architecture, is to create a number of smaller application servers, rather than a single large server. This has 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 needs to 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.

5.5 For more information

- *IBM WebSphere V5.0 Applications: Ensuring High Performance and Scalability, SG24-6198.*

6

Naming

In this chapter we describe the concepts behind the naming functionality provided as part of IBM WebSphere Application Server. We will cover:

- New features in IBM WebSphere Application Server
- WebSphere naming architecture
- Interoperable Naming Service (INS)
- Distributed CosNaming
- Configured bindings
- Initial contexts
- Federation of namespaces
- Interoperability
- Examples
- Naming tools

6.1 New features

Author Comment: name space or namespace ?? that is the question ..

The following are new features of a WebSphere 5.0 namespace:

- Namespace is distributed

  For additional scalability, the namespace for a cell is distributed among the various servers. The Deployment Manager, node agent and application server processes all host a name server. In previous releases, there was only a single name server for an entire administrative domain.

  In WebSphere releases prior to 5.0, all servers shared the same default initial context, and everything was bound relative to that initial context. In WebSphere 5.0, the default initial context for a server is its server root. System artifacts, such as EJB homes and resources, are bound to the server root of the server they are associated with.

- Transient and persistent partitions

  The name space is partitioned into transient areas and persistent areas. Server roots are transient. System-bound artifacts such as EJB homes and resources are bound under server roots. There is a cell persistent root, which can be used for cell-scoped persistent bindings, and a node persistent root, which can be used to bind objects with a node scope.

Author Comment: There is talk that the node persistent root will be removed from WAS 5.0... need to check this

- Federated namespace structure

  A namespace is a collection of all names bound to a particular name server. A namespace can contain naming context bindings to contexts located in other servers. If this is the case, the name space is said to be a *federated namespace* as it is a collection of namespaces from multiple servers. The namespaces link together to cooperatively form a single logical namespace.

  In a federated namespace, the real location of each context is transparent to client applications. Clients have no knowledge that multiple name servers are handling resolution requests for a particular requested object.

  The namespace for the WebSphere 5.0 cell is federated among the Deployment Manager, node agents and application servers of the cell. Each such server hosts a name server. All name servers provide the same logical view of the cell namespace, with the various server roots and persistent partitions of the namespace being interconnected by means of the single logical namespace.

- Configured bindings

  The configuration graphical interface and script interfaces can be used to configure bindings in various root contexts within the name space. These bindings are read-only and are bound by the system at server startup.

- Support for CORBA Interoperable Naming Service (INS) object URLs

  WebSphere 5.0 contains support for CORBA object URLs (corbaloc and corbaname) as JNDI provider URLs and lookup names, as required by the J2EE 1.3 specification.

6.2 WebSphere naming architecture

The WebSphere naming implementation is composed of JNDI and CosNaming. JNDI provides the client-side access to naming and presents the programming model used by application developers. CosNaming provides the server-side implementation and is where the namespace is actually stored. JNDI essentially provides a client-side wrapper of the namespace stored in CosNaming, and interacts with the CosNaming server on behalf of the client.

The model of JNDI over CosNaming has existed in several releases of WebSphere. With J2EE 1.3, limited CosNaming functionality is now required for interoperability between application servers from different vendors. This level of CosNaming, know as Interoperable Naming Service (INS) has been added for IBM WebSphere Application Server.

The following subsections provide a summary of the WebSphere naming architecture, its federated namespace and its support for JNDI.

For an explanation of the WebSphere implementations of INS and Distributed CosNaming, see Section 6.3, “Interoperable Naming Service (INS)” on page 119 and Section 6.4, “Distributed CosNaming” on page 121 respectively.

6.2.1 Components

The naming architecture is used by clients of WebSphere applications to obtain references to objects related to those applications, such as EJB homes. These objects are bound into a mostly hierarchical structure, referred to as a *name space*. In this structure, all non-leaf objects are called *contexts*. Leaf objects can be contexts and other types of objects.

The name space structure consists of a set of *name bindings*, each consisting of a name relative to a specific context and the object bound with that name. For example, the name "myApp/myEJB" consists of one non-leaf binding with the name “myApp,” which is a context. The name also includes one leaf binding with the name "myEJB," relative to "myApp." The object bound with the name "myEJB" in this example happens to be an EJB Home reference. The whole name "myApp/myEJB" is relative to the initial context, which can be viewed as a starting place when performing naming operations.

The name space can be accessed and manipulated through a *name server.* Users of a name server are referred to as naming clients. Naming clients typically use Java Naming and Directory Interface (JNDI) to perform naming operations. Naming clients can also use the Common Object Request Broker Architecture (CORBA) CosNaming interface.

Figure 6-1 summarizes the naming architecture and its components.

Author Comment: Need a figure here... may need to do from scratch. Want the deployment manager, 2 x nodes. On each node, have the node agent, appservers and JMS servers plus show the nameservice and bootstrap ports on each.

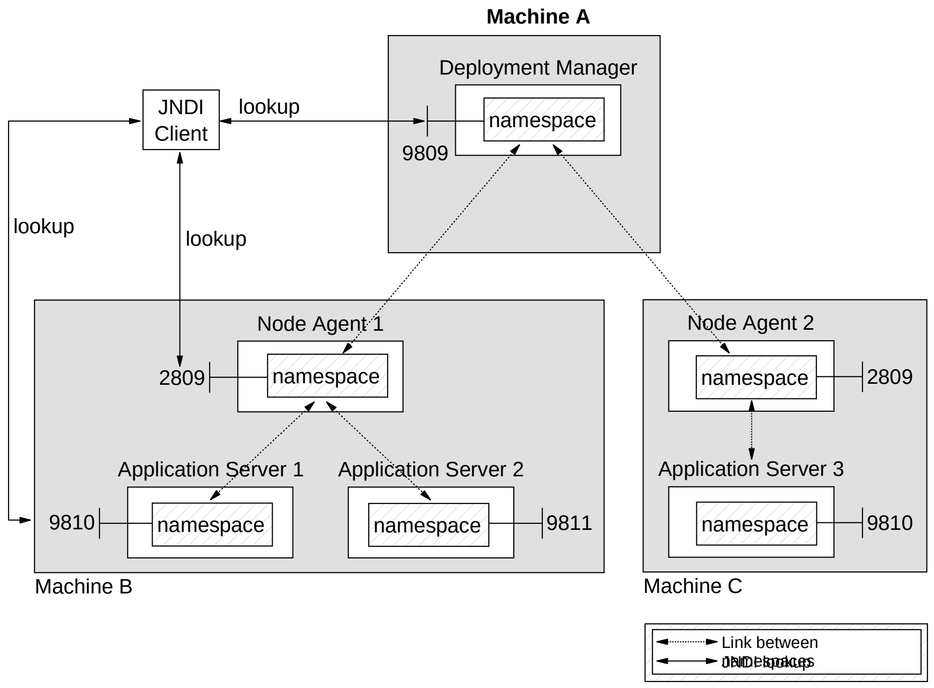


Figure 6-1 Naming topology

Notice that all WebSphere 5.0 processes (except the JMS Server) host their own Name Service and local namespace.

6.2.2 JNDI support

Each IBM WebSphere Application Server managed process (JVM) includes:

- A name server to provide shared access to its components.
- An implementation of the javax.naming JNDI package which allows users to access the WebSphere name server through the JNDI naming interface.

Note: The JNDI implementation provided by IBM WebSphere Application Server is based on version 1.2 of the JNDI interface, and was tested with version 1.2.1 of Sun's JNDI SPI (Service Provider Interface).

IBM WebSphere Application Server does not provide implementations for the following Java extension packages:

- javax.naming.directory

- javax.naming.ldap

Nor, does it support interfaces defined in the javax.naming.event package.

However, to provide access to LDAP servers, the JDK shipped with IBM WebSphere Application Server supports Sun's implementation of:

- javax.naming.ldap
- com.sun.jndi.ldap.LdapCtxFactory

6.2.3 JNDI bindings

There are three options available for binding EJB (<ejb-ref>) and resource (<resource-ref>) object names to the IBM WebSphere Application Server namespace:

- Simple name
- Corbaname
- Compound / fully qualified name

Simple name

The WebSphere 4.0 style of simple name binding is guaranteed to succeed only on a single server. It can be used in a servlet or EJB, if it is certain that the object being looked up is located on the same application server.

Example 6-1 Simple name form of JNDI binding

```
ejb/webbank/Account
```

Corbaname

The corbaname binding is always guaranteed to work. However it requires that the correct path to the object be known at deployment time.

Example 6-2 Corbaname form of JNDI binding

```
corbaname::myhost1:9812/NameServiceServerRoot#ejb/webbank/Account
```

Compound name

The fully qualified (compound name) JNDI name is always guaranteed to work.

Example 6-3 Compound name form of JNDI binding

```
cell/nodes/node1/servers/server1/ejb/webbank/Account
```

Note: The use of compound names for JNDI bindings is the recommendation for WebSphere 5.0.

6.2.4 Federated namespace

Figure 6-2 provides an overview of the federated namespace used in IBM WebSphere Application Server.

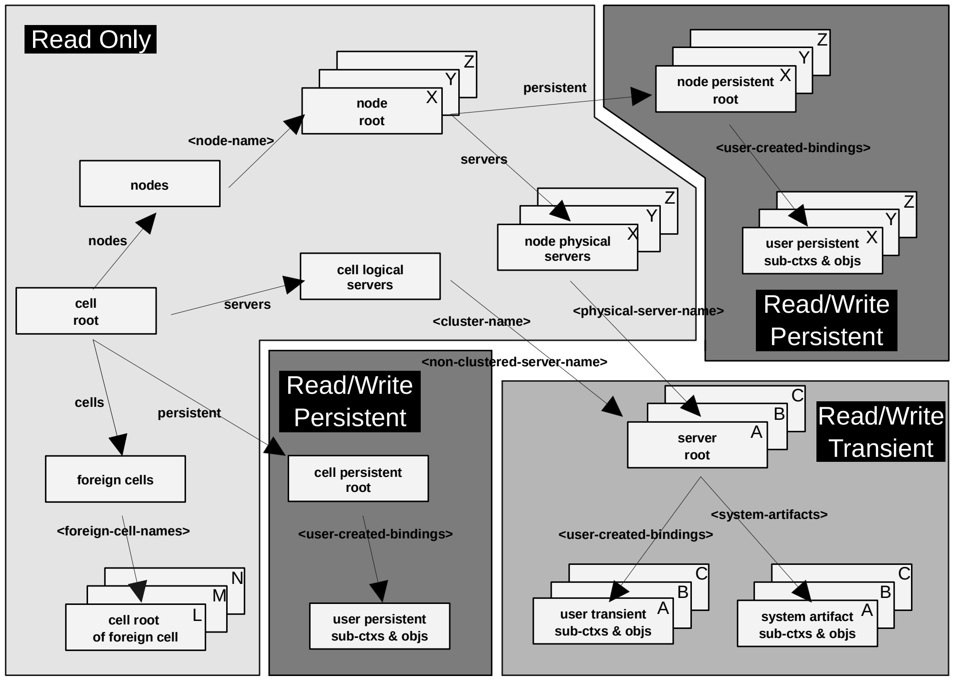


Figure 6-2 Federated namespace

Author Comment: The above figure needs to be redone for 2 reasons:
1) The node persistent root may be removed from WAS 5.0.
2) There is no longer a (top)/servers in the cell JNDI namespace. Each server is now referenced via the node that manages them.
3) Add text that shows a concrete example (i.e. shows an entry at each level)
4) the following text refers to the colors in the fig .. this is a black/white book so need a different way to distinguish.

The namespace can be broken down into a number distinct partitions that differ in whether they are updateable and/or persistent.

Read only partition (A)

Partition A is a reflection of the topology and is read-only. That is, this part of the name space cannot be changed programmatically as it is based on the configuration rather than runtime settings.

This partition of the namespace is persistent, with the data stored in the XML repository containing the topological information.

Persistent partitions (B and C)

Partitions B and C are meant primarily for the storage of resource configuration, such as data sources, JMS destinations etc. This data can be modified using accessing the JNDI APIs directly, or via the admin clients (which access the APIs on the user’s behalf). The persistent data is stored to a group of XML files.

There are two persistent partitions in the federated namespace:

- Cell persistent root (B)

  Used to register persistent objects that are available to all the nodes and managed processes of a cell. A binding created under the cell persistent root is saved as part of the cell configuration and continues to exist until it is explicitly removed.

  Applications that need to create additional persistent object bindings associated with the cell can bind those objects under the cell persistent root.

  Note:

  - The cell persistent root is not designed for transient, rapidly-changing bindings. Instead, the bindings should be more static in nature, such as part of application setup or configuration, and not created at runtime.
  - In order to bind objects to the cell persistent root, the deployment manager and all node agents in the cell must be running.

  An important role of the cell persistent root is as the initial context for clients running in previous WebSphere releases. If an EJB is to be accessed by WebSphere 4.0.x or 3.5.x clients, a binding for it must be added to the cell persistent root. See Section 6.5, “Configured bindings” on page 123 for details.

- Node persistent root (C)

  Used to register persistent objects that are available to the nodes and its managed processes. Similar to the cell partition except that each node has its own node persistent root. A binding created under a node persistent root is saved as part of that node's configuration and continues to exist until it is explicitly removed.

Applications that need to create additional persistent object bindings associated with a specific node can bind those objects under that particular node's node persistent root.

Note:

- The node persistent area is not designed for transient, rapidly-changing bindings. Instead, the bindings should be more static in nature, such as part of application setup or configuration, and not created at runtime.
- In order to bind objects to a node persistent root, the node agent for the node must be running.
- In the federated name space, there is no node root for the Deployment Manager node since no node agent or application servers run in that node.

The node persistent root plays no special role in interoperability with WebSphere clients of previous releases.

Author Comment: The node persistent root may disappear from WAS 5.0. If so, then this section must be removed.

Transient partitions (D)

Partition D in Figure 6-2 is updateable through APIs, and is meant for information such as EJB bindings and JNDI names.

This partition of the namespace is transient. Each time a server process starts, it reads settings from the filesystem, e.g.. EJB deployment descriptors, to register the necessary objects in this space.

Note: The Name Service of each managed process is a listener to configuration changes. This means that the local namespace will be updated automatically when configuration changes occur. For example, there is no need to restart a node agent to update its namespace when a new application server is created.

6.2.5 Local namespace structure

The structure of the federated namespace is hierarchical, with each process' local namespace federated / linked using corbaloc URLs that allow transparent name searches both within a single local namespace and from one local namespace to another.

The contents of the cell, node and process local namespaces are described in the following sections.

Cell level namespace

The cell level namespace, hosted by the Deployment Manager, has a structure as shown in Example 6-4.

Note: (top) represents the root of the federated namespace.

Example 6-4 Cell level namespace

```
(top)
(top)/cell
      Linked to context: Net1Network
(top)/nodes
(top)/nodes/Net1Manager
(top)/nodes/Net1Manager/servers
(top)/nodes/Net1Manager/servers/dmgr
(top)/nodes/Net1Manager/servers/dmgr/jta
(top)/nodes/Net1Manager/servers/dmgr/jta/usertransaction
(top)/nodes/Net1Manager/servers/dmgr/cell
      Linked to context: Net1Network
(top)/nodes/Net1Manager/servers/dmgr/servername
(top)/nodes/Net1Manager/servers/dmgr/RMIConnector
(top)/nodes/Net1Manager/servers/dmgr/TransactionFactory
(top)/nodes/Net1Manager/servers/dmgr/thisNode
      Linked to context: Net1Network/nodes/Net1Manager
(top)/nodes/Net1Manager/cell
      Linked to context: Net1Network
(top)/nodes/Net1Manager/domain
      Linked to context: Net1Network
(top)/nodes/Net1Manager/node
      Linked to context: Net1Network/nodes/Net1Manager
(top)/nodes/Net1_JH
      Linked to URL: corbaloc::mka0kkcf:2809/NameServiceNodeRoot
(top)/deploymentManager
      Linked to context: Net1Network/nodes/Net1Manager/servers/dmgr
(top)/persistent
(top)/persistent/cell
      Linked to context: Net1Network
(top)/domain
      Linked to context: Net1Network
(top)/legacyRoot
      Linked to context: Net1Network/persistent
```

```
(top)/cells
(top)/cellname
(top)/clusters
```

The cell level namespace contains:

- A hierarchy of contexts for nodes and the servers managed by each node.
- A full set of entries for the Deployment Manager node (<depmgr hostname>Manager) and the Deployment Manager server (dmgr).
- The objects registered in JNDI by the dmgr server.
- A corbaloc URL link to the local namespace of each of the other nodes in the cell, e.g.. Net1_JH in Example 6-4.
- A number of cross-links for the federated namespace:
  - /cells
  - /clusters
  - /legacyRoot
- A link to the cell persistent root, /persistent/cell.

Note: Because of the hierarchical structure of the cell <-> node <-> server relationship, the following naming conventions and constraints exist:

1. No two nodes can have the same name. Node names must be unique within a cell.
2. Two application servers on different nodes can have the same name.
3. Two application servers on the same node must have different names. Application server names must be unique within a node.
4. The JMS Server is called *jmsserver* on each node.

Node level namespace

A node level namespace, hosted by a node agent, has a structure as shown in Example 6-5.

Example 6-5 Node level namespace

```
(top)
(top)/cell
      Linked to context: Net1Network
(top)/nodes
(top)/nodes/Net1_JH
(top)/nodes/Net1_JH/nodeAgent
      Linked to context: Net1Network/nodes/Net1_JH/servers/Net1_JH
(top)/nodes/Net1_JH/node
```

```
      Linked to context: Net1Network/nodes/Net1_JH
(top)/nodes/Net1_JH/nodename
(top)/nodes/Net1_JH/persistent
(top)/nodes/Net1_JH/domain
      Linked to context: Net1Network
(top)/nodes/Net1_JH/servers
(top)/nodes/Net1_JH/servers/server1
      Linked to URL: corbaloc::mka0kkcf:9810/NameServiceServerRoot
(top)/nodes/Net1_JH/servers/Net1_JH
(top)/nodes/Net1_JH/servers/Net1_JH/cell
      Linked to context: Net1Network
(top)/nodes/Net1_JH/servers/Net1_JH/servername
(top)/nodes/Net1_JH/servers/Net1_JH/RMIConnector
(top)/nodes/Net1_JH/servers/Net1_JH/thisNode
      Linked to context: Net1Network/nodes/Net1_JH
(top)/nodes/Net1_JH/cell
      Linked to context: Net1Network
(top)/deploymentManager
      Linked to URL: corbaloc::mka0kkcf:9809/NameServiceServerRoot
(top)/persistent
(top)/persistent/cell
      Linked to context: Net1Network
(top)/domain
      Linked to context: Net1Network
(top)/legacyRoot
      Linked to context: Net1Network/persistent
(top)/cells
(top)/cellname
(top)/clusters
```

The node level namespace contains:

- A full set of entries for the node and the node agent server (<nodename> or nodeAgent).
- A corbaloc URL link to the local namespace root (NameServiceServerRoot) of each application server managed by the node.

  Note: JMS Servers are not registered in the federated namespace, since they do not host a local namespace.

- A corbaloc URL link to the root of the Deployment Manager local namespace (NameServiceServerRoot).
- A number of cross-links for the federated namespace:
  - /cells
  - /clusters

  – /legacyRoot

- A link to the cell persistent root, /persistent/cell.
- A link to the node persistent root, /nodes/<nodename>/persistent.

Author Comment: The node persistent root may disappear from WAS 5.0. If so, then the above line must be removed, plus the node persistent root removed from the example.

Managed process level namespace

A process level namespace, hosted by a managed process, has a structure as shown in Example 6-6.

Example 6-6 Managed process level namespace

```
(top)
(top)/persistent
(top)/persistent/cell
      Linked to context: Net1Network
(top)/legacyRoot
      Linked to context: Net1Network/persistent
(top)/domain
      Linked to context: Net1Network
(top)/cells
(top)/clusters
(top)/cellname
(top)/cell
      Linked to context: Net1Network
(top)/deploymentManager
      Linked to URL: corbaloc::mka0kkcf:9809/NameServiceServerRoot
(top)/nodes
(top)/nodes/Net1_JH
(top)/nodes/Net1_JH/persistent
(top)/nodes/Net1_JH/nodename
(top)/nodes/Net1_JH/domain
      Linked to context: Net1Network
(top)/nodes/Net1_JH/cell
      Linked to context: Net1Network
(top)/nodes/Net1_JH/servers
(top)/nodes/Net1_JH/servers/Net1_JH
      Linked to URL: corbaloc::mka0kkcf:2809/NameServiceServerRoot
(top)/nodes/Net1_JH/servers/server1
(top)/nodes/Net1_JH/servers/server1/plantsby
(top)/nodes/Net1_JH/servers/server1/plantsby/InvQCF
(top)/nodes/Net1_JH/servers/server1/plantsby/LoginHome
top)/nodes/Net1_JH/servers/server1/plantsby/ReportGeneratorHome
(top)/nodes/Net1_JH/servers/server1/plantsby/MailerHome
(top)/nodes/Net1_JH/servers/server1/plantsby/CatalogHome
```

```
(top)/nodes/Net1_JH/servers/server1/plantsby/InvQ
(top)/nodes/Net1_JH/servers/server1/url
(top)/nodes/Net1_JH/servers/server1/TransactionFactory
(top)/nodes/Net1_JH/servers/server1/servername
(top)/nodes/Net1_JH/servers/server1/RMIConnector
(top)/nodes/Net1_JH/servers/server1/thisNode
      Linked to context: Net1Network/nodes/Net1_JH
(top)/nodes/Net1_JH/servers/server1/cell
      Linked to context: Net1Network
(top)/nodes/Net1_JH/servers/server1/ejb
(top)/nodes/Net1_JH/servers/server1/ejb/ivtEJBObject
(top)/nodes/Net1_JH/servers/server1/eis
(top)/nodes/Net1_JH/servers/server1/eis/jdbc
(top)/nodes/Net1_JH/servers/server1/eis/jdbc/PlantsByWebSphereDataSource_CMP
(top)/nodes/Net1_JH/servers/server1/jta
(top)/nodes/Net1_JH/servers/server1/jta/usertransaction
(top)/nodes/Net1_JH/servers/server1/jdbc
(top)/nodes/Net1_JH/servers/server1/jdbc/PlantsByWebSphereDataSource
(top)/nodes/Net1_JH/node
      Linked to context: Net1Network/nodes/Net1_JH
(top)/nodes/Net1_JH/nodeAgent
      Linked to URL: corbaloc::mka0kkcf:2809/NameServiceServerRoot
```

The process level namespace contains:

- A full set of entries for objects registered in the local namespace of the process. These entries include resources (JDBC, JMS, etc.) read from the resources.xml of the process, as well as those registered at runtime by applications, e.g.. EJB homes.
- A corbaloc URL link to the local namespace root (NameServiceServerRoot) of the node agent, for both the real node agent server name (<nodename>) and its alias (/nodes/<nodename>/nodeAgent).
- A corbaloc URL link to the root of the Deployment Manager local namespace (NameServiceServerRoot).
- A number of cross-links for the federated namespace:
  - /cells
  - /clusters
  - /legacyRoot
- A link to the cell persistent root, /persistent/cell.

- A link to the node persistent root, /nodes/<nodename>/persistent.

Author Comment: The node persistent root may disappear from WAS 5.0. If so, then the above line must be removed, plus the node persistent root removed from the example.

6.3 Interoperable Naming Service (INS)

In order to meet the requirements of the J2EE 1.3 specification, support for Interoperable Naming Service (INS) CosNaming is required. The INS allows J2EE application servers to deal with and understand names formulated according to the CORBA 2.3 naming scheme. The main advantage of INS is that it improves interoperability with other application server products, as well as CORBA servers.

The naming architecture of IBM WebSphere Application Server is compliant with the Interoperable Naming Service (INS).

The requirements of INS CosNaming include:

- *corbaloc* and *corbaname* URLs must be supported, in addition to the IIOP URL supported in WebSphere 4.0.
  - corbaloc

    Designates an endpoint, such as a host machine.
  - corbaname

    Designates an object’s name.
- The default bootstrap port must be 2809, as compared to the default of 900 used in earlier versions of IBM WebSphere Application Server.

6.3.1 Bootstrap ports

Every WebSphere 5.0 process, except the JMS Server, has a bootstrap server and hence port assignment.

Each process on a given machine (and WebSphere logical node) requires unique ports, including the bootstrap port. The default port assignments are:

- Single server

  Default for application server is 2809. Each subsequently created application server will be assigned a unique port number that does not conflict.
- Network deployment

Default for node agent is 2809. Application servers are each assigned a unique non-default port, either explicitly by the Administrator, or automatically determined by the administration tool.

6.3.2 CORBA URLs

Author Comment: We mention CORBA and ORBs several times throughout the chapter and probably throughout the book. We should make sure there is a brief definition of these somewhere.

CORBA URL syntax (both corbaloc and corbaname) is supported by IBM WebSphere Application Server.

corbaloc

The corbaloc form of the CORBA 2.3 URL has the following syntax.

```
corbaloc:<protocol>:<addresslist>/<key>
```

Table 6-1 corbaloc options

Setting	Description
protocol	The protocol used for the communication. Valid values are: iiop.
addresslist	List of one or more addresses (host name and port number). The addresses are separated by commas, and each address has a colon prefix.
key	Type of root to access. See Table 6-5 on page 129 for details.

The following examples illustrate how the corbaloc URL can range from simple to complex, depending upon whether fault tolerance (request retry with second, third etc. server) is required.

Example 6-7 corbaloc - basic

```
corbaloc::myhost
```

Example 6-8 corbaloc - cell's namespace root via specific server.

```
corbaloc:iiop:1.2@myhost.raleigh.ibm.com:9344/NameServiceCellRoot
```

Example 6-9 corbaloc - server namespace root with fault tolerance

```
corbaloc::myhost1:9333,:myhost2:9333,:myhost2:9334/NameServiceServerRoot
```

Note: corbaloc URLs are usually used for the provider URL when retrieving an InitialContext.

corbaname

The corbaloc form of the CORBA 2.3 URL has the following syntax.

```
corbaname:<protocol>:<addresslist>/<key>#<INS string-formatted-name>
```

Table 6-2 corbaname options

Setting	Description
protocol	The protocol used for the communication. Valid values are: iiop.
addresslist	List of one or more addresses (hostname and port number). The addresses are separated by commas, and each address has a colon prefix.
key	Type of root to access. See Table 6-5 on page 129 for details.
`<INS string-formatted-name>`	Fully qualified path to entry under the specific root context.

The following examples illustrate how the corbaname URL can range from simple to complex, depending upon whether fault tolerance (request retry with second, third, etc. server) is required.

Example 6-10 corbaname - fully qualified name access

```
corbaname::myhost:9333#cell/nodes/node1/servers/server5/someEjb
```

Example 6-11 corbaname - object accessed via specific server root

```
corbaname::myhost:9333/NameServiceServerRoot/someEjb
```

Note: corbaname URLs are usually used when performing a direct URL lookup using a previously obtained InitialContext, e.g.. ic.lookup("urlstring").

6.4 Distributed CosNaming

One of the advantages of the new distributed nature of CosNaming in IBM WebSphere Application Server, is that it removes the bottleneck of having a single name server for all naming lookups.

In WebSphere 5.0, each WebSphere process (Deployment Manager, node agent and application server) hosts its own ORB, NameService and local namespace. As such, lookups are made by accessing the NameService on the most convenient process, they are not bottle necked through a single server process in the cell.

Note: The JMS Server process does not host an ORB, NameService or a local namespace. Any references to JMS objects running in this process are actually registered in the local namespaces of the application server(s) that access the JMS objects.

To enable the new distributed naming architecture, the following major changes have been made to CosNaming:

- The IBM WebSphere Application Server Naming architecture uses CORBA CosNaming as its foundation.
- The CosNaming architecture has been changed to support a distributed and federated collection of CosNaming servers:
  - Each Deployment Manager, node agent and application server is a CosNaming server. Each server is responsible for managing the names of the objects bound locally.
  - Objects are bound into the local context.
  - Lookups start in the local process and end in the process where the target object is located.
  - This reduces the dependency of the WAS network on a single name server for all lookups.
- A single logical namespace exists across the cell. The separate namespaces of each server process are linked / federated via context links in the cell namespace. It is possible to navigate to specific subcontexts, as every server cluster and non-clustered server has its own context stored in a level of the cell namespace
- The contents of the federated namespace are mostly transient, built from configuration data read from the XML configuration files on the startup of each server process.
- Persistent roots are provided at the cell and node level of the namespace in order to provide locations where objects can be persistently bound. These bindings are persisted to XML files on the filesystem.
- Each separate server process has its own bootstrap port, thereby reducing bottlenecks.

6.5 Configured bindings

The configured bindings feature allows objects to be added to the namespace using the admin interfaces (`wsadmin` or Web admin console).

The IBM WebSphere Application Server Naming component provides new functionality by which an administrator can explicitly add bindings to the cell namespace without having to write code. This allows an administrator to configure an "alias" (via the admin client) in a persistent namespace that refers to an actual reference in one of the local namespaces. This provides an additional level of indirection for 5.0 names.

The functionality is useful in the following areas:

- Federation of namespaces

  As long as it is CORBA 2.3 compliant (supporting INS), the namespace of other WebSphere 5.0 cells, WebSphere 4.0 administrative domains, third party application servers and even CORBA servers can be federated into the cell's namespace.

- Interoperability with WebSphere 4.0

  The default context of WebSphere 4.0 clients is the global (legacy) context. However, WebSphere 5.0 processes bind their objects in local transient namespaces. Therefore, in order for WebSphere 4.0 clients to lookup and access objects in WebSphere 5.0 without requiring changes to the client, requires that the WebSphere 5.0 object be bound into the legacy namespace accessible to the client. This is where configured bindings come in. An alias can be configured into the legacy namespace. When used by the WebSphere 4.0 client, the client will be transparently redirected to the real object reference in one of the cell's local namespaces.

6.5.1 Types of object

The following types of object can be bound using configured bindings:

- EJB hosted by a server in the cell

  The configured binding identifies an EJB Home based on its configured JNDI name and the server in which it is deployed.

  A possible use of this is to put a binding for an EJB into the cell-scoped namespace so that a lookup can be done without knowledge about the server in which the EJB is deployed. This mechanism is useful for allowing WebSphere 4.0 clients to look up WebSphere 5.0 EJBs without having to redeploy.

- CORBA object

The configured binding identifies a CORBA object bound somewhere in this or another namespace by using a corbaname URL string. Included is also an indicator of whether the object is a CosNaming NamingContext, in which case the binding is actually a federated link from one namespace to another.

- JNDI name

  The configured binding identifies a provider URL and a JNDI name that can be used to lookup an object. This can be used to reference a resource or (other Java serialized object) bound elsewhere in this namespace or another namespace.

- String constant

  Can be used to bind environment data into the namespace.

6.5.2 Types of binding reference

There are a number of different types of references that can be specified for configured bindings. Valid types are summarized in Table 6-3.

Table 6-3 Types of binding reference

Binding type	Required Settings
EjbNameSpaceBinding	1. Name in namespace is relative to configured root. 2. JNDI name of EJB. 3. Server or server cluster where EJB is deployed.
CorbaObjectNameSpaceBinding	1. Name in namespace is relative to configured root. 2. corbaname URL. 3. Indicator if the target object is a federated context object, or a leaf node object
IndirectLookupNameSpaceBinding	1. Name in namespace is relative to configured root. 2. Provider URL. 3. JNDI name of object.
StringNameSpaceBinding	1. Name in namespace is relative to configured root. 2. Constant string value.

The configured bindings can be relative to one of the following context roots:

- Server root
- Node persistent root
- Cell persistent root

6.6 Initial contexts

In WebSphere, an initial context for a name server is associated with a bootstrap host and bootstrap port. These combined values can be viewed as the address of the name server which owns the initial context. To get an initial context, the bootstrap host and port for the initial context's name server must be known.

JNDI clients should assume the correct environment is already configured so there is no need to explicitly set property values and pass them to the InitialContext constructor.

However, a JNDI client may need to access a name space other than the one identified in its environment. In this case, it is necessary to explicitly set the javax.naming.provider.url (provider URL) property used by the InitialContext constructor. A provider URL contains bootstrap server information that the initial context factory can use to obtain an initial context. Any property values passed in directly to the InitialContext constructor take precedence over settings of those same properties found elsewhere in the environment.

Two different provider URL forms can be used with WebSphere's initial context factory:

- CORBA object URL (new for J2EE 1.3)
- IIOP URL

CORBA object URLs are more flexible than IIOP URLs and are the recommended URL format to use. CORBA object URLs are part of the OMG CosNaming Interoperable Naming Specification. The IIOP URLs are the legacy JNDI format, but are still supported by the WebSphere initial context factory.

The following examples illustrate the use of these URLs.

Using a CORBA object URL

An example of using a corbaloc URL with a single address to obtain an initial context is shown in Example 6-12.

Example 6-12 Initial context using CORBA object URL

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL, "corbaloc:iiop:myhost.mycompany.com:2809");

Context initialContext = new InitialContext(env);
```

Using a CORBA object URL with multiple addresses

CORBA object URLs can contain more than one bootstrap server address. This feature can be used in WebSphere when attempting to obtain an initial context from a server cluster. The bootstrap server addresses for all servers in the cluster can be specified in the URL. The operation will succeed if at least one of the servers is running, eliminating a single point of failure.

Note: There is no guarantee of any particular order in which the address list will be processed. For example, the second bootstrap server address may be used to obtain the initial context even though the first bootstrap server in the list is available.

An example of using a corbaloc URL with multiple addresses to obtain an initial context is shown in Example 6-13.

Example 6-13 Initial context using CORBA object URL with multiple addresses

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL,
"corbaloc::myhost1:2809,:myhost2:2809,:myhost3:2809");

Context initialContext = new InitialContext(env);
```

Using a CORBA object URL from non-WebSphere JNDI

To access a WebSphere name server from a non-WebSphere environment, such that the WebSphere initial context factory is not used, a corbaloc URL must be used that has an object key of *NameServiceServerRoot* to identify the server root context.

The server root is where system artifacts such as EJB homes are bound. The default key of NameService can be used when fully-qualified names are used for JNDI operations.

Example 6-14 shows a CORBA object type URL from a non-WebSphere JNDI implementation. It assumes full CORBA object URL support by the non-WebSphere JNDI implementation.

Example 6-14 Using a CORBA object URL from non-WebSphere JNDI

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;

Hashtable env = new Hashtable(); env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.somecompany.naming.TheirInitialContextFactory");
env.put(Context.PROVIDER_URL,
"corbaname:iiop:myhost.mycompany.com:2809/NameServiceServerRoot");

Context initialContext = new InitialContext(env);
```

Using an IIOP URL

The IIOP type of URL is a legacy format which is not as flexible as CORBA object URLs. However, URLs of this type are still supported by the WebSphere initial context factory.

Example 6-15 shows an IIOP type URL as the provider URL.

Example 6-15 Initial context using an iiop URL

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL, "iiop://myhost.mycompany.com:2809");

Context initialContext = new InitialContext(env);
```

6.6.1 Setting initial root context

Each server contains its own server root context, and when bootstrapping to a server, the server root is the default initial JNDI context. Most of the time, this is the desired initial context, since system artifacts such as EJB homes are bound at this point. However, other root contexts exist which may contain bindings of interest. It is possible to specify a provider URL to select other root contexts.

Note: The default is that the name will be resolved based upon the context associated with the server whose bootstrap the client is connected.

The initial root context can be selected using the following settings:

- CORBA object URL
- Name space root property

Default initial context

The default initial context depends on the type of client. Table 6-4 summarizes the different categories of clients and the corresponding default initial context.

Table 6-4 Default initial context Vs. client type

Client type	Description	Default initial context
WebSphere 5.0 JNDI	The JNDI interface is used by EJB applications to perform name space lookups. WebSphere clients by default use WebSphere's CosNaming JNDI plug-in implementation.	Server root
WebSphere JNDI prior to 5.0	WebSphere clients running in releases prior to 5.0 by default use WebSphere's 4.0 CosNaming JNDI plug-in implementation.	Cell persistent root (legacy root)
Other JNDI	Some applications may perform name space lookups with a non-WebSphere CosNaming JNDI plug-in implementation.	Cell root
CORBA		Cell root

Selecting initial root context with a CORBA object URL

There are several object keys registered with the bootstrap server which can be used to select the root context to be used as the initial context. To select a particular root context with a CORBA object URL object key, set the object key to the corresponding value. The default object key is NameService. Using JNDI, this will yield the server root context.

Table 6-5 lists the different root contexts and their corresponding object key.

Table 6-5 CORBA object URL root context values

Root context	CORBA object URL object key	Description
Server root	NameServiceServerRoot	Server root for the server being accessed.
Cell persistent root	NameServiceCellPersistentRoot	The persistent cell root for the server being accessed.
Cell root	NameServiceCellRoot	The cell root for the server being accessed.
Node root	NameServiceNodeRoot	The node root for the server being accessed.

Note: The name server running in the Deployment Manager process has no node root registered under the NameServiceNodeRoot key, because there is no node agent or application servers running in its node.

Example 6-16 shows the use of a corbaloc URL with the object key set to select the cell persistent root context as the initial context.

Example 6-16 Select cell persistent root context using corbaloc URL

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL,
"corbaloc:iiop:myhost.mycompany.com:2809/NameServiceCellPersistentRoot")
;

Context initialContext = new InitialContext(env);
```

Selecting initial root context with namespace root property

The initial root context can be selected by passing a name space root property setting to the InitialContext constructor. Generally, the object key setting described above is sufficient. Sometimes a property setting might be preferable.

For example, the root context property can be set on the java invocation to make it transparent to the application which server root is being used as the initial context. The default server root property setting is *defaultroot*, which will yield the server root context.

Tip: If a simple name is used, the root context that will be assumed can be set by passing the `com.ibm.websphere.naming.namespaceroot` property to InitialContext.

Table 6-6 name space root values

Root context	CORBA object URL object key
Server root	bootstrapserverroot
Cell persistent root	cellpersistentroot
Cell root	cellroot
Node root	bootstrapnoderoot

The name space root property is used to select the default root context only if the provider URL does not contain an object key or contains the object key, *NameService*. Otherwise, the property is ignored.

Example 6-17 shows use of the name space root property to select the cell persistent root context as the initial context.

Tip: WebSphere makes available constants that can be used instead of hard-coding the property name and value, e.g..

```
env.put(PROPS.NAME_SPACE_ROOT, PROPS.NAME_SPACE_ROOT_CELL_PERSISTENT);
```

Example 6-17 Use of name space root property to select cell persistent root context

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;
import com.ibm.websphere.naming.PROPS;

Hashtable env = new Hashtable();
```

```
env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL, "corbaloc:iiop:myhost.mycompany.com:2809");
env.put(PROPS.NAME_SPACE_ROOT,
PROPS.NAME_SPACE_ROOT_CELL_PERSISTENT);

Context initialContext = new InitialContext(env);
```

6.7 Federation of name spaces

Federating name spaces involves binding contexts from one name space into another name space. In an IBM WebSphere Application Server name space, federated bindings can be created with the following restrictions:

- Federation is limited to CosNaming name servers. A WebSphere name server is a Common Object Request Broker Architecture (CORBA) CosNaming implementation.

  Federated bindings to other CosNaming contexts can be created, but bindings to LDAP name server implementation contexts cannot.
- If JNDI is used to federate the name space, the WebSphere initial context factory must be used to obtain the reference to the federated context. If any other initial context factory implementation is used, the binding may not be able to be created, or the level of transparency may be reduced.
- A federated binding to a non-WebSphere naming context has the following functional limitations:
  - JNDI operations are restricted to the use of CORBA objects. For example, EJB homes can be looked up, but non-CORBA objects such as data sources cannot.
  - JNDI caching is not supported for non-WebSphere name spaces. This only affects the performance of lookup operations.
- Do not federate two WebSphere single server name spaces. If this is done, incorrect behavior may result. If federation of WebSphere name spaces is required, then servers running under IBM WebSphere Application Server Network Deployment need to be used.

As an example, assume that a name space, Name Space 1, contains a context under the name "a/b." Also assume that a second name space, Name Space 2, contains a context under the name "x/y." If context "x/y" in Name Space 2 is bound into context "a/b" in Name Space 1 under the name "f2," the two name spaces are federated. Binding "f2" is a federated binding because the context associated with that binding comes from another name space. As shown in Figure 6-3, from Name Space 1, a lookup of the name "a/b/f2" would return the context bound under the name "x/y" in Namespace 2. Furthermore, if context "x/y" contained an EJB Home bound under the name "ejb1," the EJB home could be looked up from Namespace1 with the lookup name "a/b/f2/ejb1." Notice that the name crosses namespaces. This fact is transparent to the naming client.

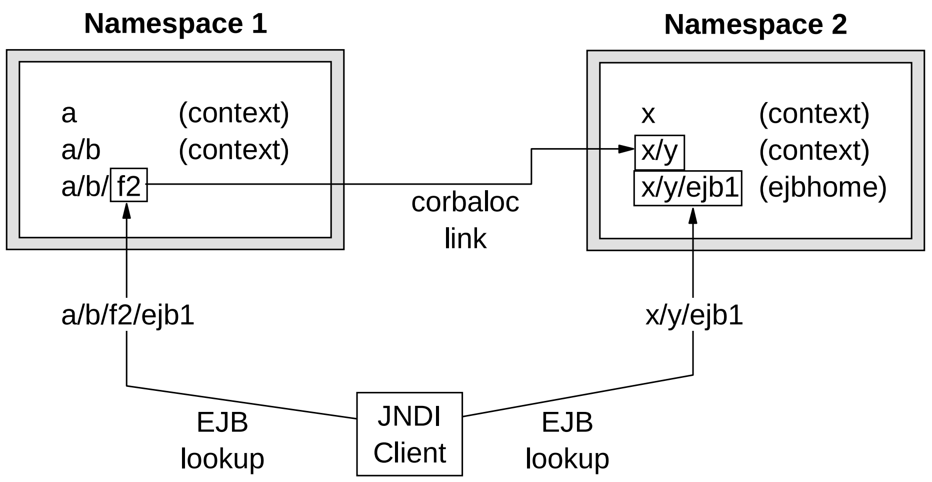


Figure 6-3 JNDI access using federated namespaces

6.8 Interoperability

IBM WebSphere Application Server provides support for interoperating with previous releases of WebSphere and with non-WebSphere JNDI clients:

- EJB clients running on WebSphere 3.5.x or 4.0.x, accessing EJB applications running on WebSphere 5.0.
- EJB clients running on WebSphere 5.0, accessing EJB applications running on WebSphere 3.5.x or 4.0.x servers.
- EJB clients running in an environment other than WebSphere, accessing EJB applications running on WebSphere 5.0 servers.

6.8.1 WebSphere 3.5 / 4.0 EJB clients

Applications migrated from previous WebSphere releases may still have clients running in a previous release. The default initial JNDI context for EJB clients running on previous versions of WebSphere is the cell persistent root (legacy root). However, the home for an EJB deployed in release 5.0 is bound to its server's server root context. In order for the EJB lookup name for down-level clients to remain unchanged, configure a binding for the EJB home under the cell persistent root. For more information regarding configured bindings, see Section 6.5, “Configured bindings” on page 123.

The following options are available for enabling interoperability with WebSphere 4.0 clients:

- Set the client’s default initial context to legacyRoot. This option is equivalent to the cell persistent root of WebSphere 5.0.
- Redeploy the clients using the Application Assembly Tool (AAT). So that the JNDI names can be fixed to reflect the actual fully qualified names in the WebSphere 5.0 namespace.
- Use aliases for the names the clients look up. These transparently redirect to the correct object in the WebSphere 5.0 namespace. This option uses configured bindings.

Important: In order to interoperate with WebSphere 5.0, EJB clients running on WebSphere 3.5.x must be running at 3.5.5, or at 3.5.3 / 3.5.4 with e-fix PQ51387 (or an updated e-fix) installed.

Options for EJB lookup

The following options exist for supporting EJB lookup from a WebSphere 4.0 client to a WebSphere 5.0 hosted EJB:

1. Redeploy the WebSphere 4.0 client.

   The <ejb-ref> needs to be updated to reflect the WebSphere 5.0 compatible JNDI name.

2. In WebSphere 5.0, configure EjbNameSpaceBinding:

   a. Use the same JNDI name as looked up by the WebSphere 4.0 client.

   b. Identify the JNDI name and server (or cluster) of the target EJB.

   c. Configure the binding in the cell persistent root.

Options for resources bound in external namespace

1. Redeploy the WebSphere 4.0 client.

   The <resource-ref> needs to be updated to reflect the WebSphere 5.0 compatible JNDI name.

2. In WebSphere 5.0, run the program to bind the resource into the WebSphere 5.0 cell persistent root.

3. In WebSphere 5.0, configure IndirectLookupNameSpaceBinding:

   a. Use the same JNDI name as looked up by the WebSphere 4.0 client.

b. Specify the provider URL and JNDI name of the namespace where the resource is already bound (a WebSphere 4.0 name space).

c. Configure the binding in the cell persistent root.

6.8.2 WebSphere 3.5 / 4.0 servers

The default initial context for a WebSphere 3.5 or 4.0 server is the correct context. WebSphere 5.0 clients simply look up the JNDI name under which the EJB home is bound.

Important: To enable WebSphere 5.0 clients to access version 3.5.x and 4.0.x servers, the down-level installations must have e-fix PQ60074 (or an updated e-fix) installed.

6.8.3 EJB clients hosted by non-WebSphere environment

When an EJB application running in WebSphere 5.0 is accessed by a non-WebSphere EJB client, the JNDI initial context factory is presumed to be a non-WebSphere implementation. In this case, the default initial context will be the cell root. If the JNDI service provider being used supports CORBA object URLs, the following corbaname format can be used to look up the EJB home.

Example 6-18 provides an example of this form of EJB Home lookup. According to the URL in the example, the bootstrap host and server are *myHost* and *2809*, and the EJB is installed in a server cluster named *myCluster*. The EJB is bound in that cluster under the name *myEJB*.

Note: The server name could just as well be the name of a non-clustered server. This form of lookup will work:

- With any name server bootstrap host and port configured in the same cell.
- If the bootstrap host and port belongs to a member of the cluster itself.

Example 6-18 Use corbaname format for EJB home lookup

```
initialContext.lookup("corbaname:iiop:myHost:2809#cell/nodes/node1/servers/myCl
uster/myEJB");
```

To avoid a single point of failure, the bootstrap server host and port for each cluster member could be listed in the URL as shown in Example 6-19.

Example 6-19 Use corbaname format with multiple addresses for EJB Home lookup

```
initialContext.lookup("corbaname:iiop:host1:9810,host2:9810#cell/nodes/node1/se
rvers/myCluster/myEJB");
```

The name prefix *cell/nodes/<nodename>/servers/<clustername>/* is not necessary if bootstrapping to the cluster itself, but it will always work. The prefix is required however, when looking up EJBs in other clusters. The server binding for the prefix used to access another cluster is implemented in a way that avoids a single point of failure during a lookup.

If the JNDI initial context factory being used does not support CORBA object URLs, the initial context can be obtained from the server, and the lookup can be performed on the initial context as shown in Example 6-20.

Example 6-20 Use corbaname format with multiple addresses for EJB Home lookup

```
Hashtable env = new Hashtable();
env.put(CONTEXT.PROVIDER_URL, "iiop://myHost:2809");
Context ic = new InitialContext(env);
Object o = ic.lookup("cell/nodes/node1/servers/myCluster/myEJB");
```

This form of lookup will work from any server in the same cell as the EJB Home being looked up. However, this approach does not allow multiple hosts and ports to be specified in the provider URL and hence does not incorporate the availability advantages of a corbaloc or corbaname URL with multiple hosts and ports belonging to the server cluster members.

6.9 Examples

The following examples hilight a number of different server topologies and the affect the topologies have on the use of the Naming service:

- Single server.
- Single server with non-default port.
- Two single servers on same box.
- Two Network Deployment application servers on same box.
- WebSphere 4.0 client.

6.9.1 Single server

In the single server environment, the naming functionality works in exactly the same way as in WebSphere 4.0. There is only one server, only one root context and therefore no ambiguity in the location of a named object. This is illustrated by the example in Figure 6-4.

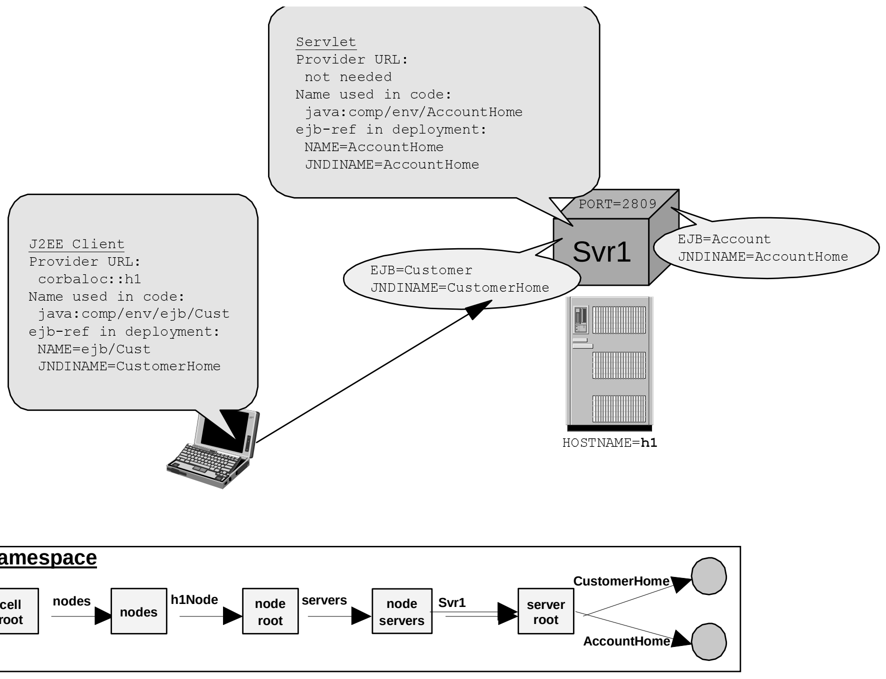


Figure 6-4 Single server

By accessing the server root directly, the client (J2EE or servlet) does not need to traverse the cell namespace (cell root -> servers -> server root -> object).

Note: Even in a single server case, clients can still use the fully qualified JNDI name to lookup an object from the cell root. This would remove any dependency on the particular topology in use, at the cost of a small performance degradation due to the extra lookup to retrieve the server root from the cell namespace.

In a single server case, the server root acts as the default bootstrap, and therefore should be assigned port 2809. In this case, the provider URL used by clients external to the server process, does not need to include a port number.

Tip: If the named object is looked up by a client running in the same process, then a provider URL (and hence corbaloc) does not need to be provided. By default the lookup will be performed against the local process' namespace.

Table 6-7 illustrates the different Provider URL settings required by clients in the same and different process to the named object.

Table 6-7 Lookup settings required for single server

Component	Provider URL	JNDI name
Servlet (same process	*Not needed*	CustomerHome
J2EE client (external process)	corbaloc::<hostname>	CustomerHome

6.9.2 Single server with non-default port

The single server topology can be configured so that the bootstrap of the server uses a non-default port. Instances where this can prove useful include:

- Multiple instances of an Application Server are running on the same box. Each server has its own bootstrap port, so only one of the servers can use 2809. The other servers must be configured to use non-default ports.

This is illustrated by the example in Figure 6-5.

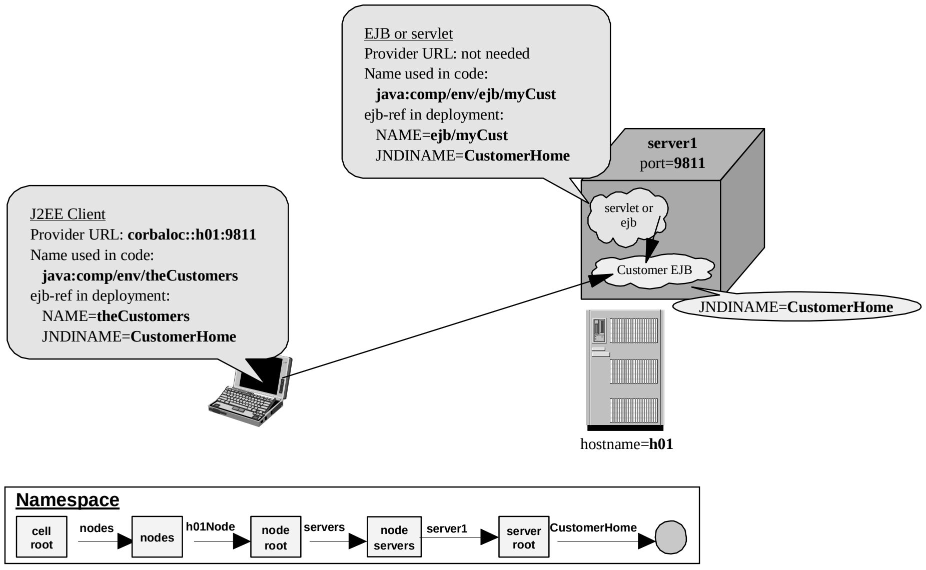


Figure 6-5 Single server with non-default port

As shown by Figure 6-5, the only component affected is the J2EE client (external). Servlets and EJBs hosted in the same process as the named object will by default lookup objects in the local namespace, unless specifically instructed otherwise.

Table 6-8 illustrates the different Provider URL settings required by clients in the same and different process to the named object.

Table 6-8 Lookup settings required for single server on non-default port

Component	Provider URL	JNDI name
Servlet (same process)	*Not needed*	CustomerHome
J2EE client (external process)	corbaloc::<hostname>:<port#>	CustomerHome

6.9.3 Two single servers on same box

When more than one instance of the application server runs on a single machine, then each server's bootstrap must be configured to run on a different port.

In this case a number of new lookup possibilities exist:

- J2EE component in one server looking up objects in the other server.

This is illustrated by the example in Figure 6-6.

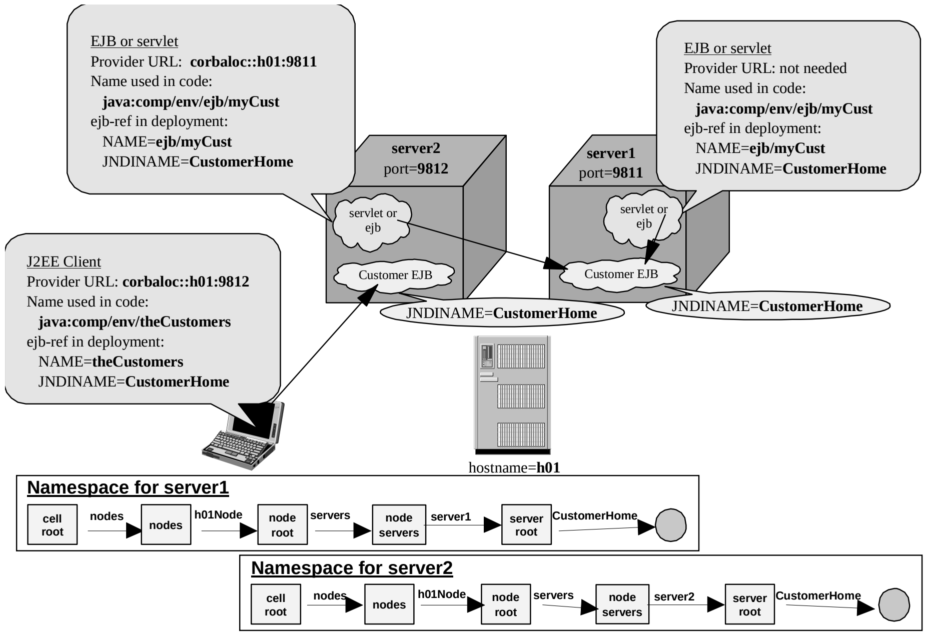


Figure 6-6 Two single servers on same box

Since each of the application server namespaces are separate, the different objects can be registered in each under the same name, e.g.. CustomerHome. There is no problem with name collision.

Note: The fully qualified JNDI name can be used to uniquely identify the name registered in one server, from the name in another, e.g.. if objects are registered under the name CustomerHome on two servers, the specific name can be looked up using:

```
cell/nodes/<nodename>/servers/<server1>/CustomerHome
cell/nodes/<nodename>/servers/<server2>/CustomerHome
```

Table 6-9 illustrates the different Provider URL settings required by clients in the same and different process to the named object.

Table 6-9 Lookup settings required for two single servers on same box

Component	Provider URL	JNDI name
Servlet (same process)	*Not needed*	CustomerHome
Servlet (external process)	corbaloc::<hostname>:<port#>	CustomerHome
J2EE client (external process)	corbaloc::<hostname>:<port#>	CustomerHome

6.9.4 Two Network Deployment application servers on same box

The configuration becomes more complex when we move from a IBM WebSphere Application Server Base to a Network Deployment configuration. In this topology, there can be a number of separate application servers as well as a node agent process, all of which have a bootstrap port and host a local namespace:

- node agent

  The node agent is the default bootstrap for the box, and therefore has its bootstrap port configured on 2809.

- application servers

  The application servers are not the default bootstrap, and therefore each is configured to use a non-default bootstrap port.

This is illustrated by the example in Figure 6-7.

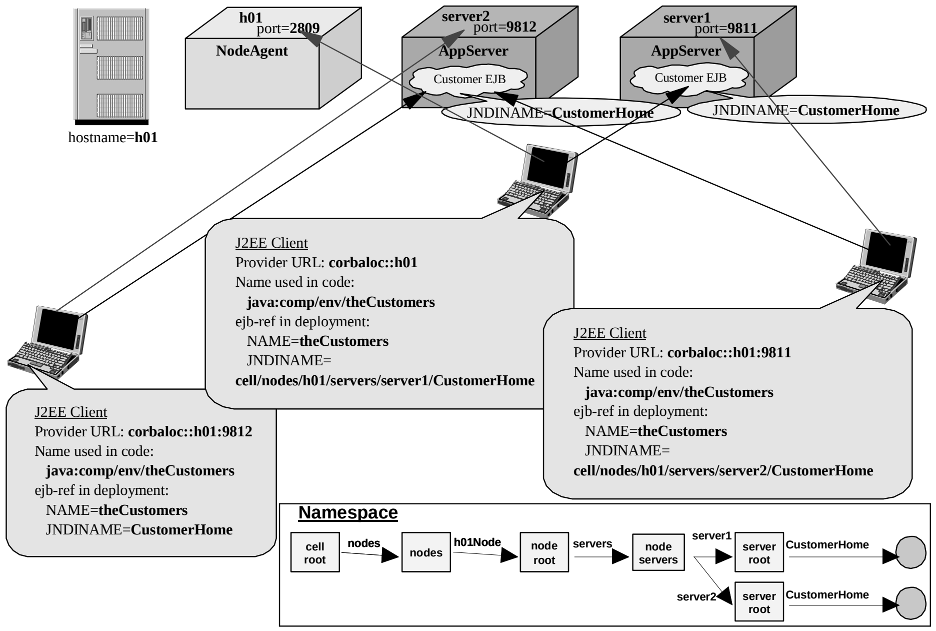


Figure 6-7 Two network deployment application servers on same box

Unless a client specifies a specific application server in its provider URL, the lookup will be performed on the node agent. In order for the lookup to succeed, the bindings have to specify the fully qualified name of the object, i.e.

```
cell/nodes/<nodename>/servers/<servername>/<name of object>
```

That is, the client needs to specify where the object is located. This is a big difference compared to the behavior in WebSphere 4.0, where all named objects were registered in a single global namespace.

Tip: In a Network Deployment configuration where objects may be referenced by clients other than in the same process, it is recommended that all lookup be performed via the node agent, and all JNDI names be fully qualified. This removes any dependency on the particular topology.

Table 6-10 illustrates the different Provider URL settings required by clients in the same and different process to the named object.

Table 6-10 Lookup settings required for two Network Deployment servers on same box

Component	Provider URL	JNDI name
Servlet (same process)	*Not needed*	CustomerHome
Servlet (external process accessing local namespace to access local object)	*Not needed*	cell/nodes/<nodename>/servers/<server2>/CustomerHome
Servlet (external process accessing other appserver's namespace to access object on that appserver)	corbaloc::<appserver2 hostname>:<port#>	CustomerHome
J2EE client (external process accessing appserver1 with object located on appserver1)	corbaloc::<appserver hostname>:<port#>	CustomerHome
J2EE client (external process accessing node agent)	corbaloc::<node agent hostname>	cell/nodes/<nodename>/servers/<server2>/CustomerHome
J2EE client (external process accessing appserver1 with object located on appserver2)	corbaloc::<appserver1 hostname>:<port#>	cell/nodes/<nodename>/servers/<server2>/CustomerHome

6.9.5 WebSphere 4.0 client

In WebSphere 4.0 there is no need to specify a path to a named object, since all objects are registered in a single global namespace. Although convenient, this causes naming conflicts since no two objects can be registered across all application servers with the same names.

The use of configured bindings (aliases) in the cell persistent root provides a mechanism by which the 4.0 style naming structure can be mapped to the fully qualified names of 5.0. This is illustrated by the example in Figure 6-8.

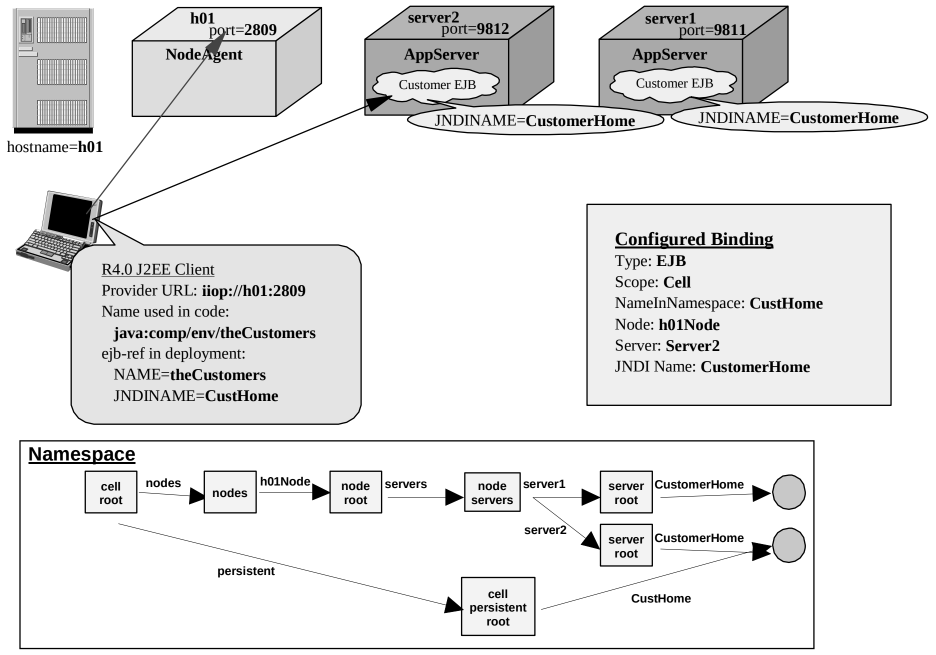


Figure 6-8 WebSphere 4.0 client

Table 6-11 illustrates the different Provider URL settings required by clients in the same and different process to the named object.

Table 6-11 Lookup settings required for WebSphere 4.0 client interoperability

Component	Provider URL	JNDI name
V4.0 client	iiop://<hostname>:2809	CustHome

where CustHome is the name registered in the cell level persistent root (legacy root) for cell/nodes/<nodename>/servers/<servername>/CustomerHome.

The WebSphere 4.0 client accesses the JNDI alias registered in the cell persistent root (legacy root) of the WebSphere 5.0 cell. The WebSphere 5.0 runtime *transparently* redirects the client to the actual JNDI entry located in a specific local namespace hosted by one of the name servers of the cell.

6.10 Naming Tools

IBM WebSphere Application Server provides the following tools for the support of the naming architecture.

6.10.1 dumpNameSpace

The **`dumpNameSpace`** command-line tool can be run against any bootstrap port and get a listing of the names bound with that provider URL.

The output of the command:

- Does not present a full logical view of the namespace.
- Shows CORBA URLs where the namespace transitions to another server.

  The tool indicates that certain names point to contexts that are external to the current server (and its namespace). The links show the transitions necessary to perform a lookup from one namespace to another.

Tip: An invocation of the **`dumpNameSpace`** command cannot generate a dump of the entire namespace, only the objects bound to the bootstrap server and links to other local namespaces that compose the federated namespace. Use the correct hostname and port number for the server to be dumped.

Syntax

```
dumpNameSpace [options]

where all arguments are optional
```

Table 6-12 Options for dumpNameSpace

Option	Description
-host <hostname>	Host name of bootstrap server. If not defined, then default is localhost.
-port <portnumber>	Bootstrap server port number. If not defined, then default is 2809.
-factory <factory>	Initial context factory to be used to get initial context. The default of com.ibm.websphere.naming.WsnInitialContextFactory should be ok for most use.

Option	**Description**
-root [cell | server | node | host | legacy | tree | default]	*For WS 5.0 or later:* cell: dumpNameSpace default. Dump the tree starting at the cell root context. server: Dump the tree starting at the server root context. node: Dump the tree starting at the node root context. (Synonymous with "host") *For WS 4.0 or later:* legacy: DumpNameSpace default. Dump the tree starting at the legacy root context. host: Dump the tree starting at the bootstrap host root context (Synonymous with "node") tree: Dump the tree starting at the tree root context. *For all WebSphere and other name servers:* default: Dump the tree starting at the initial context which JNDI returns by default for that server type. This is the only -root choice that is compatible with WebSphere servers prior to 4.0 and with non-WebSphere name servers.
-url <url>	The value for the java.naming.provider.url property used to get the initial JNDI context. This option can be used in place of the -host, -port, and -root options. If the -url option is specified, the -host, -port, and -root options are ignored.
-startAt <context>	The path from the requested root context to the top level context where the dump should begin. Recursively dumps subcontexts below this point. Defaults to empty string, i.e., root context requested with the -root option.
-format <format>	jndi: Display name components as atomic strings. ins: Display name components parsed per INS rules (id.kind). The default format is "jndi".

Option	Description
-report <length>	short: Dumps the binding name and bound object type, which is essentially what JNDI Context.list() provides. long: Dumps the binding name, bound object type, local object type, and string representation of the local object (i.e., IORs, string values, etc., are printed). The default report option is "short".
-traceString <tracespec>	Trace string of the same format used with servers, with output going to the file "DumpNameSpaceTrace.out".
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 6-21 dumpNameSpace usage

```
$ cd c:\ibm\was50\AppServer\bin

Get help on options:
$ dumpNameSpace -?

Dump server on localhost:2809 from cell root:
$ dumpNameSpace

Dump server on localhost:2806 from cell root:
$ dumpNameSpace -port 2806

Dump server on yourhost:2811 from cell root:
$ dumpNameSpace -port 2811 -host yourhost

Dump server on localhost:2809 from server root:
$ dumpNameSpace -root server‘

Dump server at corbaloc
dumpNameSpace -url corbaloc:iiop:yourhost:901
```

7

Asynchronous messaging

In this chapter we describe the concepts behind the asynchronous messaging functionality provided as part of WebSphere. We will cover:

- Asynchronous messaging
- WebSphere support for asynchronous messaging
- Integral JMS provider
- WebSphere MQ JMS provider
- Generic JMS provider
- WebSphere clusters and MQ clusters
- JMS scenarios

7.1 Asynchronous messaging

Asynchronous messaging provides a method for communication based on the Java Message Service (JMS) programming interface. JMS provides a common mechanism for Java programs (clients and J2EE applications) to create, send, receive, and read asynchronous requests, as JMS messages.

7.1.1 Concepts

Base asynchronous messaging

The base support for asynchronous messaging using JMS includes the common set of JMS interfaces and associated semantics that define how a JMS client can access the facilities of a JMS provider.

This support enables J2EE applications, as JMS clients, to exchange messages asynchronously with other JMS clients by using JMS destinations (queues or topics). A J2EE application can use JMS queue destinations for point-to-point messaging and JMS topic destinations for publish/subscribe messaging. A J2EE application can explicitly poll for messages on a destination then retrieve messages for processing by business logic beans.

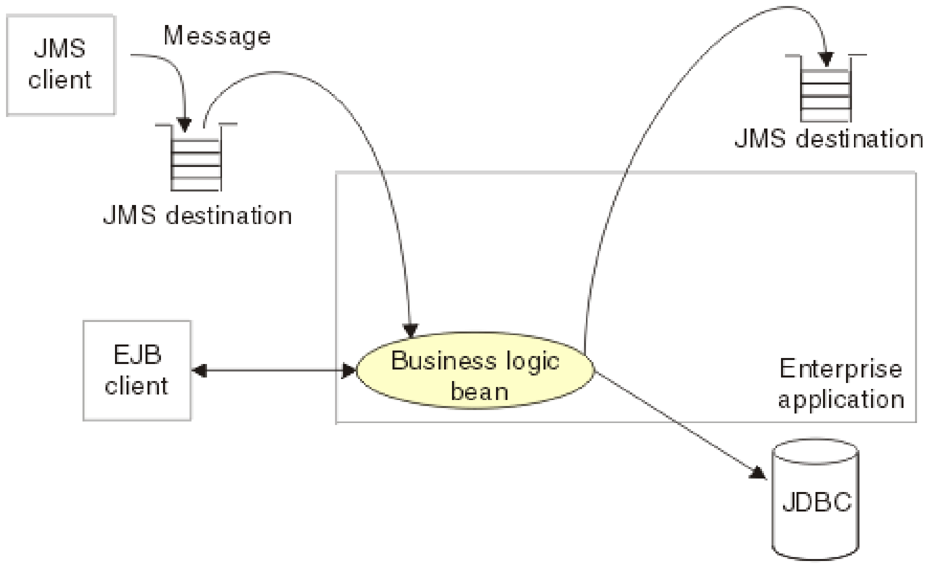


Figure 7-1 Asynchronous messaging using JMS

Figure 7-1 shows an application polling a JMS destination to retrieve an incoming message, which it processes with a business logic bean. The business logic bean uses standard JMS calls to process the message. For example, to extract data or to send the message on to another JMS destination.

Author Comment: JMS/XA support (below) is not defined ..

With the base JMS/XA support, the J2EE application uses standard JMS calls to process messages, including any responses or outbound messaging. Responses can be handled by an enterprise bean acting as a sender bean, or handled in the enterprise bean that receives the incoming messages. Optionally, this process can use two-phase commit within the scope of a transaction. This level of functionality for asynchronous messaging is called bean-managed messaging, and gives an enterprise bean complete control over the messaging infrastructure. For example, for connection and session pool management. The common container has no role in bean-managed messaging.

For more information on asynchronous messaging using JMS, see Section 7.1.3, “Java Message Service” on page 151.

Message driven beans

Application server products compliant with the J2EE 1.3 specification provide support for automatic asynchronous messaging using message driven beans (MDBs), a type of enterprise bean defined in the EJB 2.0 specification, and JMS listeners, part of the JMS application server facilities (ASF). Messages are automatically retrieved from JMS destinations, optionally within a transaction, then sent to the MDB in a J2EE application, without the application having to explicitly poll JMS destinations.

For more information regarding asynchronous messaging with MDBs, see Section 7.1.4, “Message driven beans” on page 155.

Enterprise messaging

J2EE 1.3 compliant application server products may provide J2EE applications with another level of functionality for asynchronous messaging called *extended messaging*. The common container manages the messaging infrastructure, and extra standard types of messaging beans are provided to add functionality to that provided by MDBs. This level of functionality enables application developers to concentrate on the business logic to be implemented by the enterprise beans and to leave the messaging usage to standard messaging objects and configuration of the common container.

Author Comment: Is it “enterprise messaging” or “extended messaging”? What does “common container” mean?

Note: Further explanation of *Enterprise Messaging* is beyond the scope of this Redbook. For further details, see `http://www-3.ibm.com/software/webservers/appserv/infocenter.html`.

7.1.2 Application usage

Applications can use the following styles of asynchronous messaging:

- Point-to-point

  Point-to-point applications use queues to pass messages between each other. The applications are called point-to-point, because a client sends a message to a specific queue and the message is picked up and processed by a server listening to that queue. It is common for a client to have all its messages delivered to one queue. A queue can contain a mixture of messages of different types.

- Publish/subscribe

  Publish/subscribe systems provide named collection points for messages, called topics. To send messages, applications publish messages to topics. To receive messages, applications subscribe to topics; when a message is published to a topic, it is automatically sent to all the applications that are subscribers of that topic. By using a topic as an intermediary, message publishers are kept independent of subscribers.

Note: Both styles of messaging can be used in the same application.

Applications can use asynchronous messaging in the following ways:

- One-way

  An application sends a message, and does not want a response. This pattern of use is often referred to as a datagram.

- Request / response

  An application sends a request to another application and expects to receive a response in return.

- One-way and forward

An application sends a request to another application, which sends a message to yet another application.

Note: These messaging techniques can be combined to produce a variety of asynchronous messaging scenarios.

7.1.3 Java Message Service

The Java Message 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.

JMS supports both the point-to-point and publish/subscribe messaging models, described in Section 7.1.2, “Application usage” on page 150.

Note: An application server product compliant with the J2EE 1.3 specification is required to provide support for both the JMS API and a built-in JMS provider implementation. WebSphere 4.0 only provided support for the JMS API. A JMS provider implementation had to be installed separately.

Components

The main components of the Java Message Service are shown in Figure 7-2, from the JMS provider, through a connection to a destination, then to an application (acting as a JMS client) that processes the message retrieved from the destination.

Author Comment: Need to draw / or redraw the following. It came from the infocenter. Should at least connect the JMS provider to the JMS destinations.

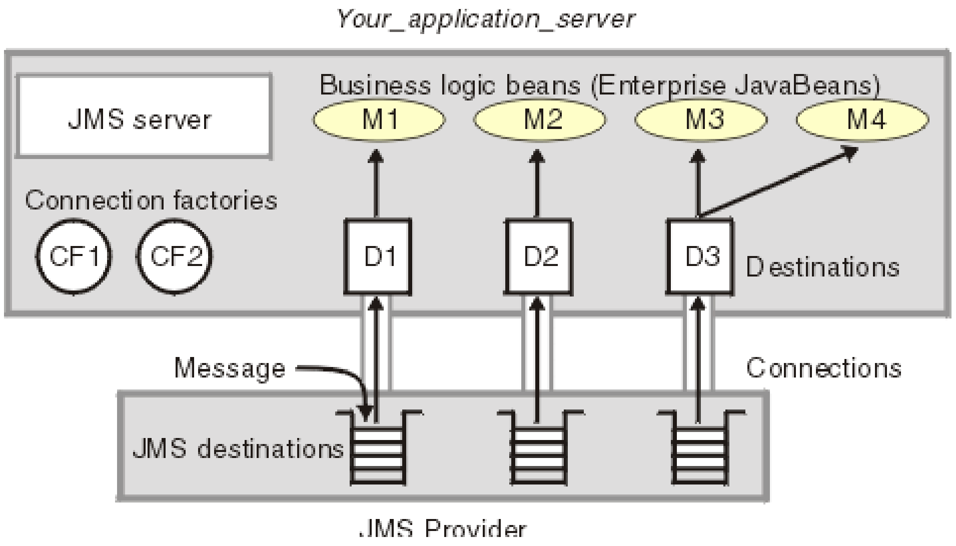


Figure 7-2 JMS components

JMS provider

A JMS provider is a base messaging system and related Java classes that implement the JMS API. The resources of the JMS provider are accessed through JMS connection factory and destination objects.

JMS server

The JMS functions of the JMS provider are accessed by the JMS server within the application server. Each JMS server is responsible for managing a WebSphere and a broker:

- Queue manager

  The queue manager is the provider of the point-to-point (queue) service.

- Broker

  The broker is the provider of the publish/subscribe (topic) service.

JMS administered objects

In JMS certain objects are created by the administrator and stored in a directory.

The Java Naming and Directory Interface (JNDI) namespace is used to hold references to JMS administered objects, encapsulates settings necessary for connecting to and using the Queues and/or Topics of messaging systems.

The JMS administered objects are:

- ***JMS connection factories***

  A connection factory is used to create connections with the JMS provider for a specific JMS queue or topic destination. Each connection factory encapsulates the configuration parameters needed to create a connection to a JMS destination.

  Storing the connection details in JNDI makes the application connection code vendor independent. There are two types of JMS connection factory:

  – QueueConnectionFactory

    Encapsulates the settings necessary to connect to a queue-based messaging system.

  – TopicConnectionFactory

    Encapsulates the settings necessary to connect to a topic-based messaging system.

- ***JMS destinations***

  A JMS destination provides a specific endpoint for messages. A J2EE application can explicitly poll for messages on a destination then retrieve messages for processing by business logic beans (enterprise beans). Destination objects are references to JMS queues and topics. Storing these objects in JNDI separates the application from knowledge of the queueing topology or topic space.

  Note: JMS destination objects can also be used to encapsulate vendor-specific properties, without having to expose those properties to the application.

Accessing JNDI with JMS

In order for an application to access the JMS administered objects from the JNDI namespace, a JMS provider resource needs to be configured in the application server. The JMS provider resource encapsulates the following information, necessary for making connection to the context root of the appropriate JNDI namespace:

- InitialContextFactory

  Messaging system specific class that can make a connection to the context root of the appropriate JNDI namespace.

- Provider URL

  A URL that defines the protocol, hostname, port number and root context key necessary to locate the JNDI namespace.

Using J2EE references with JMS

The J2EE platform adds a further level of indirection to JMS administered objects:

- Alias references are used by the application code.
- Aliases are mapped to real JMS administered objects during application deployment.

It is possible to deploy multiple independently developed applications into the same application server, with each application using the same names for JMS queues or topics.

To avoid name collision, J2EE uses the *java:comp/env* mechanism. In this mechanism, each application looks up the name in its own personal (java:comp/env) environment. The administrator can define different resolutions of the lookup name for each application, and makes the container responsible for returning the correct references from JNDI for each local lookup.

Example 7-1 shows an example of accessing the JMS queue connection factory and destination using the J2EE local application (java:comp/env) JNDI namespace.

Example 7-1 Using JMS with java:comp/env

```
import javax.jms.*;
import javax.naming.*

// Get the JNDI initial context
InitialContext initCtx = new InitialContext();

// get the queue connection factory
QueueConnectionFactory qcf = (QueueConnectionFactory)
initCtx.lookup(“java:comp/env/jms/myQCF”);

// Create a connection
QueueConnection conn = qcf.createQueueConnection();

// Create a JMS session
QueueSession session = conn.createQueueSession(false,
Session.AUTO_ACKNOWLEDGE);

// get the queue used to send a message
Queue q = (Queue) lookup(“java:comp/env/jms/myQueue”);

// Send a message...
QueueSender sender = session.createSender(q);
sender.send(session.createTextMessage(“Hello world”));
sender.close();
```

For details on how resource reference aliases are bound on application deployment to real JMS administered objects, see Chapter 20, “Deploying an application” on page 835.

7.1.4 Message driven beans

An application server product compliant with the J2EE 1.3 specification will provide support for automatic asynchronous messaging using message driven beans (MDBs), a type of enterprise bean defined in the EJB 2.0 specification.

Note: WebSphere 4.0 does not provide support for MDBs, although the Enterprise Edition provides a “work-alike” called message beans that leverages stateless session EJBs. However the WebSphere 4.0 container does not have these services integrated and does not implement the EJB 2.0 specification.

Components

The major components involved in messaging using MDBs are shown in Figure 7-3.

Author Comment: redraw .. from infocenter

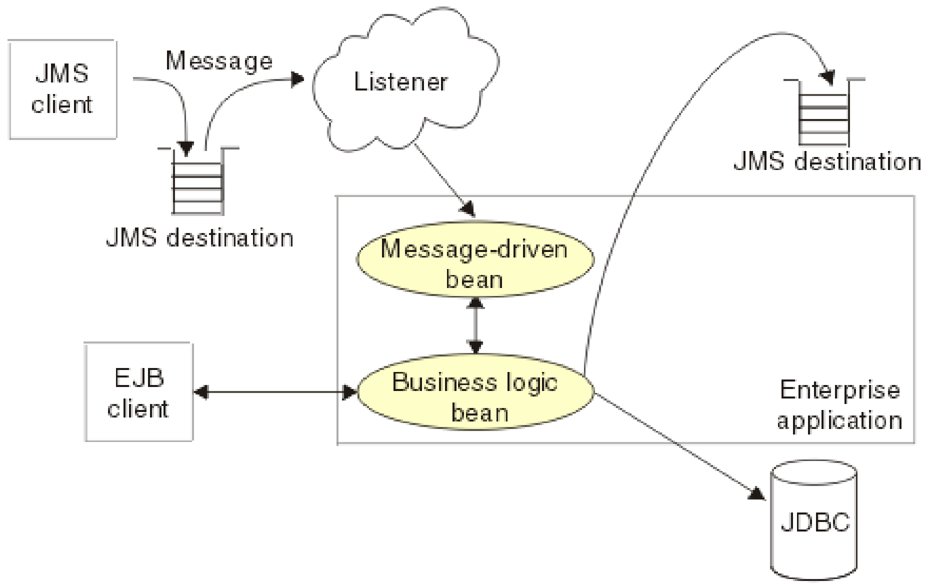


Figure 7-3 Messaging using message driven beans

Note: This figure shows an incoming message being passed by a JMS listener to a MDB, which passes the message on to a business logic bean for business processing. This messaging is controlled by the listener manager.

The support for MDBs is based on the message listener, which comprises the following components:

Listener manager
Controls and monitors one or more listeners.

Listener
Each listener monitors a JMS destination for incoming messages. When a message arrives on the destination, the listener passes the message to a new instance of a user-developed MDB for processing. The listener then looks for the next message without waiting for the bean to return.

JMS destination
Represents a queue or topic.

Message driven beans

A stateless component that is invoked by the J2EE container as a result of the arrival of a JMS message at a particular JMS destination (queue or topic). The MDB is in a sense, triggered by the arrival of the message.

Messages arriving at a destination being processed by a listener have no client credentials associated with them; the messages are anonymous. To enable some level of security, a listener assumes the credentials of the application server process for the invocation of the MDB.

Tip: It is recommended that MDBs be developed so that they delegate the business processing of incoming messages to another enterprise bean, to provide clear separation of message handling and business processing. This also enables the business processing to be invoked by either the arrival of incoming messages or by direct invocation of the EJB.

Transaction support

MDBs can handle messages read from JMS destinations within the scope of a transaction. If transaction handling is specified for a JMS destination, the JMS listener starts a global transaction before it reads any incoming message from that destination. When the MDB processing has finished, the JMS listener commits or rolls back the transaction (using JTA transaction control).

Note:

1. All messages retrieved from a specific destination have the same transactional behavior.
2. If non-WebSphere XA connection factories are used, then the JMS resources are not recovered if the server fails.

7.2 WebSphere support for asynchronous messaging

WebSphere support for asynchronous messaging can be grouped into the following categories:

- Support for J2EE 1.3.
- Support for Java Message Service (JMS).
- Support for message driven beans (MDBs).

7.2.1 WebSphere support for J2EE 1.3

The changes relating to asynchronous messaging in WebSphere 5.0 are brought about by the requirements of the J2EE 1.3 specification, to which WebSphere 5.0 is compliant. In order to be compliant with the specification, an application server product must:

1. Provide an implementation of the JMS Server 1.0.

   In J2EE 1.3, the JMS API becomes an integral part of the J2EE platform.

   WebSphere includes an embedded JMS Server that can be chosen as an option for the base installation. The embedded JMS Server is fully compliant with the JMS 1.0.2 specification.

2. Support the message driven beans programming model, defined in the EJB 2.0 specification.

   WebSphere includes an implementation of MDBs that is fully compliant with the EJB 2.0 specification.

7.2.2 WebSphere support for JMS

WebSphere provides support for JMS in the following areas.

JMS providers

The JMS functionality provided by WebSphere includes support for three types of JMS provider:

- Integral JMS provider (embedded messaging)

  In WebSphere 4.0, JMS support requires the installation of a separate JMS product, e.g.. WebSphere MQ for point-to-point messaging and either WebSphere MQ Integrator or the MA0C product extension for publish/subscribe messaging.

  WebSphere includes an integral JMS provider that is integrated with the product and can be chosen as an installation option.

  Note: There can only be one JMS Server (integral JMS provider) per node.

  For detailed information on the integral JMS provider, see Section 7.3, “Integral JMS provider” on page 166.

- WebSphere MQ JMS provider

  WebSphere 5.0 supports the use of full WebSphere MQ as the JMS provider. The product is tightly integrated with the WebSphere installation, with

WebSphere providing the JMS client classes and administration interface, while WebSphere MQ provides the queue-based messaging system.

For detailed information on the WebSphere MQ JMS provider, see Section 7.4, “WebSphere MQ JMS provider” on page 181.

- Generic JMS providers

  WebSphere 5.0 supports the use of generic JMS providers, as long as they implement the ASF component of the JMS 1.0.2 specification.

  For detailed information on generic JMS providers, see Section 7.5, “Generic JMS provider” on page 186.

Note:

1. There can be more than one JMS Provider per node. That is, a node can be configured to concurrently make use of any combination (or all) of: integral JMS provider, WebSphere MQ JMS provider and Generic JMS provider.
2. It is possible to have both the internal JMS provider and full WebSphere MQ concurrently installed on the same machine. However, there will only be one copy of the JMS client classes (MA88) and MQ transport code (binaries). This has the consequence that both must be at the same version and patch level.

The WebSphere administration clients (wsadmin and Web administration console) provide configuration and/or management of JMS resources for each of these types of providers:

Table 7-1 JMS providers Vs. JMS resources

JMS Provider	**Provider resource**	**Configurable JMS resources**
Integral	WebSphere JMS provider	WebSphere Queue Connection Factories WebSphere Topic Connection Factories WebSphere Queue Destinations WebSphere Topic Destinations
WebSphere MQ	WebSphere MQ JMS provider	WebSphere MQ Queue Connection Factories WebSphere MQ Topic Connection Factories WebSphere MQ Queue Destinations WebSphere MQ Topic Destinations
Generic	Generic JMS provider	Not configurable - can only setup InitialContextFactory to access externally configured resources.

For details on the configuration of each type of JMS provider and their associated JMS connection factory and JMS destination resources, see Chapter 18, “Configuring WebSphere resources” on page 623.

JMS administered objects

The tool used to add the provider’s administered objects to a JNDI namespace is dependent upon the namespace targeted:

- WebSphere namespace

  Generic JMS provider would need to provide an administration tool that can access the WebSphere namespace. The JMS objects of the Integral and WebSphere MQ JMS providers can be administered using the WebSphere administration tools.

- External namespace

  Integral and WebSphere MQ resources need to be registered with the WebSphere namespace. Generic JMS providers can register their JMS administered objects in external namespaces as long as their tool has support for that namespace.

JNDI namespace

The JNDI namespace can be either:

- The federated namespace of WebSphere.
- An external JNDI namespace.

In order for an application hosted by WebSphere to access the JMS administered objects from the JNDI namespace, requires the JMS provider resource to be configured in WebSphere. The JMS provider resource encapsulates the information shown in Table 7-2, necessary for making connection to the context root of the appropriate JNDI namespace.

Table 7-2 JMS Provider settings

Setting	Integral JMS provider	WebSphere MQ JMS provider	Generic JMS provider
External Initial Context Factory	Not configurable	Not configurable	Configurable
External Provider URL	Not configurable	Not configurable	Configurable
Classpath	Not configurable	Not configurable	Configurable

Setting	Integral JMS provider	WebSphere MQ JMS provider	Generic JMS provider
Native library path	Not configurable	Not configurable	Configurable

As shown in Figure 7-4, the WebSphere 5.0 support for generic JMS providers works by registered "alias" entries, in the application server's local namespace, that refer to the real JMS administered objects (JMS connection factory and JMS destination) in the external namespace.

Jeff: Check this pic ..should we label the Namespace on the app server "local namespace" and the namespace on the generic JMS provider "external namespace"? Also, I couldn't quite make out the word under the queue .. was it "broker"? . Shows the objects created by configuration of a generic JMS provider: the local JNDI entries, their references to the external namespace, the external provider's real objects etc....

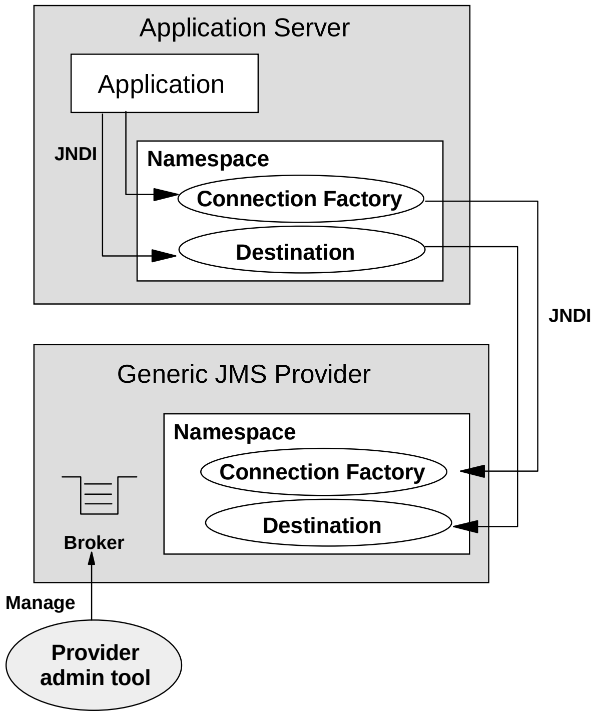


Figure 7-4 Generic JMS provider components

See Chapter 18, “Configuring WebSphere resources” on page 623 for details on the configuration of each type of JMS provider and its JMS resources.

JMS transaction integration

In order for JMS operations to be part of a global transaction, the container must be made aware of the JMS session used by the operations. Unfortunately, the J2EE 1.3 specification does not define how JMS should be enlisted into a container’s global transactions.

WebSphere 5.0 uses the J2C Connection Manager runtime to provide the container services necessary to support transactions.

Note: WebSphere 4.0 implements a solution to this problem specifically for WebSphere MQ by wrappering the MQ ConnectionFactory, Connection and Session objects with WebSphere-specific versions that are stored in JNDI in place of the formal WebSphere MQ objects. However, if the non-WebSphere specific JMS classes were used, then any transactions in an application’s JMS code could not be handled by the WebSphere 4.0 container, preventing the application from participating in global (two-phase commit) transactions.

JMS resource pooling

In WebSphere 5.0, the JMS Connection and session resources are managed by the J2C Connection Manager. The advantage of this approach is that it enables the integration of container services such as transaction and security.

JMS administration tools

The support provided by WebSphere administration tools for configuration of JMS providers differs depending upon the provider. Table 7-3 provides a summary of the support.

Author Comment: For those settings that are integrated into was but not exposed in the console .. need to make sure we say how to set them somewhere and index it.

Table 7-3 WebSphere administration support for JMS provider configuration

Configurable objects	Integral JMS provider	WebSphere MQ JMS provider	Generic JMS provider
Initial context factory and provider URL	N **	N **	Y

Configurable objects	**Integral JMS provider**	**WebSphere MQ JMS provider**	**Generic JMS provider**
Messaging system objects (queues / topics)	Y	N	N
JMS administered objects (JMS connection factory and JMS destination)	Y	Y	N
** The settings are not exposed in the Web administration console.			

The key points are:

- WebSphere provides functionality so that JMS administered objects for both the integral JMS provider and WebSphere MQ JMS provider can be configured using the WebSphere administration tools. There is no need to use the JMSAdmin tool, provided with the WebSphere MQ MA88 JMS extension, to administer WebSphere MQ JMS provider resources.
- The WebSphere administration tools can be used to configure and manage the actual queue manager, queues and topics of the internal JMS provider only.
- For the WebSphere MQ and generic JMS providers, the messaging system and its resources must be configured using the provider's native tools. The WebSphere administration tools do not provide this functionality.

For further details on the use of WebSphere administration tools for administering JMS resources, see Chapter 18, “Configuring WebSphere resources” on page 623.

7.2.3 WebSphere support for message driven beans

WebSphere provides support for MDBs in the following areas.

MDB components

The IBM WebSphere Application Server support for MDBs is based on JMS message listeners and the message listener service, and builds on the base support for JMS described in Section 7.1.4, “Message driven beans” on page 155.

The main components of WebSphere support for MDBs are shown in Figure 7-5, from the JMS provider through a connection to a destination, listener port, then deployed MDB that processes the message retrieved from the destination.

Author Comment: Need to redraw .. from infocenter

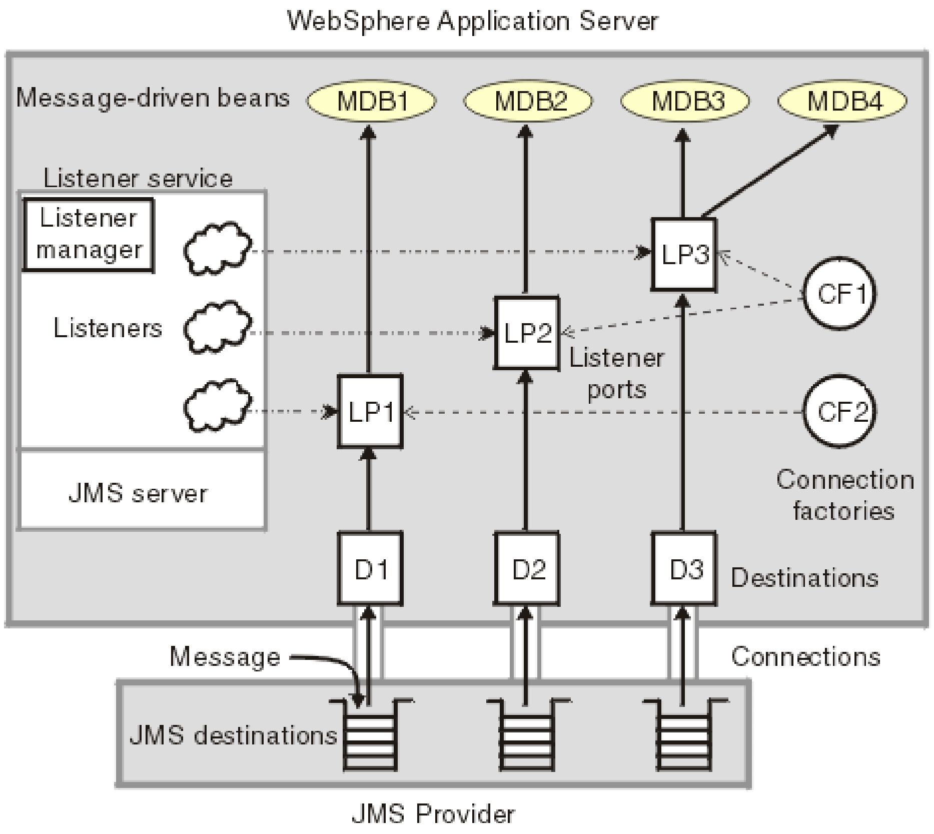


Figure 7-5 Components of WebSphere MDB implementation

Message listener service

An extension to the JMS functions of the JMS provider provides a listener manager, which controls and monitors one or more JMS listeners.

Listener

Each listener monitors either a JMS queue destination (for point-to-point messaging) or a JMS topic destination (for publish/subscribe messaging).

The listener is the part of the application server that invokes the MDB when the message arrives. It is fully integrated with the EJB container.

Listener recovery from messaging domain failures is new to WebSphere 5.0. For example, if the external queue manager fails in WebSphere 4.0, the listener stops and must be manually restarted. In WebSphere 5.0, the listener automatically recovers in such scenarios.

Each listener completes several steps for the JMS destination that it is to monitor, including:

- Creating a JMS Server session pool, and allocating JMS server sessions and session threads for incoming messages.
- Interfacing with JMS ASF to create JMS connection consumers to listen for incoming messages.
- If specified, starting a transaction and requesting that it is committed (or rolled back) when the EJB method has completed.
- Processing incoming messages by invoking the onMessage() method of the specified MDB.

Connection factory

Used to create connections with the JMS provider for a specific JMS queue or topic destination. Each connection factory encapsulates the configuration parameters needed to create a connection to a JMS destination.

Listener port

Defines the association between a connection factory, a destination, and a deployed MDB. Listener ports are used to simplify the administration of the associations between these resources.

MDB resource configuration

When a deployed MDB is installed, it is associated with a listener port and the listener for a destination. When a message arrives on the destination, the listener passes the message to a new instance of a MDB for processing.

When an application server is started, it initializes the listener manager based on the configuration data. The listener manager creates a dynamic session thread pool for use by listeners, creates and starts listeners, and during server termination controls the cleanup of listener message service resources. The association between Destinations, Connection factories, Listener Ports and MDBs is shown in Figure 7-6.

Author Comment: The following figure is from pp17 of the JMS.pdf course material. It doesn’t really mirror the text. Need to look at expanding the text or changing the pic.

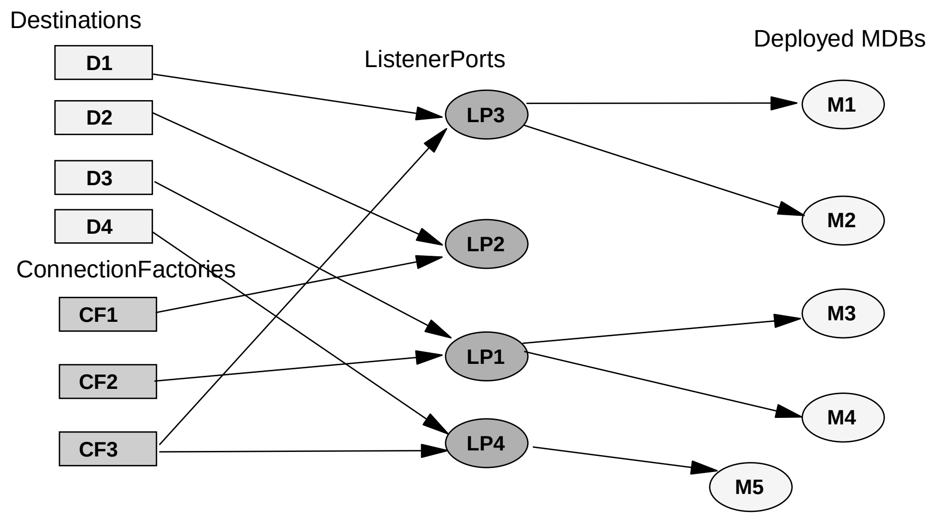


Figure 7-6 Configuration of WebSphere MDB resources

MDB administration tools

The configuration of MDBs is fully integrated with the WebSphere administration tools. For details on the configuration of MDB resources, see Chapter 18, “Configuring WebSphere resources” on page 623.

Migration of Websphere 4.0 message beans

WebSphere provides the mb2mdb script for use in migrating WebSphere 4.0 Message Beans to fully compliant MDBs. In addition, the samples provided with WebSphere have been migrated to use MDBs.

For details on the mb2mdb script and its use, see .. need reference here

7.3 Integral JMS provider

WebSphere includes an embedded JMS provider called the Integral JMS provider. This provider is composed of the following IBM products:

1. WebSphere MQ
2. WebSphere MQ MA88 SupportPac - JMS client classes.

3. Greyhound broker - an early release of WebSphere Event Broker, providing the publish/subscribe service.

Note:

1. The version of each of these products that is integrated into the integral JMS provider is reduced in footprint and function compared to the independently available product.
2. For details of the versions of each product supplied, see Chapter 18, “Configuring WebSphere resources” on page 623.

7.3.1 Recommended use

The integral JMS provider is recommended for use in the following scenarios:

- When the functionality of full WebSphere MQ is not required

  This will simplify configuration and management of the messaging functionality.

- When first introducing messaging into applications

  The choice of using the integral JMS provider does not prevent full WebSphere MQ being installed, configured and used as the JMS provider at a later date as operators become more familiar with messaging. Reinstallation of WebSphere is not required.

Note: The use of the integral JMS provider does not change existing application usage scenarios for applications using JMS.

7.3.2 Key features

There are several features that are common to both the point-to-point and publish/subscribe functions:

- The integral JMS provider supports the requirements of the J2EE 1.3 specification:
  - Support for the JMS 1.0.2 specification.
  - Providing an implementation of JMS Server 1.0.
  - Full compliance with the J2EE 1.3 compliance tests.
- The integral JMS provider does not support:
  - APIs other than those defined in the JMS API.
  - Interoperability with WebSphere MQ queue managers.
- Security is integrated with the WebSphere security system, with user and group access being configured using the WebSphere administration tools.

- The JMS Server hosts the broker and manages the external WebSphere MQ processes.
- All JMS client access, either point-to-point or publish/subscribe is performed using WebSphere MQ client (TCP/IP) mode.

> **Note:** This is a consequence of the use of a SVRCONN channel for all client access. The SVRCONN channel allows a security exit to be configured to link the WebSphere MQ native process into the WebSphere security system. The JMS Server provides a security listener port (when WebSphere global security is enabled) that is called by the security exit to pass security requests through to WebSphere.

- Tracing is integrated with the WebSphere tracing infrastructure, with tracing configuration performed via the WebSphere administration tools.

Point-to-point messaging

The point-to-point functionality of the integral JMS provider is provided by a built-in WebSphere MQ server, providing a WebSphere MQ queue manager and queues.

All point-to-point JMS clients access the queue manager listener port directly, using WebSphere MQ client (TCP/IP) mode. The JMS Server is not part of the messaging communication.

Publish/subscribe messaging

The publish/subscribe functionality of the integral JMS provider includes two different access mechanisms:

- Direct

  Used for direct access to the greyhound broker, bypassing any queueing mechanisms in order to provide high performance.

  It does not provide full function publish/subscribe , with it providing only non-persistent, non-durable, non-transactional subscriptions.

- Queued

  Used for full function publish/subscribe - providing support for durable, persistent, transactional subscriptions.

  This mechanism accesses the broker via the queue manager, which is configured to support the broker.

> **Note:** MDBs can only be used with topics that use the QUEUED access mechanism.

7.3.3 Limitations

The major limitations of the integral JMS provider are:

1. It is not possible to exchange messages with queue managers outside WebSphere.

   Note: This limitation does not preclude remote JMS clients from accessing the internal JMS provider of a WebSphere 5.0 node. It only prevents the queue manager of the internal JMS provider from exchanging messages with other queue managers, either within the same cell (administrative domain) or outside the cell.

2. Limited publish/subscribe broker capabilities, including:
   – No message flows.
   – No message transformation.
   – No database prerequisite.
   – No shared topic spaces.

Author Comment: “No database prerequisite” doesn’t sound like a limitation. Should this say “no database support”?

3. Queue manager administration and configuration is not provided by the WebSphere administration tools.

   Administration of the queue manager and channels requires the use of the createmq and deletemq scripts provided with the WebSphere installation. See Section 12.6, “System management tools” on page 435 for details on these scripts.

Author Comment: check later and make sure this is still true.

4. It is pre-installed onto each node, but none of its attributes are updateable via the WebSphere administration tools.
5. No mechanism is provided for sending test messages to queues.
6. No mechanism is provided for clearing messages from queues.
7. Queue manager clustering is not supported.

8. No support for JMS Server failover.

Author Comment: It isn't clear to me how many queue managers you can have ..

7.3.4 Runtime components

The integral JMS provider is fully integrated with the WebSphere runtime. The provider runs in one of two modes (base or network deployment), depending upon whether the node has been added to a Network Deployment cell (administrative domain).

Base configuration

Figure 7-7 shows the internal messaging components present in a base WebSphere configuration, that is, an application server that has not been added to a Network Deployment cell.

Jeff .. check the figure: Need a figure here showing the components (and component interactions) for the integral JMS provider in the base configuration, i.e.. runs in the application server JVM.

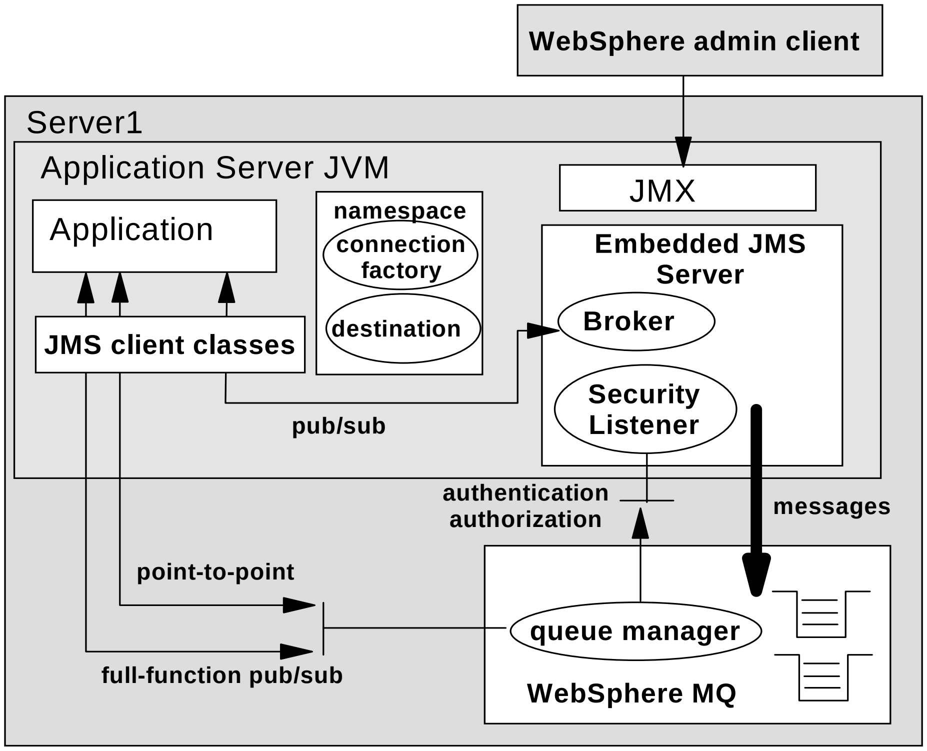


Figure 7-7 Base configuration internal JMS Server components

The key points of the base configuration are:

1. The JMS Server runs embedded in the application server JVM, with the broker hosted within the JVM and the queue-based messaging system running as native WebSphere MQ processes, separate to the JVM.
2. The JMS Server manages the external WebSphere MQ processes, providing management functions to:
   – Start and stop the queue manager.
   – Manage the queue manager, e.g.. define new queues.
   – Provide a security listener port that the WebSphere MQ security exit uses to connect to the WebSphere security system. This is used to provide authentication and authorization checks when WebSphere security is enabled.
3. The security port is defined in the application server's server definition (server.xml) file:

```
<WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/servers/<appserver>/ser
ver.xml
```

> **Note:** Since the WebSphere MQ processes are native, running outside the WebSphere application server JVM, they do not have access to the WebSphere security system. The embedded messaging component provides the listener in order to forward security checks to WebSphere on behalf of WebSphere MQ.

4. Point-to-point messaging involves a client using the JMS client classes (MA88) to make a client mode (TCP/IP) request directly to the local internal WebSphere MQ queue manager. The internal JMS provider does not use bind mode, even though the WebSphere MQ server is on the same machine (local).

   The port number of the queue manager listener port is defined by the JMSSERVER_QUEUED_ADDRESS entry in the following configuration file:

   ```
   <WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml
   ```

Example 7-2 Definition of JMS Server queue manager listener port

```
<specialEndpoints xmi:id="NamedEndPoint_14"
endPointName="JMSSERVER_QUEUED_ADDRESS">
        <endPoint xmi:id="EndPoint_17" host="mka0kkcf" port="5558"/>
</specialEndpoints>
```

5. The default publish/subscribe messaging involves a client using the JMS client classes (MA88) to make a direct call on the greyhound broker running within the application server JVM. The broker listens on a TCP/IP port for incoming direct requests.

   The port number of the broker direct port is defined by the JMSSERVER_DIRECT_ADDRESS entry in the following configuration file:

   ```
   <WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml
   ```

Example 7-3 Definition of JMS Server broker direct port

```
<specialEndpoints xmi:id="NamedEndPoint_13"
endPointName="JMSSERVER_DIRECT_ADDRESS">
        <endPoint xmi:id="EndPoint_16" host="mka0kkcf" port="5559"/>
</specialEndpoints>
```

Note: Full function publish/subscribe messaging involves a client using the JMS client classes (MA88) to make a client mode (TCP/IP) publishing request directly to the local internal WebSphere MQ queue manager, rather than directly accessing the broker. The internal queue manager is configured to use the broker to handle any publish requests it receives.

6. All administration and management of the embedded JMS Server, broker and WebSphere MQ is performed via the WebSphere administration tools.

   The WebSphere administration tools access the embedded JMS Server by connecting to the JMX management port(s) of the application server, defined in the file:

   `<WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml`

Example 7-4 Definition of JMS Server JMX administration port

```
<specialEndpoints xmi:id="NamedEndPoint_18"
endPointName="SOAP_CONNECTOR_ADDRESS">
       <endPoint xmi:id="EndPoint_18" host="mka0kkcf" port="8876"/>
</specialEndpoints>
```

7. All JMS administered objects are registered in the application server's local JNDI namespace, using the WebSphere administration tools.

   The definitions of the JMS administered objects are stored in the resources definition file (resources.xml) at the scope (cell, node or managed server) at which it was created. For further details on the location and content of the *resources.xml* files, see Chapter 12, “System Management” on page 405.

   Note:

   - If a JMS resource is defined at the cell scope, it will be available to each application server managed in that cell. Each application server will separately configure that resource and register it in its local application server namespace.
   - If a JMS resource is defined at the node scope, it will be available to each application server managed by that node. Each application server will separately configure that resource and register it in its local application server namespace.

   See Chapter 18, “Configuring WebSphere resources” on page 623 for details on the configuration of these objects.

8. All WebSphere MQ queues are created, updated or deleted using the WebSphere administration tools. The standard WebSphere MQ tools are not available for this task

Author Comment: are deletemq, etc. considered WebSphere administration tools? I would think this is OK .. need a more specific reference below.

See Chapter 18, “Configuring WebSphere resources” on page 623 for details on the configuration of these WebSphere MQ based messaging objects.

Network Deployment configuration

Figure 7-8 shows the internal messaging components present in a Network Deployment node configuration, that is, an application server that has been added to a Network Deployment cell.

Author Comment: Need a figure here showing the components (and component interactions) for the integral JMS provider in the ND configuration, i.e.. runs in a standalone managed server JVM.

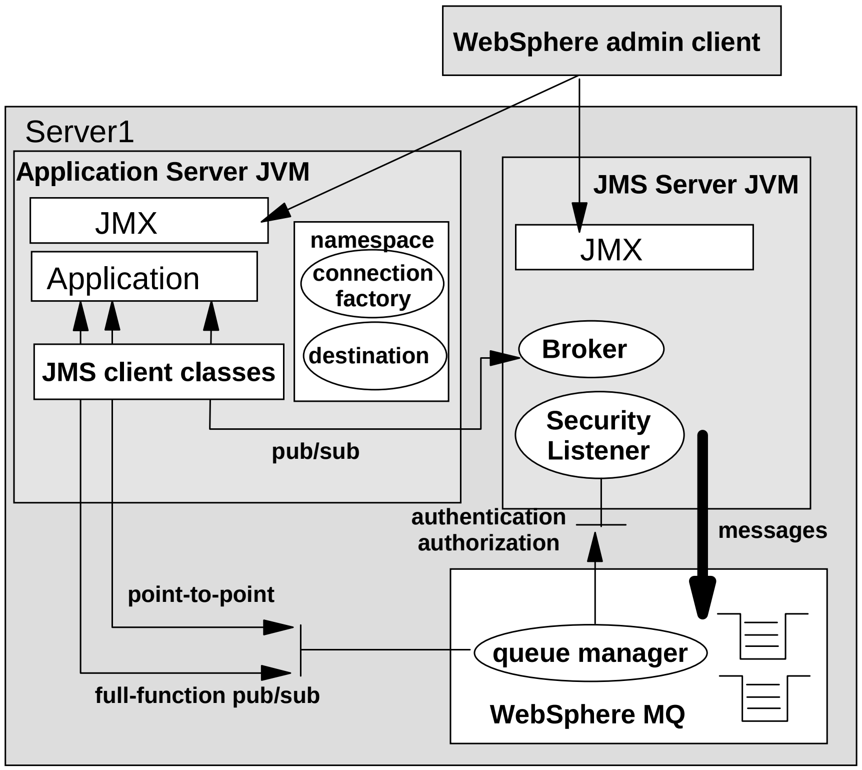


Figure 7-8 Network Deployment configuration internal JMS Server components

The key points of the Network Deployment configuration are:

1. The JMS Server runs in its own dedicated JVM (jmsserver) that is created as part of adding a node to a cell. The jmsserver is a WebSphere 5.0 managed process, having its own server definition (server.xml) in the node's configuration directory, as well as dedicated IP ports defined in the node's serverindex.xml.

   See Chapter 12, "System Management" on page 405 for details on server definitions and configuration files.

2. The *jmsserver* managed process is a specialized server instance, as its configuration does not include:

   – A Web container
   – An EJB container
   – A name service
   – An Object Request Broker (ORB)

Note:

1. There can only be one JMS Server (integral JMS provider) per node.
2. There can be more than one JMS Provider per node. That is, a node can be configured to concurrently make use of any combination (or all) of: integral JMS provider, WebSphere MQ JMS provider and Generic JMS provider.

3. The JMS Server on a node is independent of the JMS Servers on all other nodes in a cell.
4. The dedicated *jmsserver* process manages the external WebSphere MQ processes, providing management functions to:
   – Start and stop the queue manager.
   – Manage the queue manager, for example, to define new queues.
5. Provide a security listener port that the WebSphere MQ security exit uses to connect to the WebSphere security system. This is used to provide authentication and authorization checks when WebSphere security is enabled.

   The security port is defined in the jmsserver's server definition (server.xml) file:

   ```
   <WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/servers/jmsserver/serve
   r.xml
   ```

   Note: Since the WebSphere MQ processes are native, running outside the WebSphere application server JVM, they do not have access to the WebSphere security system. The embedded messaging component provides the listener in order to forward security checks to WebSphere on behalf of WebSphere MQ

6. Point-to-point messaging involves a client using the JMS client classes (MA88) to make a *client* mode (TCP/IP) request directly to the local WebSphere MQ queue manager. The JMS Server does not use *bind* mode, even though the WebSphere MQ server is on the same machine (local).

   The port number of the queue manager listener port is defined by the JMS_QUEUED_PORT entry in the following configuration file:

   ```
   <WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml
   ```
7. The default high performance publish/subscribe messaging involves a client using the JMS client classes to make a direct call on the greyhound broker running within the JMS Server JVM. The broker listens on a TCP/IP port for incoming direct requests.

The port number of the broker direct port is defined by the JMS_DIRECT_PORT entry in the following configuration file:

```
<WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml
```

Note: Full function publish/subscribe messaging involves a client using the JMS client classes (MA88) to make a client mode (TCP/IP) publishing request directly to the local internal WebSphere MQ queue manager, rather than direct accessing the broker. The internal queue manager is configured to use the broker to handle any publish requests it receives.

8. All administration and management of the JMS Server, broker and WebSphere MQ is performed via the WebSphere administration tools.

   The WebSphere administration tools access the JMS Server process by connecting to its JMX SOAP management port, defined in the configuration file:

   ```
   <WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml
   ```

9. All JMS administered objects are registered, using the WebSphere administration tools, in the local JNDI namespace of each application server hosting an application that uses the JMS Server.

   JMS resources (queues and topics) defined in a particular node's JMS Server are accessible from anywhere in the cell.

   The definitions of the JMS administered objects are stored in the resources definition file (*resources.xml*) at the scope (cell, node or managed server) at which it was created. For further details on the location and content of the *resources.xml* files, see Chapter 12, "System Management" on page 405.

Author Comment: Need more specific refs

   See Chapter 18, "Configuring WebSphere resources" on page 623 for details on the configuration of these JMS resources.

10. All WebSphere MQ queues are created, updated or deleted using the WebSphere administration tools. The standard WebSphere MQ tools are not available for this task.

    See Chapter 18, "Configuring WebSphere resources" on page 623 for details on the configuration of these WebSphere MQ based messaging objects.

7.3.5 Administration

The integral JMS provider is fully integrated with the WebSphere administration. In fact, the administration tools that come standard with WebSphere MQ are not provided with the WebSphere MQ included with the integral JMS provider.

All administration must be performed using the WebSphere administration tools. For further details see Chapter 18, “Configuring WebSphere resources” on page 623.

Note: The JMS Server is installed using default WebSphere MQ settings which may not meet high load requirements. If required, direct editing of the MQ settings could be performed to adjust the JMS Server MQ performance. See *WebSphere MQ System Administration Guide*, SC33-1873-02 for further details.

Author Comment: More specific reference .. where do we directly edit the MQ settings as mentioned in the note above? Make sure we know how to do this at the correct scope so they don’t get wiped out in ND

7.3.6 Security considerations

Security for the WebSphere internal JMS provider operates as a part of the WebSphere global security, and is enabled only when global security is enabled.

Authentication

When security for the WebSphere internal JMS provider is enabled, JMS connections made to the JMS provider are authenticated and access to JMS resources owned by the JMS provider is by controlled by access authorizations.

All JMS connections to the JMS provider must provide a user ID and password for authentication. The user ID and password do not need to be provided by the application. The user ID and password provided are authenticated using standard authentication procedures. If authentication is successful then the JMS connection is allowed to continue; if the authentication fails then the connection is ended.

Author Comment: Need more info here .. or point to the place where we talk about it. Can it use LDAP? Is it integrated with the rest of WAS security? Same info needs to be in the other security sections also.

Authorization

Authorization to access JMS resources owned by the internal JMS provider is controlled by authorization settings in the *integral-jms-authorizations.xml* file located under <WAS_ROOT>/config/cells/<cellname>. The settings are summarized in Table 7-4.

Note: The authorization settings are not configurable nor viewable via the WebSphere administration tools. The only mechanism currently provided is to manually edit the appropriate authorizations file.

Author Comment: Check at GM to see if this is now configurable via the tools.

Table 7-4 Authorization rights for JMS destinations

Object	Authorization rights
Queue	Read Write
Topic	Publish Subscribe Persist

For cell-wide authorizations, edit the file in the cell directory:

```
<WAS_ROOT>/config/cells/<cellname>/integral-jms-authorizations.xml
```

For node-wide authorizations, copy the cell-wide file into the node directory and edit its settings as required:

```
<WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/integral-jms-authorizations
.xml
```

An example is shown in Example 7-5.

Author Comment: Is the file name .integral-jms-authorizations.xml or integral-jms-authorisations.xml (s or z) .. do a global change on text & index markers.

Example 7-5 integral-jms-authorizations.xml file

```
<integral-jms-authorizations>
  <dynamic-update>true</dynamic-update>
  <queue-admin-userids>
    <userid>adminid1</userid>
    <userid>adminid2</userid>
  </queue-admin-userids>

  <queue-default-permissions>
    <permission>write</permission>
  </queue-default-permissions>

  <queue>
    <name>q1</name>
    <public>
    </public>
    <authorize>
      <userid>useridr</userid>
      <permission>read</permission>
    </authorize>
    <authorize>
      <userid>useridw</userid>
      <permission>write</permission>
    </authorize>
  </queue>

  <topic>
    <name>a/b/c</name>
    <public>
      <permission>+sub</permission>
    </public>
    <authorize>
      <userid>useridpub</userid>
      <permission>+pub</permission>
    </authorize>
  </topic>
</integral-jms-authorizations>
```

For further details on the use and content of the *integral-jms-authorizations.xml* file, see the documentation comments included at the top of the file.

Author Comment: Need a reference here to the WebSphere 5.0 Security book.

7.3.7 Interoperability

The integral JMS provider does not interoperate with other WebSphere MQ queue managers. As such, it cannot be used for heterogeneous WebSphere MQ environments.

The integral JMS provider is able to interoperate with the following clients:

1. JSPs, servlets and EJBs running in WebSphere 5.0

   Any application server based client can directly use any JMS Server in the cell, simply by using the JMS connection factory and JMS destination associated with the chosen JMS Server.

2. WebSphere 5.0 J2EE application clients

   Any J2EE application client can use any JMS Server in the cell, simply by:

   – Using the JMS connection factory and JMS destination associated with the chosen JMS Server.

   – Using the JMS client classes (MA88) delivered with WebSphere 5.0.

3. WebSphere 4.0 clients

   A client hosted in a WebSphere 4.0 environment can make use of the integral JMS provider, as long as:

   – WebSphere 4.0 is at the correct patch level for JNDI interoperability with WebSphere 5.0.

   – WebSphere 4.0 has been upgraded to use a version of the JMS client classes compatible with WebSphere 5.0.

   – The JMS administered objects have configured bindings (aliases) created by the administrator in the WebSphere 5.0 legacy namespace root.

   See Chapter 6, “Naming” on page 105 for further details on WebSphere 4.0 client interoperability with WebSphere 5.0.

Author Comment: If you have 4.0 clients working with the 5.0 integral client, they don’t have to know the queue names? They use JNDI? Not sure how this works.

7.4 WebSphere MQ JMS provider

In order to use the WebSphere MQ JMS provider as an external JMS provider with WebSphere, the following components must be installed, in addition to the WebSphere installation:

1. WebSphere MQ
2. WebSphere MQ MA88 SupportPac - JMS classes.
3. Broker for publish/subscribe messaging. There are three alternatives that will work with WebSphere:

   – WebSphere MQ Publish and Subscribe

- WebSphere MQ Integrator
- WebSphere MQ MA0C SupportPac - Publish/Subscribe

Author Comments:
1. Check to make sure this is the name of MA0C. ..
2. Jeff, & in the note below, is the first note supposed to say "integral JMS provider" or "WebSphere MQ JMS provider"?
3. Note 2 below needs to be more specific

.

Note:

1. The version of each of these products that is integrated into the integral JMS provider is reduced in footprint and function compared to the independently available product.
2. For details of the versions of each product supplied, see Chapter 18, "Configuring WebSphere resources" on page 623.
3. The use of MA0C is discouraged, as the other brokers provide a much more robust production publish/subscribe environment.

Although WebSphere provides an integral JMS provider based on WebSphere MQ, it is expected that in most instances the WebSphere MQ JMS provider would be the preferred choice. Reasons for this are:

1. The integral JMS provider can only be used from within a WebSphere environment. However, many production environment use WebSphere MQ for heterogeneous integration, allowing J2EE applications to communicate with other MQ applications that may be written in languages other than Java and/or hosted in environments other than WebSphere.
2. Taking advantage of the advanced messaging topologies supported by WebSphere MQ. The topologies include:
   - Queue manager clustering

     Supports higher throughput levels and continuous availability for new messages, e.g.. those messages that trigger MDBs.
   - High availability (HA) technologies

     Use shared disks to enable both the messaging service and individual in-flight messages to be highly available. Such technologies include: HACMP on AIX, Microsoft Cluster Server on Windows and SunCluster on Solaris.

   - Broker collectives

     This uses multiple copies of WebSphere MQ Integrator.
   - z/OS shared queues

     High availability messaging.
3. Customers currently using full WebSphere MQ will be more familiar with its issues and management. There is no need to become familiar with a custom form of WebSphere MQ.

Author Comment: Should we mention MQI here? I would assume you would not be able to take advantage of MQI with the internal providers.

7.4.1 Key features

The WebSphere MQ JMS provider supplies the following features that are not supported by the integral JMS provider of WebSphere 5.0:

1. Full support for the WebSphere MQ API.
2. Support for communication with other WebSphere MQ applications, not just those applications hosted in a WebSphere environment. This includes communication with JMS and non-JMS applications.
3. Support for WebSphere MQ queue manager and channels.
4. Support for WebSphere MQ clusters.
5. Multi-broker capability.
6. Support for integration with existing WebSphere MQ environments.
7. Potential for highly available configurations.

7.4.2 Runtime Components

Jeff: check the pic .. your drawing showed the namespace half in / out of the App server. Is that what you wanted or like shown?

Need a figure here showing the components (and their interactions) for WebSphere 5.0 <--> WebSphere MQ JMS provider.

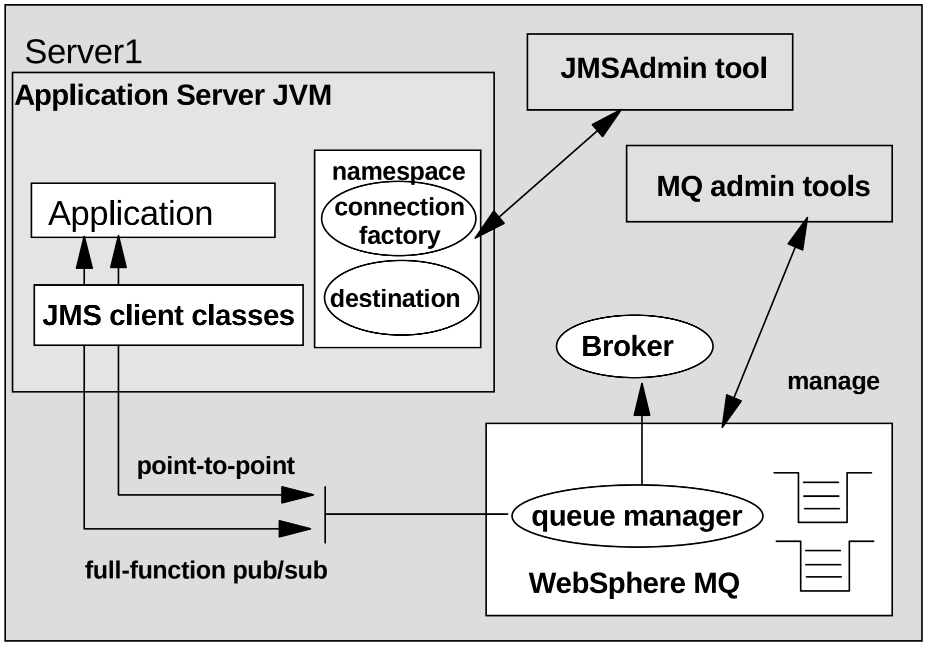


Figure 7-9 Full WebSphere MQ JMS provider components

The key points of the WebSphere MQ JMS provider configuration are:

- The broker is not hosted by the application server or the separate JMS Server process. Instead it is a separate process configured into the WebSphere MQ queue manager.
- The JMS Server is not used or required. The management of the queue manager, channels etc. is performed using the WebSphere MQ native tools. The WebSphere administration tools do not support configuration or management of these resources.

- The WebSphere administration tools are used to configure and manage the JMS administered objects (JMS connection factory and JMS destination) in the local namespace of the application server(s) that access WebSphere MQ.
- Either client (local or remote) or bind (local only) WebSphere MQ access mode can be used for JMS client access.
- The full set of WebSphere MQ functionality is available for use, including:
  - Store and forward.
  - Queue clustering.
  - Support for different brokers.

Author Comment: If we don't already have it somewhere, consider a table that compares the functions of each type of provider.

7.4.3 Administration

The WebSphere MQ JMS provider is partially integrated into WebSphere System Management. WebSphere administration tools can be used to both configure and manage WebSphere MQ JMS administered objects.

Creation and management of queue managers, channels and queues must be performed using WebSphere MQ native tools.

7.4.4 Security considerations

The WebSphere MQ JMS provider does not provide functionality to link to the WebSphere security infrastructure.

However, if both the integral JMS provider and WebSphere MQ JMS providers are installed on the same machine, WebSphere MQ can make use of WebSphere security, for authentication and authorization, if the following criteria are met:

1. The JMS Server of the integral JMS provider is configured and running.
2. The queue manager channel used for WebSphere MQ communications has been configured with the integral JMS provider's security exit (amqwascx).
3. WebSphere global security has been enabled.

Note: The JMS Server provides the security port listener used by the WebSphere MQ channel security exit to link to the WebSphere security infrastructure.

Author Comment: Need a reference here to the WAS 5 security redbook. Should probably have more info on security here since this is more of a component security issue than an application security issue.

7.4.5 Interoperability

Interoperates with other non-WebSphere related MQ installations, allowing WebSphere solutions to be integrated into heterogeneous WebSphere MQ environments.

7.5 Generic JMS provider

The generic JMS provider is recommended in the following cases:

- A non-WebSphere MQ messaging system already exists in the environment, and into which the WebSphere installation is required to integrate directly.
- Where a non-WebSphere MQ JMS provider supports functionality that is not available using WebSphere MQ, and which would be useful for the user's messaging environment.
- Connecting clients in one WebSphere 5.0 to JMS Servers in another WebSphere 5.0 cell, in order to enable inter-cell messaging.

7.5.1 Key features

Generic JMS providers have the following key features:

- Not limited to WebSphere MQ messaging systems. This may allow integration into non-WebSphere MQ based environments.
- Provides local JNDI aliases for the real JMS connection factory and JMS destination objects configured into the external namespace, hiding the use of an external provider and its resources from the application.

7.5.2 Limitations

Generic JMS providers suffer from the following limitations when used with WebSphere:

- Resources cannot be managed using WebSphere administration tools.
- Not tightly integrated into the WebSphere administration interface or service APIs.

- Connection classes may not support two-phase commit (XA) transactions linked to the WebSphere transaction service.

Author Comment: 2PC for messaging must be addressed in detail somewhere

- May require applications to use non-standard protocols and classes to connect to a remote JNDI namespace when accessing the JMS administered objects. The provider may not provide JMS tools that can register its objects in the WebSphere namespace.

7.5.3 Runtime components

The key points of the generic JMS provider configuration are:

1. Applications reference local JNDI entries set up as aliases to the remote JMS destination and JMS connection factory. It is the responsibility of the WebSphere runtime to resolve the accesses to the actual remote JNDI entries.

   The advantage of this approach is that it provides another level of indirection that protects the application from knowledge of the messaging topology.
2. The WebSphere runtime accesses the remote JNDI namespace using the provider-specific initial context factory class and provider URL.
3. The JMS administered objects in the external JNDI namespace are registered using tool(s) provided by the generic JMS provider.
4. The actual messaging objects (queues and topics) are created and managed using the provider's native tools.

See Figure 7-4 on page 161 for an overview of the components involved in WebSphere support for generic JMS providers.

Note: In order for an application to perform a lookup using the WebSphere local JNDI aliases, the generic provider's initial context factory class and JMS client classes must be in the application's classpath or the classpath of the application server in which the application is hosted.

7.5.4 Administration

The WebSphere administration tools only provide support for storing the information necessary to connect to the provider's JNDI context root:

- Initial context factory classname

- Provider URL

All configuration and management of the provider’s JMS administered objects (JMS connection factory and JMS destination) and messaging resources (queues and topics) must be performed using the provider’s own tools.

7.5.5 Security considerations

Generic JMS providers will not link to the WebSphere security infrastructure. This means that any authentication and authorization of JMS clients will have to be provided by the JMS provider itself, separate to WebSphere.

7.5.6 Interoperability

The interoperability of third party (generic) JMS providers with other providers (WebSphere MQ and others) is dependent upon the underlying message system, protocol and architecture used by the provider.

If the WebSphere MQ messaging protocol and message format are not used, then the provider will not interoperate directly with WebSphere MQ.

7.6 WebSphere clusters and MQ clusters

This section provides an overview regarding the use of WebSphere horizontal server clusters with WebSphere MQ server clustering. It describes a scenario that shows how the message listener service can be configured to take advantage of WebSphere MQ server clustering and provides some information about how to resolve potential runtime failures in the clustering scenario.

Note: WebSphere MQ server clustering is only available when the full WebSphere MQ product installed as the JMS provider.

For a WebSphere application server configured to use the message listener service, each JMS listener is used to retrieve messages from destinations defined to the server. In Figure 7-10, the listener configurations are the same for each WebSphere application server. Each application server host contains a WebSphere application server and an WebSphere MQ server. If a host is only used to distribute messages, it only contains an WebSphere MQ server. There can be many servers defined in the configuration, although for simplicity the information in this topic is based on a scenario containing only three servers.

Jeff, check this: I redrew this to add clusters & to show the application on both servers to illustrate it is the same application.

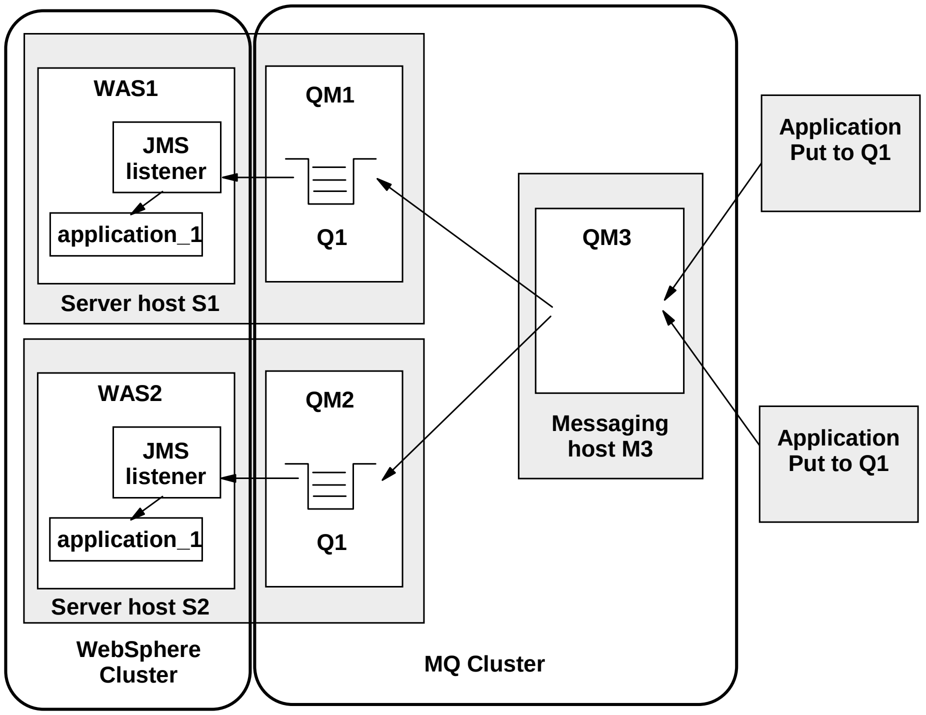


Figure 7-10 WebSphere MQ clustering and WebSphere horizontal cluster

Components

Figure 7-10 shows two WebSphere hosts, with a horizontal cluster, and a messaging host used to distribute messages for WebSphere MQ server clustering.

The scenario comprises the following three hosts:

1. Server host S1 contains the following servers:
   a. WebSphere MQ server

      The server is defined to have a queue manager, QM1, and a local queue, Q1. The queue manager belongs to a cluster. The queue is populated by the WebSphere MQ server located on host M3. Applications can add messages directly to the queue, Q1, but would not be subjected to the control of the WebSphere MQ cluster.

   b. WebSphere

      This contains a member of the horizontal server cluster, WAS1, which is configured with a JMS listener. The listener is configured to retrieve messages from JMS destination Q1.

2. Server host S2 contains the following servers:
   a. WebSphere MQ server

      The server is defined to have a queue manager, QM2, and a local queue, Q1. The queue manager belongs to the same cluster as QM1 on host S1. As with QM1, the queue is populated by the WebSphere MQ server located on host M3. Applications can add messages directly to the queue, Q1, but would not be subjected to the control of the WebSphere MQ cluster.

   b. WebSphere

      This contains a member of the horizontal server cluster, WAS2, which is configured with a JMS listener. The listener is configured to retrieve messages from JMS destination Q1.

3. Messaging host M3 contains the following servers:
   a. WebSphere MQ server

      The server is defined to have a queue manager, QM3, which also belongs to the same cluster as QM1 and QM2. Applications add messages to the queue manager and queue Q1. The cluster to which this queue manager belongs causes messages to be distributed to all other queue managers in the cluster which have queue Q1 defined.

Note:

1. Queue Q1 is not defined as a local queue on this host. If the queue was defined locally, then messages would remain on the server for local processing; messages would not be distributed by the queue manager cluster control to the other queue managers in the cluster that do have the queue defined
2. This host does not have a WebSphere application server defined. All message retrieval processing is performed by the other two application servers on hosts S1 and S2.

Recovery scenarios

There are several failure scenarios that could occur with this clustering configuration. For example:

- WAS server failures

  In this scenario the failure of any single WebSphere application server results in the messages for the specified destination remaining on the queue, until the server is restarted.

- WebSphere MQ queue manager failures

  There are two different failures to consider:

  a. Failure of a queue manager on the same host as a WebSphere application server (for example, failure of QM2 on host S2). In this case messages are delivered to the other available application servers, until the WebSphere MQ server is back online, when messages are processed as expected.
  b. Failure of the messaging host M3 and its queue manager, QM3. In this case, the result of an outage is more significant because no messages are delivered to the other queue managers in the cluster. In a fully-deployed and scaled production system, host M3 would not be designed to be a single point of failure, and additional messaging servers would be added to the MQ cluster configuration.

7.7 JMS scenarios

WebSphere support for different JMS providers provides a number of choices with which to implement an asynchronous messaging system. Table 7-5 provides a summary of the JMS functions supported by each of the scenarios. The decision on which scenario is best suited to a particular JMS messaging solution should be made by matching the scenario that best provides the required functions.

Table 7-5 JMS scenarios vs. messaging functions

Function	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Point-to-point	Y	Y	Y	Y	Y	Y	Y	Y
Persistent publish/subscribe	Y[2]	Y[2]	Y[2]	Y[2]	Y[2]	Y	Y	Y
Durable publish/subscribe	Y[2]	Y[2]	Y[2]	Y[2]	Y[2]	Y	Y	Y
Queue clustering	N	N	N	N	N	N	Y	Y
Store and forward	N	N	N	N	N	Y[1]	Y[1]	Y[1]
Shared topic space	N	N	N	N	N	N	Y	Y
JMS Server failover	N	N	N	N	N	N	Y	Y
High availability (HA) JMS Server	N	N	N	Y	N	N	Y	Y

Scenarios:
1. Integral JMS provider - single base server
2. Integral JMS provider - multiple base servers
3. Integral JMS provider - network deployment within a cell
4. Integral JMS provider - network deployment with high availability
5. Integral JMS provider - network deployment between cells
6. WebSphere MQ JMS provider - no clustered queues
7. WebSphere MQ JMS provider - clustered queues
8. WebSphere MQ JMS provider - network deployment between cells

Notes:

[1]Only if a local queue manager is present.
[2]Only if full function (queued) publish/subscribe is used.

- Persistent publish/subscribe

  All messages are delivered to a subscriber in order. No messages are lost, even in high volume situations.

- Durable publish/subscribe

  A subscriber can specify a client ID in their JMS connection and then create a durable subscriber. This causes the JMS Server to store persistently all messages matching the subscription on the server. IF the client disconnects and then reconnects with the same client ID, then it will start receiving messages from where it left off.

- Shared topic space

A publisher can publish a message on a topic to a JMS Server. Subscribers can connect to another JMS Server, subscribe to that topic and receive the message.

- Queue clustering

  Allows the sending of messages to a single logical queue to be load balanced across a number of “clustered” queues.

- Store and forward

  Logging of messages to a database so that they can be forwarded to a remote JMS Server at a later time.

7.7.1 Integral JMS - single base server

The key features of the configuration shown in Figure 7-11 are:

- All clients access the one JMS Server, hosted inside the application server JVM.
- No JMS Server failover or high availability.
- No shared topic space or store and forward.
- No queue clustering.

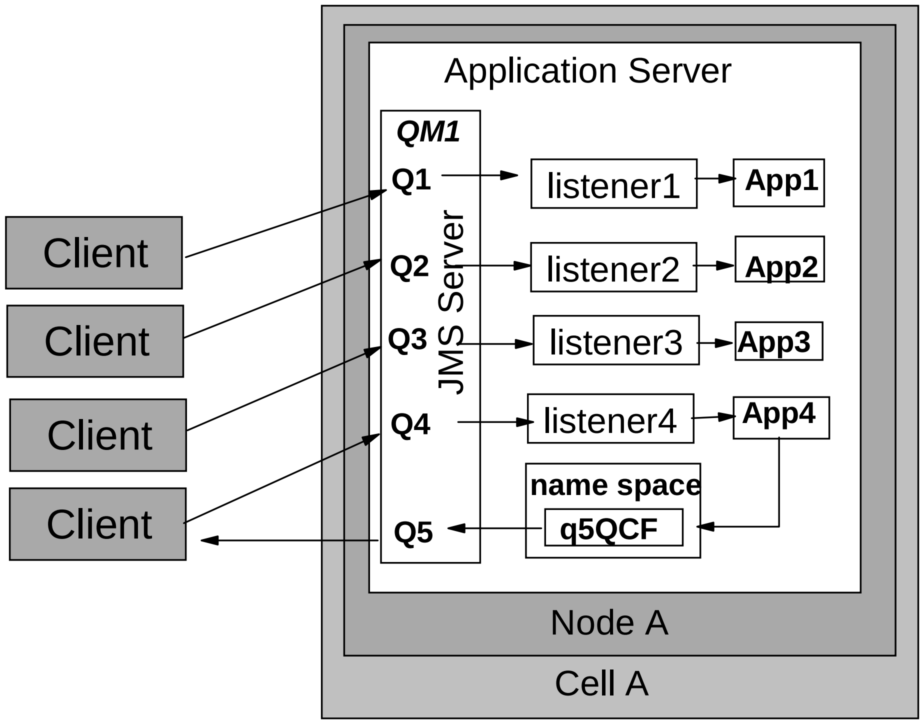


Figure 7-11 Integral JMS - single base server

7.7.2 Integral JMS - multiple base servers

The key features of the configuration shown in Figure 7-12 are:

- Applications in each application server can communicate with the JMS server embedded in the local (same) application server.
- No JMS Server failover or high availability.
- No shared topic space or store and forward.
- No queue clustering.
- In order for an application in one application server to interact with the JMS Server on the other application server, a *Generic JMS connection factory* must be configured to point at the remote JMS server.

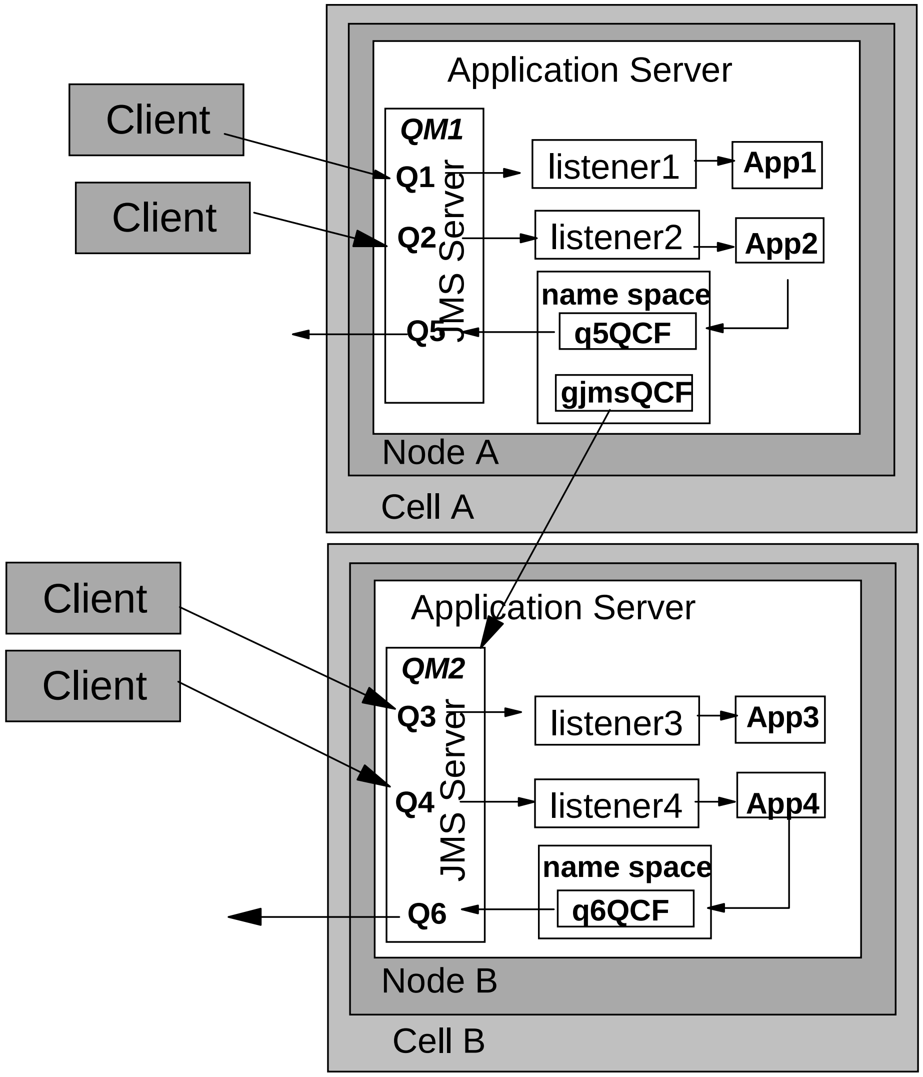


Figure 7-12 Integral JMS - multiple base servers

7.7.3 Integral JMS - network deployment within a cell

The key features of the configuration shown in Figure 7-13 are:

- Applications on any application server in the cell can connect to any JMS Server in the cell, using the JMS administered objects registered in the cell's federated namespace.

- The JMS Server runs as a separate JVM, enabling it to be started or stopped independent of any application server. A node could be configured to not run a JMS Server.
- No shared topic space or store and forward.
- No queue clustering.
- No JMS Server failover or high availability.
- No queue clustering.

Jeff: do we really need the node agent box? I know it helps illustrate that this is an administrative cell but does it have anything to do with the way messaging works?

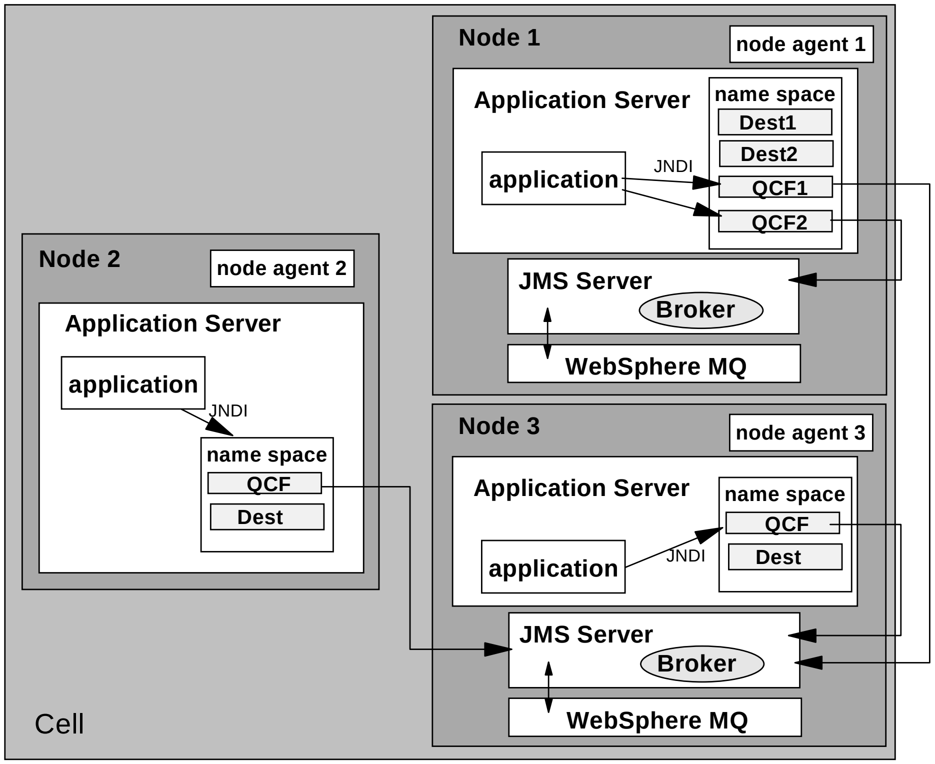


Figure 7-13 Integral JMS - network deployment within a cell

7.7.4 Integral JMS - network deployment within high availability

This is a variation of the configuration outlined in Section 7.7.3, “Integral JMS - network deployment within a cell” on page 195. Its key features are:

- One of the nodes is dedicated to running the JMS Server, i.e.. it is not used to run application server processes.
- All other nodes do not run a local JMS Server.
- The applications hosted by the application servers on all other nodes of the cell use the single dedicated JMS Server.
- The machine used for the dedicated JMS is a high availability configuration.

Jeff .. check figure:

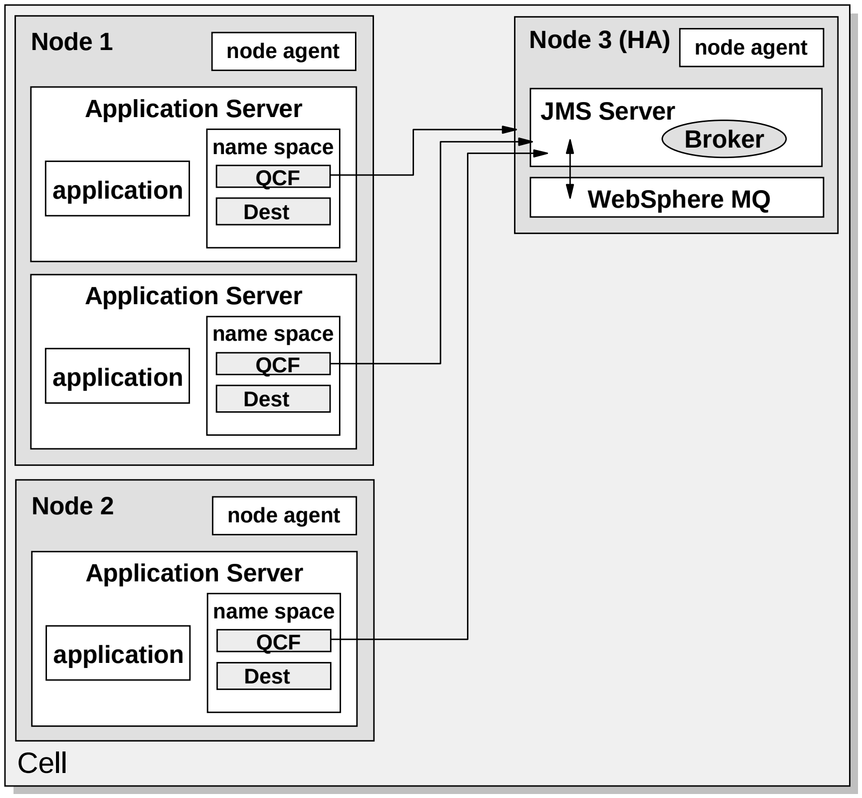


Figure 7-14 Integral JMS - network deployment with high availability

7.7.5 Integral JMS - network deployment between cells

The key features of the configuration shown in Figure 7-15 are:

- Applications in each cell can communicate with JMS servers in the same cell by using the WebSphere JMS provider resources in the cell namespace (via JNDI).
- In order for an application in one cell to interact with a JMS Server in another cell, a *generic JMS connection factory* must be configured in its cell to point at the remote JMS server.
- No shared topic space or store and forward.
- No queue clustering.
- No JMS Server failover or high availability.

Jeff .. check figure:

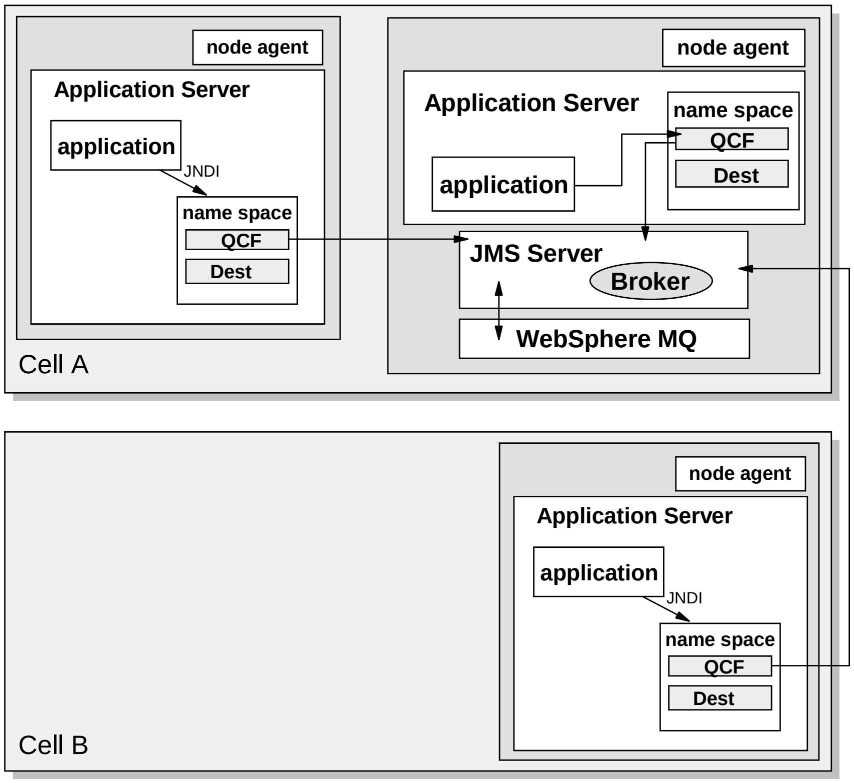


Figure 7-15 Integral JMS - network deployment between cells

7.7.6 WebSphere MQ JMS - no clustered queues

The key features of the configuration shown in Figure 7-16 are:

- If an application has a local queue manager, then store and forward is possible. WebSphere MQ communications using client mode (non-bind) cannot perform store and forward.
- Shared topic space.
- Each queue is independent and distinct.

Jeff .. check figure:

Author Comment: Need figure like pp13 “Full MQ - no clustered queues” of JMS Scenarios.pdf. However need to redo to reflect current names for Base and ND configuration appservers.

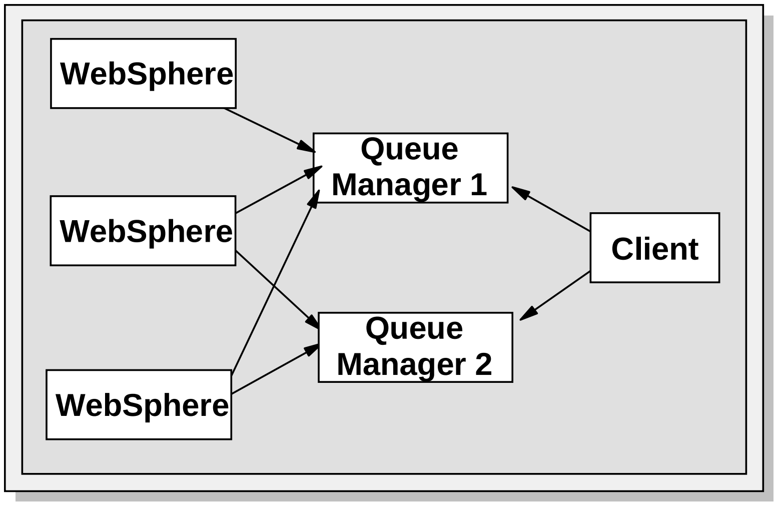


Figure 7-16   Full WebSphere MQ - no clustered queues

7.7.7 WebSphere MQ JMS - clustered queues

The key features of the configuration shown in Figure 7-17 are:

- Clustering is performed at the client end. One or more queue managers are repositories keeping track of the cluster status.

- A client asks a repository for all queue managers hosting a particular clustered Queue, and then the client load balances PUT requests between them.
- GET requests are not clustered. A client attaches to a queue on a queue manager and gets messages from it.

Author Comment: Need figure from pp14 “Full MQ - no clustered queues” of JMS Scenarios.pdf

-

Jeff .. check figure:

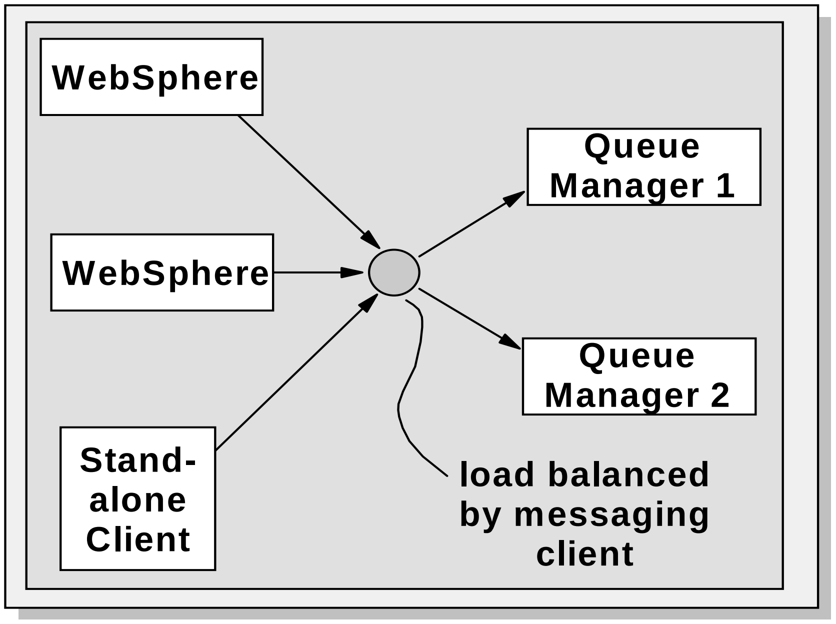


Figure 7-17 Full WebSphere MQ - clustered queues

7.7.8 WebSphere MQ JMS - network deployment between cells

The key features of the configuration shown in Figure 7-18 are:

- Shared topic space is possible between cells.
- Store and forward is possible between cells if the node sending the message has a local WebSphere MQ queue manager.

- A client can use queue clustering to spread messages over two cells. This enables one cell to backup the operation of the other
- .

Jeff .. check figure:

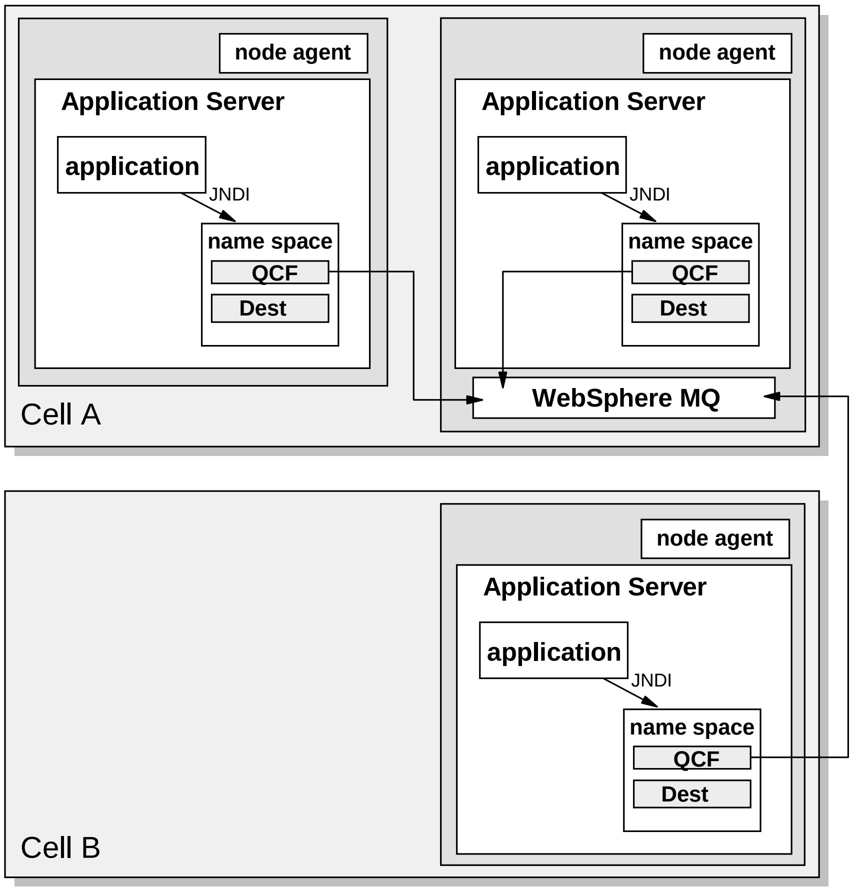


Figure 7-18 Full WebSphere MQ - network deployment between cells

7.8 Monitoring and controlling??

Author Comment: We might need an “operations” section here that tells how to display and monitor the MQ resources if it is not covered somewhere else.

7.9 Where to find more information

Author Comment: Need to credit Billy Newport, Rochester MN for the JMS Scenarios pdf, course authors?

Part 2

Installing WebSphere

linux and iSeries will be done in other projects. Need to either point to them or get the install info and put in chapters in this section.

8

Installation approach

This chapter provides an explaination of the procedures for installing, configuring and verifying IBM WebSphere Application Server Network Deployment. Since WebSphere 5.0 can be installed in a number of different configurations, this chapter provides a summary of the decsions that must be made when planning an installation to match a particular topology or architecture.

Author Comment: Should this capter use the scenarios defined in the Toplogies chapter for consitency?

This chapter is organized into the following sections:

- Planning your installation
- Perform Pre-installation tasks
- Perform WebSphere installation
- Perform Post installation tasks
- Uninstalling WebSphere
- Exxample Scenarios

8.1 Planning your installation

Since WebSphere 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 pla your installation.

Prior to performaing an installation, you should consider each of the following options, as each effects 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?
- 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?

8.1.1 Migration of an 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, or 4.0.x installation to 5.0 is provided **[WHERE]**.

This chapter concentrates on the decisions and tasks associated with a “clean”

Author Comment: Are we going to detail migration or reference a redbook here.

8.1.2 Use GUI or Silent installation

Once you have planned for the installation, you are ready to choose an installation method. Plan whether to use the installation wizard GUI, or to install silently, without a GUI as the installer program reads a file of installation option responses that you specify.

The default installation method is to click “Install the product”, you will need to modify the on the LaunchPad. This launches the installation wizard GUI. Silent installation reads all choices from the options response file, which you must prepare before installation. Silent installation is particularly useful if you install the product often.

If you choose silent installation you will need to modify the silent response file that WebSPhere will use during the installation. An example of a response file is below.

```
# Response file for WAS 5.0 Install
#
# Please read the comment lines to understand how to use this file and its
various options.
#
# Fill out or change various values carefully to avoid the installation failure
that
# an improper entry might cause.
#
# ************************************************************
# **                                                        **
# ** IMPORTANT: ENCLOSE ALL VALUES IN DOUBLE QUOTES ( "" ). **
# **                                                        **
# ************************************************************
#
# ************************************************************
# Beginning of user changes, which is the portion of the response file that you
must change.
# ************************************************************
#
# ************************************************************
# Specify whether to silently install with the -silent line.
# To install with the GUI, comment out the line with an # in column 1, or
delete the line.
# ************************************************************

-silent

# ************************************************************
# WAS Install Location
# You must change the install location for UNIX platforms. For example: for
AIX, change
```

```
# -P wasBean.installLocation="C:\Program Files\WebSphere\AppServer"
#  to
# -P wasBean.installLocation="/usr/WebSphere/AppServer"
# ************************************************************

-P wasBean.installLocation="C:\Program Files\WebSphere\AppServer"

# ************************************************************
# IHS install Location
# ************************************************************

-P ihsFeatureBean.installLocation="C:\Program Files\IBMHTTPServer"

# Set each of the following values to true or false,
# to control whether each feature is installed (true) or not installed (false)
#
# ************************************************************
# Install Server
# ************************************************************

-P serverBean.active="true"

####
#
# Begin Features for Admin
#
####

# ************************************************************
# Install Admin
# ************************************************************

-P adminBean.active="true"

# The next 2 features are part of Admin. In order that any of these
# features be installed, the property to install Admin as above must be
# set to true.
#
# ************************************************************
# Install Admin Scripting
# ************************************************************

-P adminScriptingFeatureBean.active="true"

# ************************************************************
# Install Admin Console
# ************************************************************

-P adminConsoleFeatureBean.active="true"
```

```
#
# **************************************************************
# End Features for Admin
# **************************************************************
#
#
#
# **************************************************************
# Begin Features for Application And Assembly Tools
# **************************************************************
#
# **************************************************************
# Install Application And Assembly Tools
# **************************************************************

-P applicationAndAssemblyToolsBean.active="true"

# The next 3 features are part of Application And Assembly Tools.
# To install any of these features, the property to install
# Application And Assembly Tools as above must be set to true.
#
# **************************************************************
# Install Application And Assembly Tool
# **************************************************************

-P applicationAssemblyToolBean.active="true"

# **************************************************************
# Install deploy Tool
# **************************************************************

-P deployToolBean.active="true"

# **************************************************************
# Install Ant Utility
# **************************************************************

-P antUtilityBean.active="true"

# **************************************************************
# End Features for Application And Assembly Tools
# **************************************************************
#
# **************************************************************
# Begin Features for Embedded Messaging
# **************************************************************
#
# **************************************************************
# Install Embedded Messaging
# ************************************************************
# To install either of the following sub-features,
# set the mqSeriesBean.active feature to true.

-P mqSeriesBean.active="true"

# The next two features are for Embedded Messaging Server and Client.
# ************************************************************
# Install Embedded Messaging Server.
# ************************************************************

-P mqSeriesServerBean.active="true"

# ************************************************************
# Install the Embedded Messaging Client
# ************************************************************

-P mqSeriesClientBean.active="true"

# ************************************************************
# Install IHS WebServer 1.3.24
# ************************************************************

-P ihsFeatureBean.active="true"

# ************************************************************
# Install Plugin Files
# ************************************************************

-P pluginBean.active="true"

# ************************************************************
# Install IHS plugin
# ************************************************************

-P ihsPluginBean.active="true"

# ************************************************************
# Install Apache Plugin
# ************************************************************

-P apachePluginBean.active="false"

# ************************************************************
# Install IIS Plugin
# ************************************************************

-P iisPluginBean.active="false"
```

```
# ************************************************************
# Install IPlanet Plugin
# ************************************************************

-P iplanet60PluginBean.active="false"

# ************************************************************
# Install Domino Plugin
# ************************************************************

-P dominoPluginBean.active="false"

# ************************************************************
# Install Samples
# ************************************************************

-P samplesBean.active="true"

# ************************************************************
# Begin Features for Performance and Analysis Tools
# ************************************************************
#
# ************************************************************
# Install Performance And Analysis Tools
# ************************************************************

-P performanceAndAnalysisToolsBean.active="true"

# ************************************************************
# The next 3 features are part of Performance And Analysis Tools. To install
any of these
# features, the value in the line above must be true.
# ************************************************************

# ************************************************************
# Install Tivoli Performance Viewer
# ************************************************************

-P tivoliPerfBean.active="true"

# ************************************************************
# Install Dynamic Cache Monitor
# ************************************************************

-P DCMBean.active="true"

# ************************************************************
# Installs Performance Servlet
# ************************************************************

-P performanceServletBean.active="true"

#
# ************************************************************
# End Features for Performance and Analysis Tools
# ************************************************************
#
# ************************************************************
# Install Javadocs
# ************************************************************

-P javadocBean.active="true"

# ************************************************************
# Replace "DefaultNode" with the name you intend for your default node.
# You must also replace "127.0.0.1" with the resolveable hostname
# or IP address for your machine.
# ************************************************************

-W nodeNameBean.nodeName="DefaultNode"
-W nodeNameBean.hostName="127.0.0.1"


# ************************************************************
# **                                                        **
# **                 Installing Services                    **
# **                                                        **
# ************************************************************
# The following settings install Services for IHS and WAS
# on Windows NT(TM) or Windows 2000(TM).
# Ignore these settings (or comment them out) for other operating systems.
# ************************************************************

-W serviceSettingsWizardBean.active="true"

# ************************************************************
# Install the IHS service
# ************************************************************

-W serviceSettingsWizardBean.ihsChoice="true"

# ************************************************************
# Install the WAS service
# ************************************************************

-W serviceSettingsWizardBean.wasChoice="true"
```

```
# ************************************************************
# The User Name and Password are required to install services.
# The user name must be "Administrator" or belong to the
# Administrator group and have the advanced user rights:
# "Act as part of the operating system" and "Log on a service".
#  * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
# Replace YOUR_USER_NAME with your username.
# ************************************************************

-W serviceSettingsWizardBean.userName="YOUR_USER_NAME"

# ************************************************************
# Replace YOUR_PASSWORD with your valid password.
# ************************************************************

-W serviceSettingsWizardBean.password="YOUR_PASSWORD"

# ************************************************************
# **               End Installing Services                  **
# ************************************************************
#
# ************************************************************
# Set any or all of the following to false if the launcher icon is not to be
installed.
# These settings affect a product component only when it is
# selected for installation.
# ************************************************************

-P StartServerIconBean.active="true"
-P StopServerIconBean.active="true"
-P AdminConsoleIconBean.active="true"
-P AssemblyToolIconBean.active="true"
-P SamplesGalleryIconBean.active="true"
-P TivoliPerfIconBean.active="true"
-P infoCenterIconBean.active="true"
-P firstStepsIconBean.active="true"

# ************************************************************
# Change the path to the prerequisite checker configuration file only
# if you are using a file other than the one provided in the installation
package.
#
# Notes:
# 1. You can use a relative or absolute path.
# 2. Make sure the DTD file, prereqChecker.dtd, is in the same path.
#
# ************************************************************

-W osLevelCheckActionBean.configFilePath="waspc/prereqChecker.xml"
```

```
# ************************************************************
# Specify a fully qualified path for the following values,
# including the config file name for plug-ins.
# To install a plug-in, you must specify this path. Otherwise,
# the installer fails to install plug-ins properly.
# Enclose the path value in double quotes.
#  * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
# IHS Config File Location
# ************************************************************

-W defaultIHSConfigFileLocationBean.value="C:\Program
Files\IBMHTTPServer\conf\httpd.conf"

# ************************************************************
# Apache Config File Location
# ************************************************************

-W defaultApacheConfigFileLocationBean.value=

# ************************************************************
# IPlanet Config File Location
# ************************************************************

-W defaultIPlanetConfigFileLocationBean.value=

# ************************************************************
#                  Domino Plugin Config
#  * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
# Domino notes.jar File Location
# ************************************************************

-W dominoPanelBean.notesJarFile=

# ************************************************************
# Domino names.nsf File Location
# ************************************************************

-W dominoPanelBean.namesFile=

# ************************************************************
#               End of Domino Plugin Config
# ************************************************************
#
# ************************************************************
# Disable product registration notification for silent install
# ************************************************************

-W launchPRTBean.active="false"
```

```
# **************************************************************
# Install Default App
# **************************************************************

-W installSampleAppSequenceBean.active="true"

# **************************************************************
# Install First Steps
# **************************************************************

-W firstStepsSequenceBean.active="true"

# **************************************************************
# Install IVT
# **************************************************************

-W installIVTAppSequenceBean.active="true"


# **************************************************************
# **                                                          **
# **               Silent migration                           **
# **                                                          **
#  * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
# **  Uncomment and modify all properties in this section,  **
# **        for silent migration to work properly.          **
# **************************************************************
# To direct the installer to operate on a previous version,
# tell it that one is present by uncommenting the next line.
# **************************************************************

# -W previousVersionDetectedBean.previousVersionDetected="true"

# **************************************************************
# Direct the installer to operate on a specific previous version
# by uncommenting the next line and entering one of these values:
#  * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
#   Value           Edition
#   *****           *******
#   AE              WAS Advanced Edition (V3.x, V4.0.x)
#   advanced        AE
#   AEs             WAS Advanced Single Server Edition (V4.0.x)
#   standard        WAS Standard Edition (V3.x)
# **************************************************************

# -W previousVersionPanelBean.selectedVersionEdition="AEs"

# **************************************************************
# The location where the previous version is installed
# ************************************************************

# -W
previousVersionPanelBean.selectedVersionInstallLocation="/opt/WebSphere/AppServ
er"

# ************************************************************
# The path to the configuration file for the previous version.
# Configuration filenames are:
#  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
#   Value          previousVersionPanelBean.selectedVersionEdition
#   *****          **********************************************
#   admin.config   AE
#   admin.config   advanced
#   server-cfg     AEs
#   server-cfg     standard
# ************************************************************

# -W
previousVersionPanelBean.selectedVersionConfigFile="/opt/WebSphere/AppServer/co
nfig/server-cfg.xml"

# ************************************************************
# The version number of the previous version: 4.0 4.0.1 3.5 etc...
# ************************************************************

# -W previousVersionPanelBean.previousVersionSelected="4.0"

# ************************************************************
# Indicator for migrating the previous version
# ************************************************************

# -W previousVersionPanelBean.migrationSelected="true"

# ************************************************************
# Directory for migration tools to backup information about previous version
# ************************************************************

# -W migrationInformationPanelBean.migrationBackupDir="/tmp/migrationbackup"

# ************************************************************
# Directory for storing migration logs
# ************************************************************

# -W migrationInformationPanelBean.migrationLogfileDir="/tmp/migrationlogs"
```

Once the file has been configured, you can use the following command to execute the silent installation:

```
Install.bat -options myoptionsfile
```

This will take in your response file as a parameter and begin the silent installation.

Reasons to consider using a GUI installation might be:

- First-time or inexperienced users
- One-time installations
- Determinisaton of the settings required to create the script file that will be used in later silent installations
- Testing installations involving different combinations of software component.

The choice of using a silent installation should be based on the following scenarios:

- Installation of identical configurations of WebSphere Application Server on mulitple machines
- Back up of installation settings for later use
- scripted or unattended installations
- Duplication of testing or development environments.

8.1.3 Use Typical or Custom installation

8.1.4 Use existing or create new administrative domain

8.1.5 Use Embedded HTTP Transport or stand-alone Web server

8.1.6 User remote web Server

8.1.7 Customize prerequisite checking

8.1.8 Run on non-standard ports?

8.1.9 Run with multiple NICs?

8.1.10 Run as non-root user

You can run the application server, base and net work deployment as a non root user. However,Root Authority is required to install embedded messaging. Set the following permissions before installing WebSphere Application Server:

1. Create the groups mqm and mqbrkrs if no already created.
2. Add users to mqm and root to the group mqm.
3. Add user root to the group mqbrkrs.
4. Log out and log back in to set permissions.

8.2 Perform Pre-installation tasks

8.3 Perform WebSphere installation

8.4 Perform Post installation tasks

8.5 Uninstalling WebSphere

8.6 Example Scenarios

We provide specific installation examples for Windows 200, AIX, Solaris and Linux in the chapters that follow.

Note: Explain to readers that the DB is no longer required and is not considered a core component. Therefore we willl not discuss it in this book

Author Comment: Run thru whole book and make sure any platform-specific info is updated (directory structures in particular).

Author Comment: Here's some ideas from Jeff

1) Remove the installation and configuration of the databases - as discusssed earlier, the WebSphere runtime doesn't require a DB anymore, so is only required for:

- Session Persistence to a DB... can get around this using memory-to-memory session replication.
- Application database

It's no longer a core component, so doesn't *have* to be installed, configured or therefore described by us.

2) Add new installation scenarios covering the installation and configuration of the JMS Provider messaging component. As a minimum I'd describe the internal JMS provider, but I think since most people would move up to full WebSphere MQ, there would be value in describing the installation and configuration of WebSphere MQ *with* WAS 5.0.

3) In terms of post-installation configuration tasks in the "Installation Approach" chapter, some of the tasks/decisions may need to be amended or removed. For example, I haven't seen any information on how to run the different WAS5 processes as non-root??

Author Comment: Somewhere we need to talk about migrating from a base config to a ND config. I think this would be in one of the install chapters ??

9

Windows 2000 installation steps

This chapter provides detailed procedures for installing, configuring and verifying a WebSphere Application Server environment. If you have not done so, you should review the information in Chapter 8, “Installation approach” on page 205. The infocenter, which is provided online or with the installation, will provide simple instructions on how to install WebSphere Application Server using the launchPad.bat utility. This chapter aims to provide more detailed procedures and trobleshooting tips.

This chapter will discuss the following:

- Installation planning
- Installing IBM HTTP Server
- Installing IBM WebSphere Application Server
- Installing IBM WebSphere Application Server Network Deployment
- Verifying the installation
- Installing the WebSphere HTTP Plug-in
- Adding a new node to the cell
- Configuring SSL between the Web server and the IBM WebSphere Application Server
- Installing in silent mode
- Multiple WebSphere installations
- Troubleshooting install problems

9.1 Planning your installation

This section defines the hardware and software used within the Windows 2000 envioronment to test a number of different WebSphere 5.0 configuration scenarios.

9.1.1 Hardware and software prerequisites

Author Comment: Will delete this and defer to the product documentation

IBM WebSphere Application Server Network Deployment, as well as the base installation, have the following hardware and software requirements:

Hardware

- Intel Pentium processor at 500 MHz or faster(Windows NT, 2000)
- 512 MB RAM
- 2 GB Disk space available
- Network Connection (Ethernet or Token Ring)
- CD-ROM drive

Software

- Microsoft® Windows® NT 4.0 SP 6a or Windows 2000 SP2
- IBM DB2 UDB v7.2 Enterprise Edition with FixPak 6 or later, or Oracle 8.1.7, or Microsoft SQL Server 7.0 SP2 and 2000
- Microsoft Internet Explorer v5.5 SP1, or Netscape Communicator 4.7.3 or 4.7.6.
- IBM HTTP Server version 1.3.26, or iPlanet Web Server, Enterprise Edition version 6.0.1

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 2
- IBM WebSphere Application Server, Base
- IBM IBM WebSphere Application Server Network Deployment
- IBM HTTP Server 1.3.26
- 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
<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
  - 1.8 GHz Pentium 4 CPU
  - 1.5 GB RAM
  - 19 GB hard disk
  - 1 IBM 10/100 EtherJet PCI Ethernet adapter
- Server 2
  - IBM PC 300PL, Model 65653BU
  - 1.8 GHz Pentium 4 CPU
  - 1.5 GB 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 the Installation Approach.

- 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 C- 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 D- extension of the secure two-server configuration of Scenario E, with the SSL client authentication mode enabled.
- Scenario E: basic 2 server configuration with multiple application servers in a clustered enviornment using IBM WebSphere Application Server Network Deployment

As each scenario builds on the previous we will begin will scenatrio a and evolve the toplogy on our test machine until we get to scenario E.

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 2 - 128-bit encryption.

9.3 Install Web Server

This section provides detailed instructions for installing, configuring, and verifying IBM HTTP Server V1.3.26 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 [PLANNING] for details on those tasks to be performed for each example.

Preinstallation tasks

Prior to installing IBM HTTP Server V1.3.26, 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
```

Install IBM HTTP Server

To install the Web server V1.3.26, 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 \base directory on the CD. Double-click **LaunchPad.bat** 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. This is the same GUI you will use to install the application server. However, instead of installing the application server, you are going to do a custom install to install an external HTTP server. Select **Install the Product** on the launchpad to continue.
6. Select the language you will use for the installation, then select **OK**.
7. Select **Next** on the Welcome Screen and the Lincense Agreement Screen
8. Seclect **Custom** for the installation type the select **Next** to continue.
9. The dialog will display a list f features to install. Deselect all features except IBM HTTP Server. Select **Next** to conitnue.

Figure 9-1 Installing IHS

10. On the next dialog, select the directory structure that you would like to have IHS installed to. You will also have to choose a directory structure for WebSphere, even though you are not installing it at this time. This is required because the log files created by the installation are written to the WebSphere directory. Select **Next** to continue.
11. On the base install, you will be prompted to provide a userid and password for starting and stopping the HTTP Server. The userid value will default to the administrative UID you logged in at to begin this installation. You can also choose to run UIHS as a operating system service. Make your selections and select **Next** to continue.
12. The Summary dialog will diplasy that you have chosen to install IHS, select **Next** to begin coping the files.

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** ->**Accessories**-> **Command Prompt**.

Change directory to <http_server_install_path>.

2. Create the administration user by typing the following commands:

   ```
   D:\IBMHTTPServer> 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. You may have to go to the /conf directory and create the admin.passwd file (or use the -c option with the htpasswd command).

Note: Any user ID can be used as the IBM HTTP Server administration user ID.

3. 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.

Reboot the system.

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 9-2 have been added and are running.

Table 9-2 IBM HTTP Server Windows services

Service name	Status	Startup mode	Log on as...
IBM HTTP Administration	Started	Automatic	wasadmin
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 9-2 IHS Welcome Screen

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.

Save the changes.

Create 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?**
5. Click **OK** when the Information window appears with the message:

   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.

```
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`
5. The new certificate should be listed in the Personal Certificates pane
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>/`

Since the test certificate we are using is self-signed, the browser should display a window which identifies the use of certificates. Click **Next/Yes**.

If the IBM HTTP Server has been correctly configured for SSL, then the browser should direct you to the welcome page for IHS.

9.4 Install IBM 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.

9.5 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.

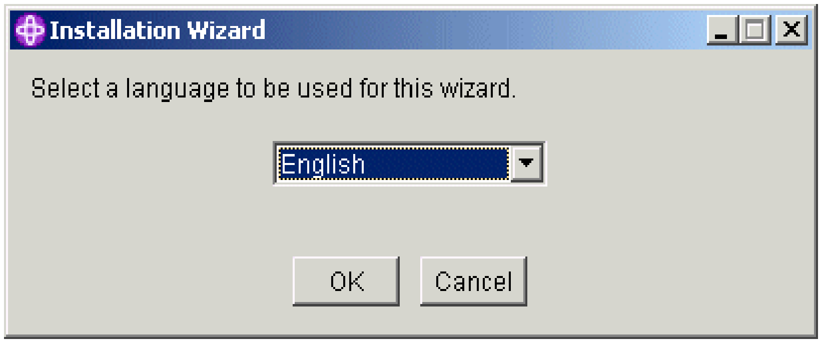
1. Issue the command:

```
D:\> net stop "IBM HTTP Server"
```

9.6 Install WebSphere, Base

To install IBM WebSphere Application Server, base using the GUI installer interface, complete the following steps on the WebSphere machine:

1. Log on as an administrator user in the local server domain (not part of the windows domain).
2. Insert the IBM WebSphere Application Server V5.0 CD.
3. Start the IBM WebSphere Application Server installation by clicking install.bat on the root of the CD.
4. In the Installer dialog, select a language to be used for the installation, select **OK**.

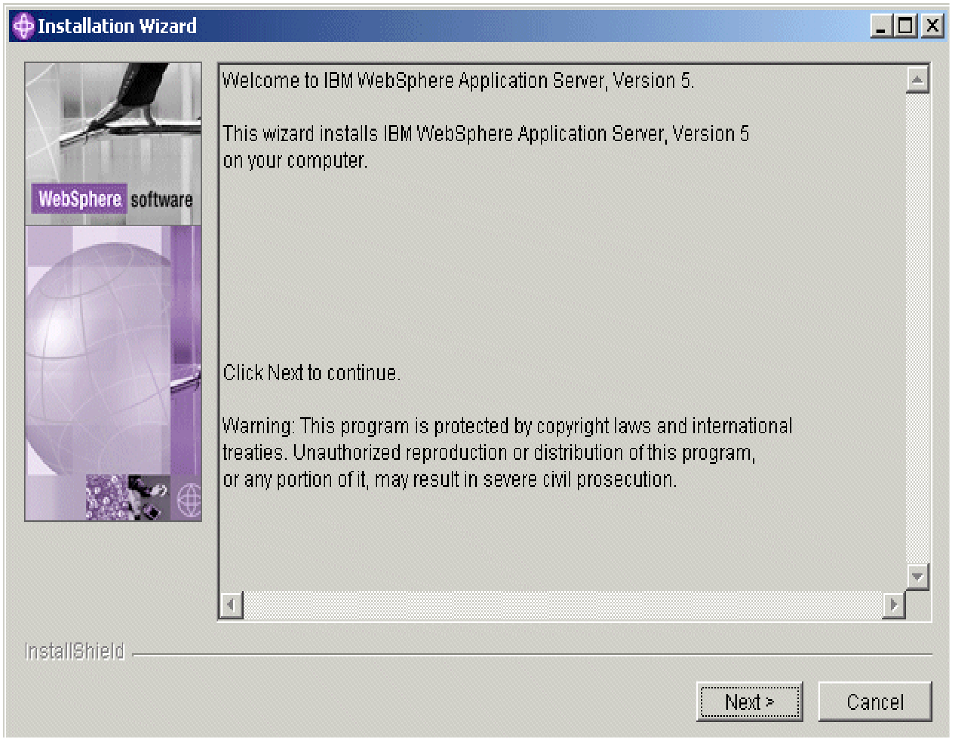

Installation Wizard

Select a language to be used for this wizard.

English

Figure 9-3 Choose language for installation

5. The welcome screen will appear, read the warnings and then click **Next** to continue..



Welcome to IBM WebSphere Application Server, Version 5.

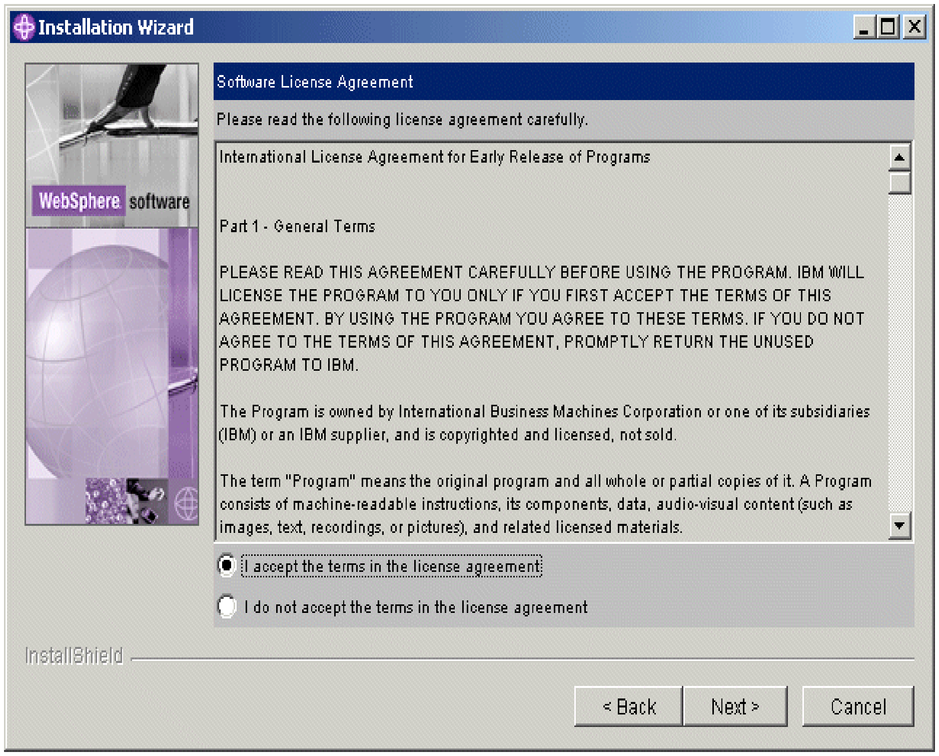
This wizard installs IBM WebSphere Application Server, Version 5 on your computer.

Click Next to continue.

Warning: This program is protected by copyright laws and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil prosecution.

Figure 9-4 Welcome screen

6. The software licence agreement dialog will appear. Read the licence agreement, and if you agree select the radio button next to “I agree with the terms of this licence agreement and select **Next** to continue.



Software License Agreement

Please read the following license agreement carefully.

International License Agreement for Early Release of Programs

Part 1 - General Terms

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE USING THE PROGRAM. IBM WILL LICENSE THE PROGRAM TO YOU ONLY IF YOU FIRST ACCEPT THE TERMS OF THIS AGREEMENT. BY USING THE PROGRAM YOU AGREE TO THESE TERMS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, PROMPTLY RETURN THE UNUSED PROGRAM TO IBM.

The Program is owned by International Business Machines Corporation or one of its subsidiaries (IBM) or an IBM supplier, and is copyrighted and licensed, not sold.

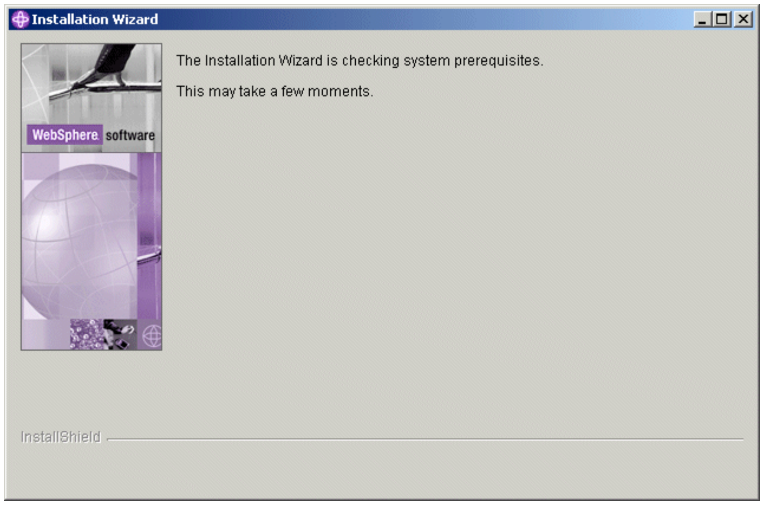
The term "Program" means the original program and all whole or partial copies of it. A Program consists of machine-readable instructions, its components, data, audio-visual content (such as images, text, recordings, or pictures), and related licensed materials.

[● I accept the terms in the license agreement]

○ I do not accept the terms in the license agreement

Figure 9-5 License Agreement Dialog

7. After accepting the licencse agreement, the installtion will check for prerequisites on the installation machine.

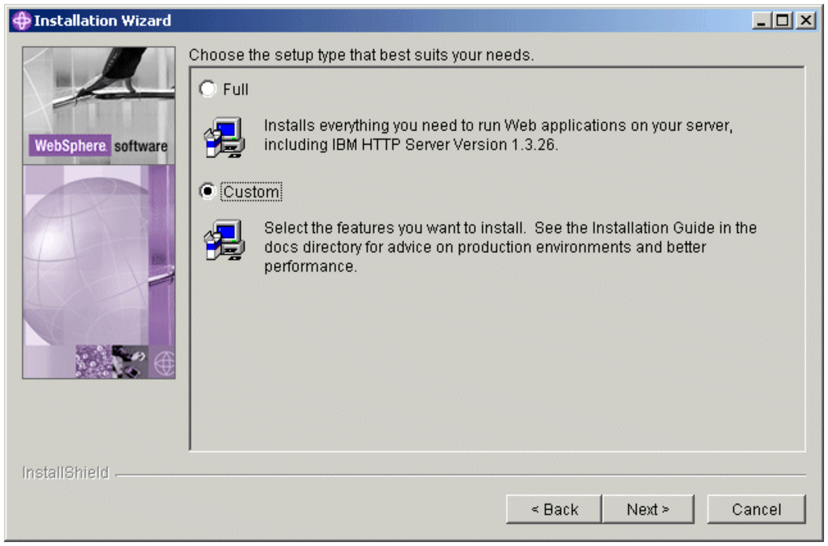


The Installation Wizard is checking system prerequisites.

This may take a few moments.

Figure 9-6 Checking for installtion prerequisites

8. Once it has confirmed that all prerequisites have been met, the dialog with display the Installation Options. WebSphere provides two installation options. Select the custom radio button and **`Next`** to continue.



Choose the setup type that best suits your needs.

Full

Installs everything you need to run Web applications on your server, including IBM HTTP Server Version 1.3.26.

[✓ Custom]

Select the features you want to install. See the Installation Guide in the docs directory for advice on production environments and better performance.

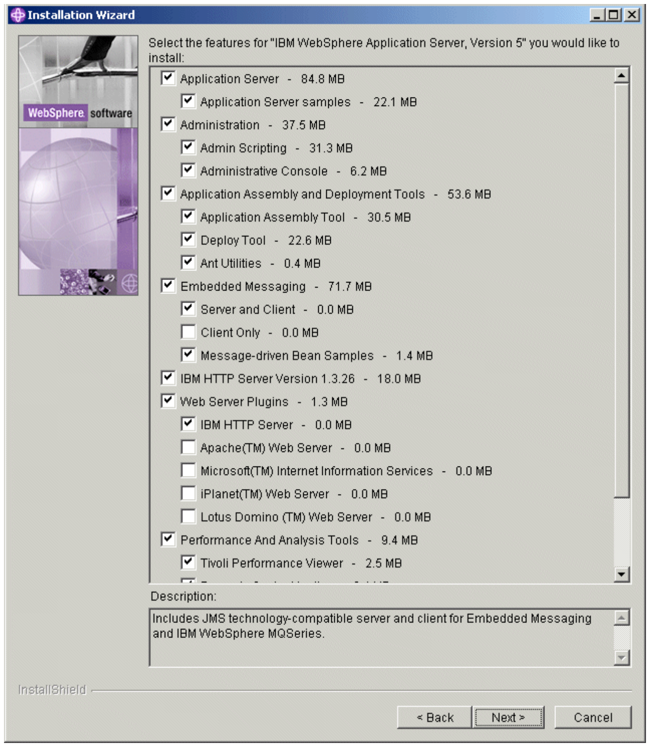
Figure 9-7

The typical installation automatically installs a WebSphere Application Server configuration consisting of the following components on a single server:

– WebSphere Application Server 5.0
– IBM Http Server 1.3.23
– WebSphere HTTP Plug-in for IBM HTTP Server
– IBM JDK ??
– Sample Applications

A custom installtion allows the user to choose among the following options

– Application Server and Samples
– Administration Utilities
– Administrative tooling
– Application Assembly and Deployment tooling
– Embedded Messaging
– Web server and plugins
– Performance and analysis tooling
– Java Documentation



Select the features for "IBM WebSphere Application Server, Version 5" you would like to install:

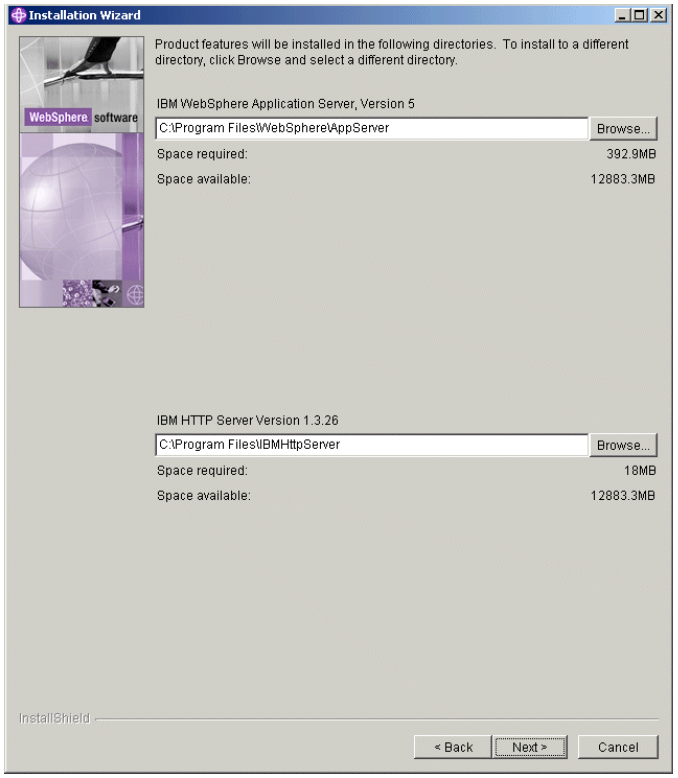
- [✓] Application Server - 84.8 MB
  - [✓] Application Server samples - 22.1 MB
- [✓] Administration - 37.5 MB
  - [✓] Admin Scripting - 31.3 MB
  - [✓] Administrative Console - 6.2 MB
- [✓] Application Assembly and Deployment Tools - 53.6 MB
  - [✓] Application Assembly Tool - 30.5 MB
  - [✓] Deploy Tool - 22.6 MB
  - [✓] Ant Utilities - 0.4 MB
- [✓] Embedded Messaging - 71.7 MB
  - [✓] Server and Client - 0.0 MB
  - [ ] Client Only - 0.0 MB
  - [✓] Message-driven Bean Samples - 1.4 MB
- [✓] IBM HTTP Server Version 1.3.26 - 18.0 MB
- [✓] Web Server Plugins - 1.3 MB
  - [✓] IBM HTTP Server - 0.0 MB
  - [ ] Apache(TM) Web Server - 0.0 MB
  - [ ] Microsoft(TM) Internet Information Services - 0.0 MB
  - [ ] iPlanet(TM) Web Server - 0.0 MB
  - [ ] Lotus Domino (TM) Web Server - 0.0 MB
- [✓] Performance And Analysis Tools - 9.4 MB
  - [✓] Tivoli Performance Viewer - 2.5 MB

Description:

Includes JMS technology-compatible server and client for Embedded Messaging and IBM WebSphere MQSeries.

Figure 9-8 Installation Options

9. The next screen will enable the user to select the installation directories for IBM WebSphere Application Server and IBM HTTP Server. Take out the reference to the Directory Program Files, and it is recommended that you do not install to the root directory (c:/ drive), and install to another drive partition. Verify the install directories are correct and select **Next** to continue



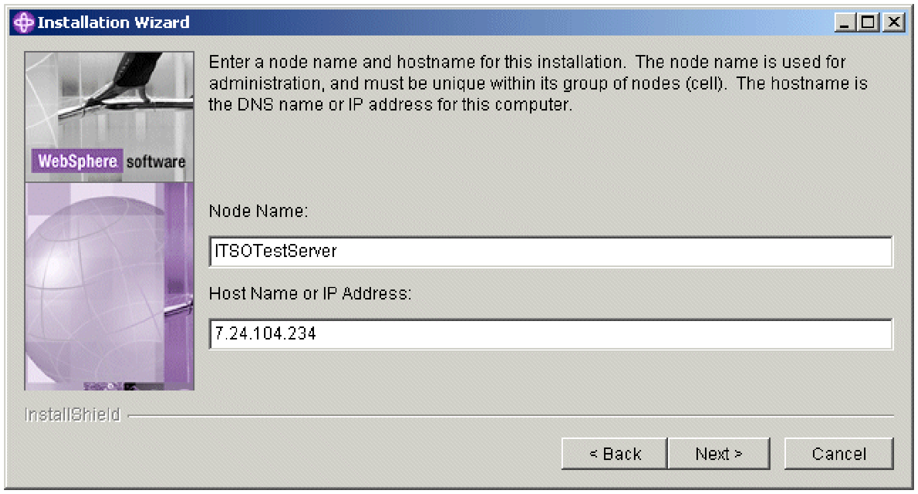
Product features will be installed in the following directories. To install to a different directory, click Browse and select a different directory.

IBM WebSphere Application Server, Version 5
C:\Program Files\WebSphere\AppServer
Space required: 392.9MB
Space available: 12883.3MB

IBM HTTP Server Version 1.3.26
C:\Program Files\IBMHttpServer
Space required: 18MB
Space available: 12883.3MB

Figure 9-9 Choosing the installation directory

The next dialog asks for the Node name which will be your default Node. It is recommended that the node name be different that the domain name. This recommendation is based on errors that will be generated if you install IBM WebSphere Application Server Network Deployment and there is a name conflict. More about this in troubleshooting. Type in the node name and hostname and select **Next** to continue.

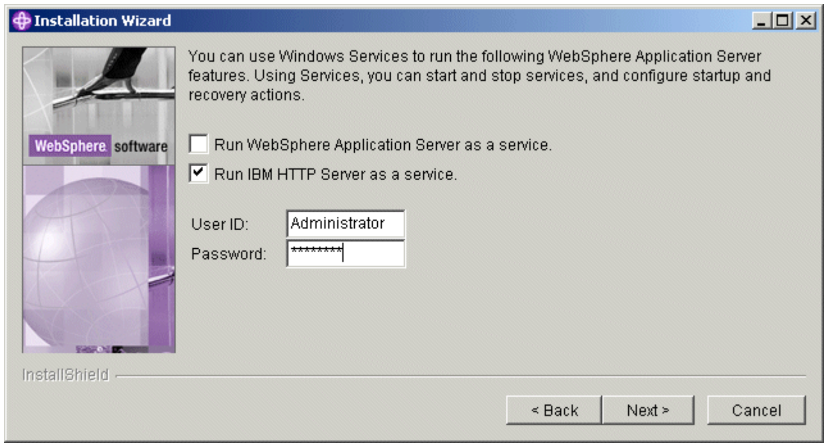


Enter a node name and hostname for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer.

Node Name: ITSOTestServer

Host Name or IP Address: 7.24.104.234

10. The next dialog will allow you to use Windos Services to start and stop services,and configure startup and recovery operations. Type in the desired user id and password. This will default to the administrative user that is logged in perfroming this installation.



You can use Windows Services to run the following WebSphere Application Server features. Using Services, you can start and stop services, and configure startup and recovery actions.

☐ Run WebSphere Application Server as a service.

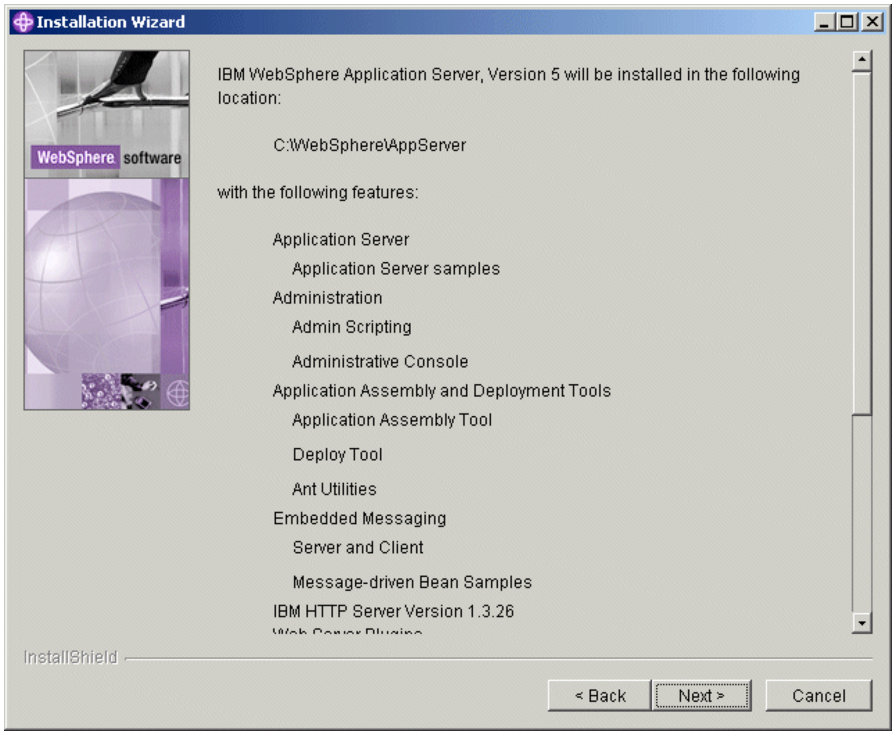
☑ Run IBM HTTP Server as a service.

User ID: Administrator

Password: ********

Figure 9-10 Setting up Windows Services

11. The last screen before installation begins shows a summary of all of the choices that were made in the custom installation. Selecting **Next** will begin the coping of files to the directory structures the user indicated during the installation. Once the selections have been reviewed, select **Next** to begin the installtion.



IBM WebSphere Application Server, Version 5 will be installed in the following location:

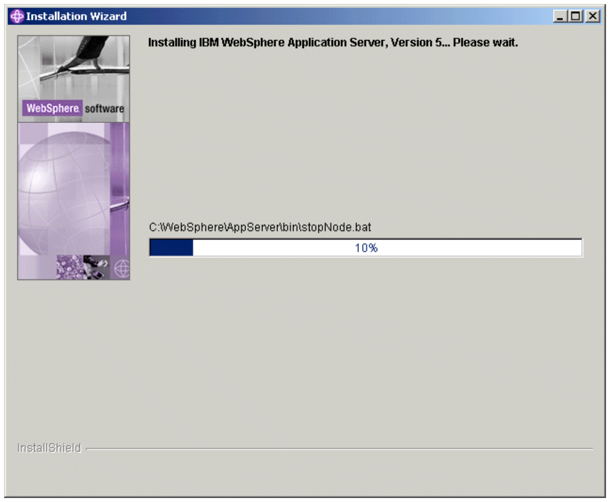
C:\WebSphere\AppServer

with the following features:

Application Server
Application Server samples
Administration
Admin Scripting
Administrative Console
Application Assembly and Deployment Tools
Application Assembly Tool
Deploy Tool
Ant Utilities
Embedded Messaging
Server and Client
Message-driven Bean Samples
IBM HTTP Server Version 1.3.26

Figure 9-11 Installation Summary

12. Once the installation begins, a progress bar will indicate the status of installtion.



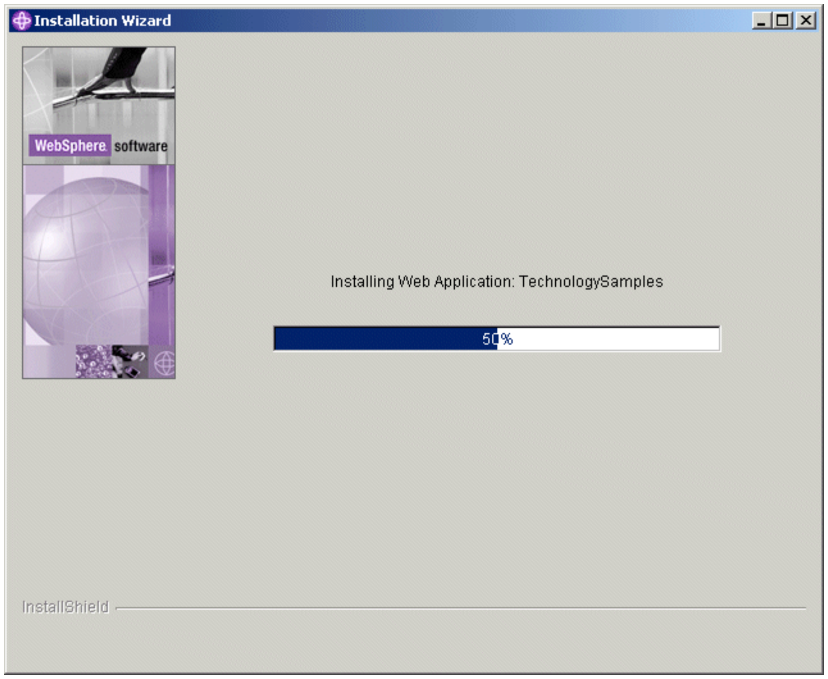
Installing IBM WebSphere Application Server, Version 5... Please wait.

C:\WebSphere\AppServer\bin\stopNode.bat

10%

Figure 9-12

13. When all of the files have been copied, the installer will begin the process of installing the sample applications that were selected during installation. It will install and start each application during this time and show a status bar indicating the progress.

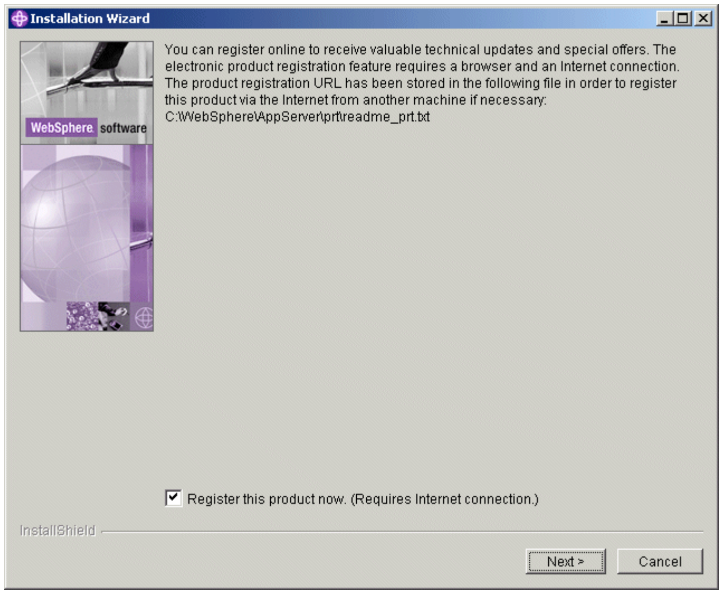


Installing Web Application: TechnologySamples

50%

Figure 9-13 Sample Application Installation

14. Once the installation is complete the installer will prompt the user to register.The user can choose to register the product immediately by checking the checkbox at the bottom of the dialog, or manually register at a later time by de selecting the checkbox.


Installation Wizard

You can register online to receive valuable technical updates and special offers. The electronic product registration feature requires a browser and an Internet connection. The product registration URL has been stored in the following file in order to register this product via the Internet from another machine if necessary:
C:\WebSphere\AppServer\prt\readme_prt.txt

☑ Register this product now. (Requires Internet connection.)

Figure 9-14 Register WebSphere Application Server

Congradulations, you have just install IBM WebSphere Application Server, Base. If you are insterested in upgrading the base to IBM WebSphere Application Server Network Deployment, continue to the next section. If you would like to verfity and test IBM WebSphere Application Server, Base; see Chapter 9.8.1, “Verifing the IBM WebSphere Application Server installation” on page 248.

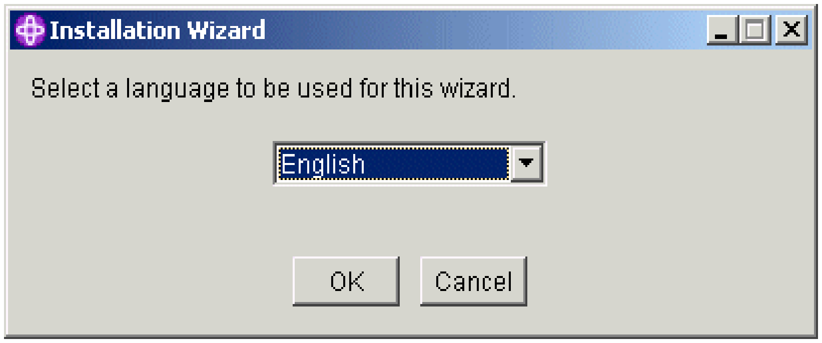
9.7 Install IBM WebSphere Application Server Network Deployment

The following procedure will use our existing base installation to upgrade to an Network Deployment intallation. It will first address the installation steps then outline the migration steps necessary to preserve any base configuration changes made prior to the ND installation.

The installation procedure for IBM WebSphere Application Server Network Deployment is as follows:

1. Log on as an administrator user in the local server domain (not part of the windows domain).
2. Insert the IBM WebSphere Application Server V5.0, Network Deployment CD.

3. Start the IBM WebSphere Application Server, Network Deployment installation by clicking install.bat on the root of the CD.
4. In the Installer dialog, select a language to be used for the installation, select **OK**.

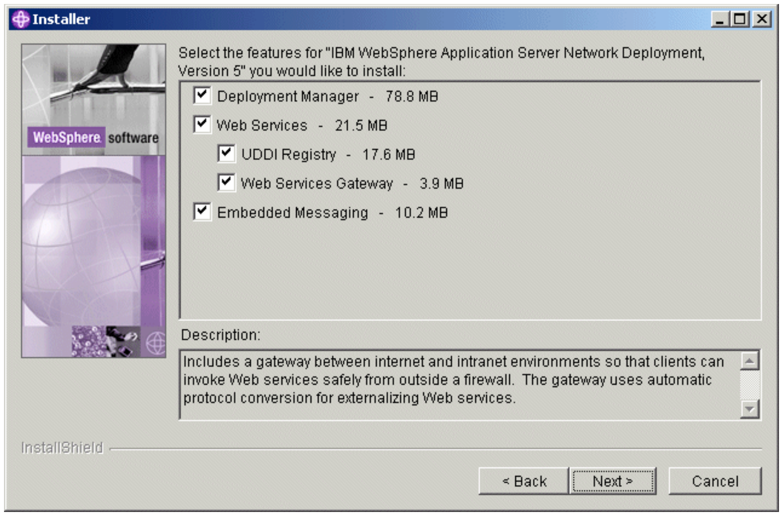


Select a language to be used for this wizard.

English

Figure 9-15 Choose language for installation

5. A welcome screen will be shown, read any warning and select **Next** to continue.
6. The License Agreement will be displayed. Read the License Agreement and if you agree to its terms, select the radio button next to “I agree to the terns of this license agreement”. Then select **Next** to continue.
7. The installer will then check the system for prereqisites, and display the feature that are available for install.Choose the features to install and select Next to continue.



Select the features for "IBM WebSphere Application Server Network Deployment, Version 5" you would like to install:

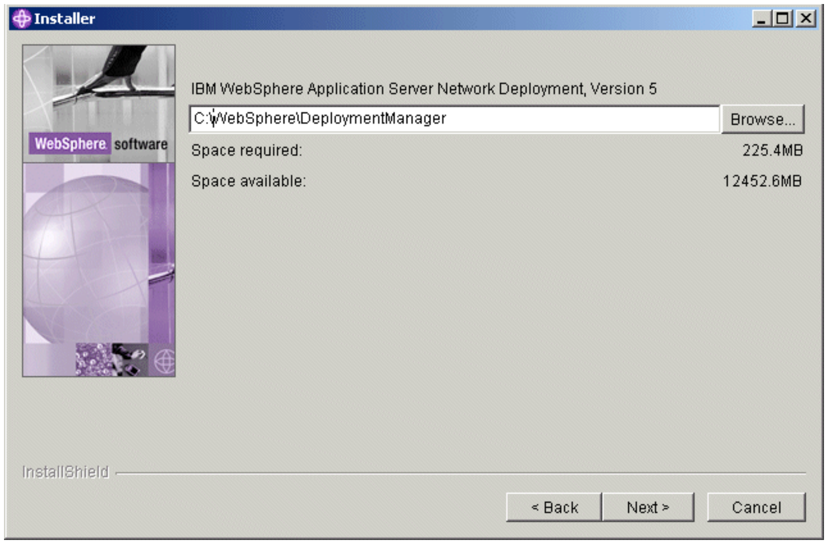
- [✓] Deployment Manager - 78.8 MB
- [✓] Web Services - 21.5 MB
  - [✓] UDDI Registry - 17.6 MB
  - [✓] Web Services Gateway - 3.9 MB
- [✓] Embedded Messaging - 10.2 MB

Description:

Includes a gateway between internet and intranet environments so that clients can invoke Web services safely from outside a firewall. The gateway uses automatic protocol conversion for externalizing Web services.

Figure 9-16 ND features to install

8. The next dialog will prompt for a drectory to install the Deployment Manager to. This should be a subdirectory under your base installation. Adjust the directory structure to add the Deployment Manager Directory as a subdirectory under WebSphere.



IBM WebSphere Application Server Network Deployment, Version 5

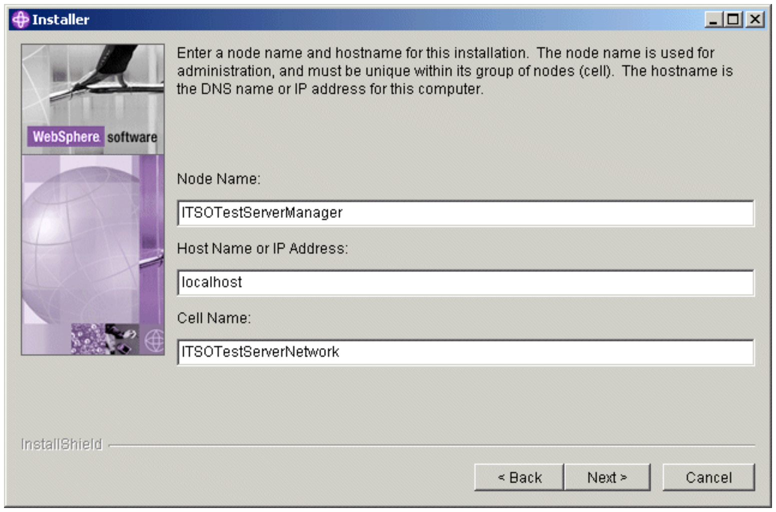
C:\WebSphere\DeploymentManager

Space required: 225.4MB

Space available: 12452.6MB

Figure 9-17 Network Deployment Installation Directory

9. The next dialog wll prompt for a Node name and host name for your installation. The node name must be unique among other nodes that may be installed, so be sure not to conflict with the node name selected for the base installation. If you select an conflicting node name, the installtion will continue with no warnings. However, you will get errors when trying to add a node to the deployment manager. (see troubleshooting for more details.), accept the deafualts (or create custom names) and select **Next** to continue.



Enter a node name and hostname for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer.

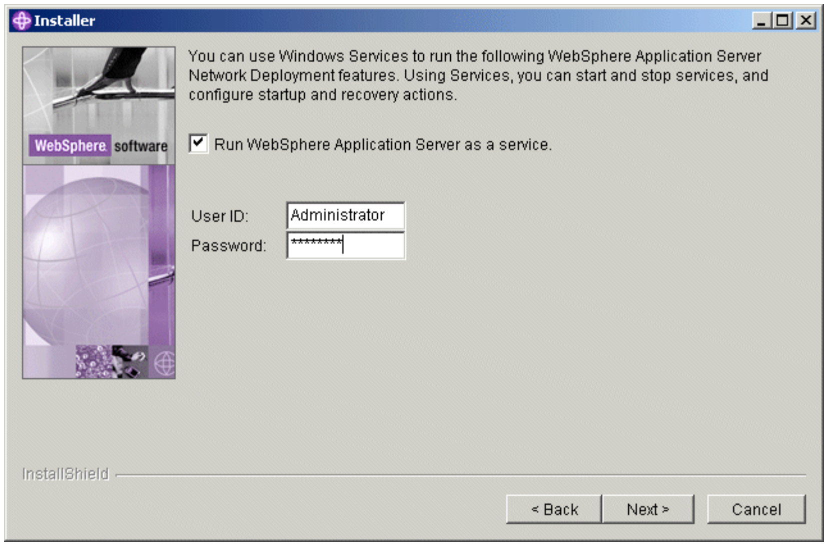
Node Name: ITSOTestServerManager

Host Name or IP Address: localhost

Cell Name: ITSOTestServerNetwork

Figure 9-18 Node and Host name selection

10. The next dialog allows the user to choose to run WebSphere Application Server as a Windows Service. Make the appropriate selection and select **Next** to continue.



You can use Windows Services to run the following WebSphere Application Server Network Deployment features. Using Services, you can start and stop services, and configure startup and recovery actions.

[✓] Run WebSphere Application Server as a service.

User ID: Administrator

Password: ********

Figure 9-19 Running ND as a Windows Service

11. The last dialog before installtion is the summary screen. The screen will detail the sections that were made during the install. Verufy that these selections are correct, and select **`Next`** to begin coping files to the machine.

Figure 9-20 Network Deployment Summary Screen

Congradualations, you have now installed IBM WebSphere Application Server Network Deployment.

9.8 Verifing the installation

9.8.1 Verifing the IBM WebSphere Application Server 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 is present.

WebSphere Windows services

Service name	Status	Startup mode	Log on as...
IBM WS Admin-Server 4.0	-	manual	wasadmin

Run verification through IVT

In order to very the installation of IBM WebSphere Application Server 5.0, the following tasks must be completed in order:

1. Use the Installation Verification Test.
2. Check the installation log.
3. Start the application server.
4. Start the Web server.

Use the Installation Verification Test

The Installation Verification Test is a tool which scans the product log files for errors and verifies core functionality of the product installation. This tool performs the following:

1. Start the application server, server1, if it has not started yet, and verify its status.
2. Verify servlet engine status
3. Verify JSP status
4. Verify EJB status

The installation program automatically starts the First Steps program. Once the dialog diplays use the following steps:

To start the Installation Verification Test from the First Step program:

1. Click **Verify Installation**.
2. The Installation Verification Test starts the verification. After the verification is completed, press **Enter**.

The Installation Verification Test can be started by the following steps(after installation):

1. Log in as root on the WebSphere machine.
2. Launch Start->Programs->IBM WebSphere->IBM WebSphere 5.0-> First Steps.
3. Select Verify Installation

Check the log file, <WAS_BASE_HOME>/logs/ivt.log, for the result.

Figure 9-21 First Steps

Check the installation log

Check the following installation log files under <WAS_BASE_HOME>/logs directory for errors.

Table 9-3 Installation log files

Component	File
WebSphere Application Server	log.txt
IBM HTTP Server	ihs_log.txt
Default Application	installDefaultApplication.log
Sample Application	installSamples.log
Admin Console	installAdminConsole.log
MDB Samples Application	installMessagingSamples.log
Pet Store Application	installPetStore.log
Plants By WebSphere Application	installPlantsByWeb.log
Technology Samples Application	installTechSamples.log
IVT Application	installIVTApp.log

The installation wizard also creates the optional_log.txt file in the $TEMP directory. Check this file if necessary.

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:

```
Alias /WSsamples /usr/WebSphere/AppServer/WSsamples
Alias /IBMWebAS/ /usr/WebSphere/AppServer/web/
LoadModule ibm_app_server_http_module
/usr/WebSphere/AppServer/bin/mod_ibm_app_server_http.so
WebSpherePluginConfig /usr/WebSphere/AppServer/config/plugin-cfg.xml
```

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 Installation Verification Test starts the application server. But the server can be started by either of the following ways:

- Start from First Steps window

  Click **Start the Server**

Check the log files, SystemOut.log and SystemErr.log, under <WAS_BASE_HOME>/logs/server1 directory. The following message should be appeared in SystemOut.log for its successful start.

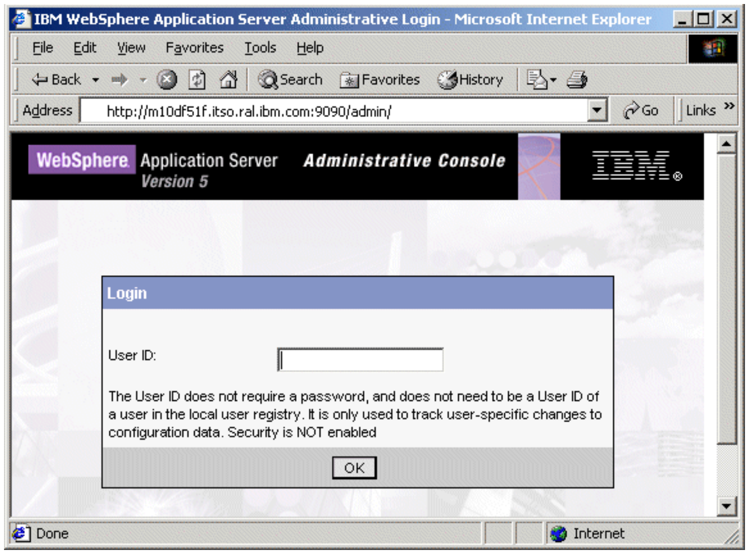
```
[9/12/02 17:14:40:617 EDT] 425dc76b WsServer      A WSVR0001I: Server
server1 open for e-business
```

To access the admin console:

1. Using a Web browser, request the following URL:

   `http://<hostname>:9090/admin`

   A window similar to the one shown is displayed in the browser.

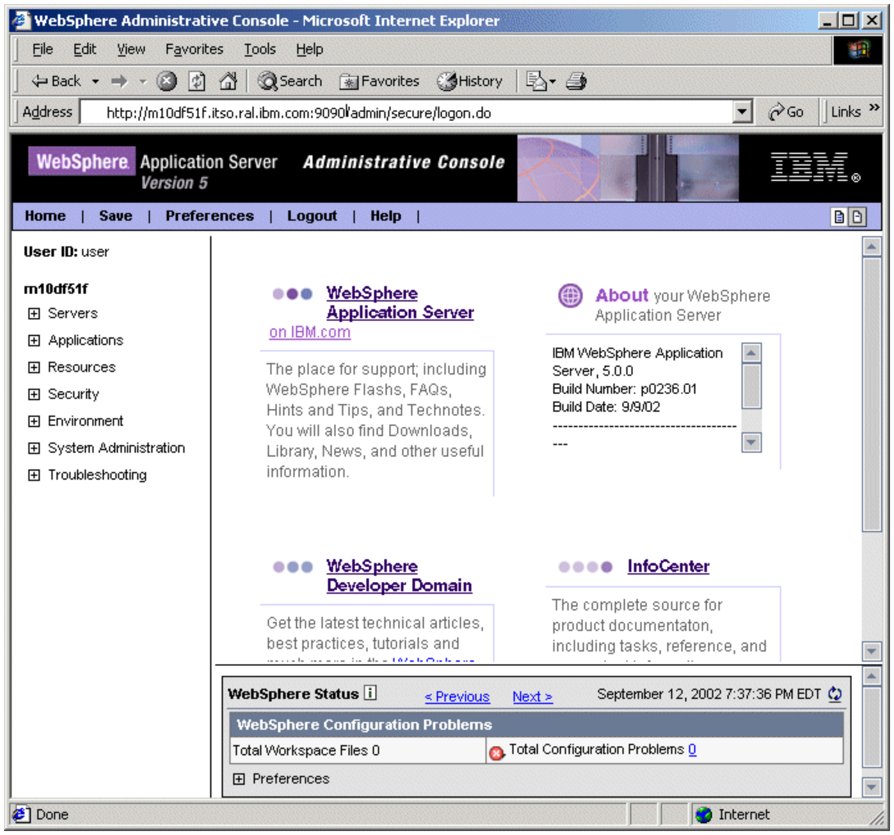

IBM WebSphere Application Server Administrative Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9090/admin/
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Login
User ID:
OK
Done
Internet

The User ID does not require a password, and does not need to be a User ID of a user in the local user registry. It is only used to track user-specific changes to configuration data. Security is NOT enabled

Figure 9-22 Admin Console Login

Note: If the port number of the admin console is changed, the correct port number should be specified.

2. Enter User ID to login. It is for tracking purpose only, so any user id is acceptable. In this example, **user** is used as User ID. The following window will appear in the browser.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
User ID: user
m10df51f
Servers
Applications
Resources
Security
Environment
System Administration
Troubleshooting
WebSphere Application Server on IBM.com
About your WebSphere Application Server
WebSphere Developer Domain
InfoCenter
WebSphere Status
< Previous
Next >
WebSphere Configuration Problems
Preferences
Done
Internet

The place for support; including WebSphere Flashes, FAQs, Hints and Tips, and Technotes. You will also find Downloads, Library, News, and other useful information.

IBM WebSphere Application Server, 5.0.0
Build Number: p0236.01
Build Date: 9/9/02

Get the latest technical articles, best practices, tutorials and

The complete source for product documentaton, including tasks, reference, and

September 12, 2002 7:37:36 PM EDT

Total Workspace Files 0
Total Configuration Problems 0

Figure 9-23 Admin Console

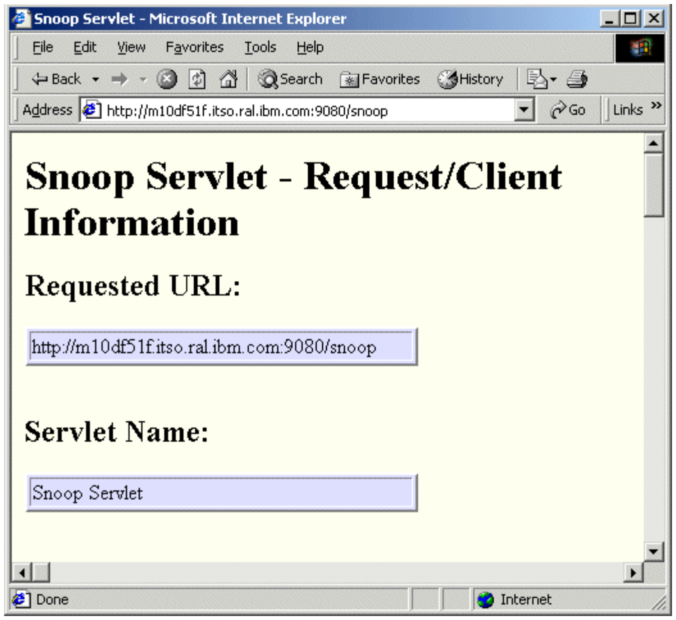
Access WebSphere Default Server.

To access the default application:

1. Using a Web browser, request the following URL:

```
http://<hostname>:9080/snoop
```

2. A window similar to the one shown in Figure below is displayed in the browser.


Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9080/snoop
Go Links
Done
Internet

Snoop Servlet - Request/Client Information

Requested URL:

http://m10df51f.itso.ral.ibm.com:9080/snoop

Servlet Name:

Snoop Servlet

Figure 9-24 Snoop Servlet

Regenerate Web server plug-in settings.

If you decide to make changes to the applicaitons installed on WebSphere, you will have to let the web server know of these changes so that requests for these new applications will get routed to the application server correctly. The following steps outline what needs to be done to generate the plug-in configuration file.

1. Using a web browser, open the Adminsitrator's Console
2. Select Environment
3. The first item is Plug-in File Generation., Select this option.
4. Slect OK to generate the plug.
5. The next dialog will allow you to view the HTTP plug-in configuration file. Select View to see the file and confirm your changes.

Restart 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"
```

9.8.2 Testing the IBM WebSphere Application Server Network Deployment

9.8.3 WebSphere Samples

The WebSphere samples gallary proviedes the following samples:

- The Plants by WebSphere application, which demonstrates several J2EE functions, using an online store that specializes in plant and garden tool sales.
- Technology Samples, which showcase enterprise beans, servlets, JavaServer Pages technology, message-driven beans, and J2EE application client.
- The Java Pet Store Application, which demonstrates J2EE technology, using an online pet store.

9.9 Install the Web Server plugin on a remote 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.

9.9.1 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.

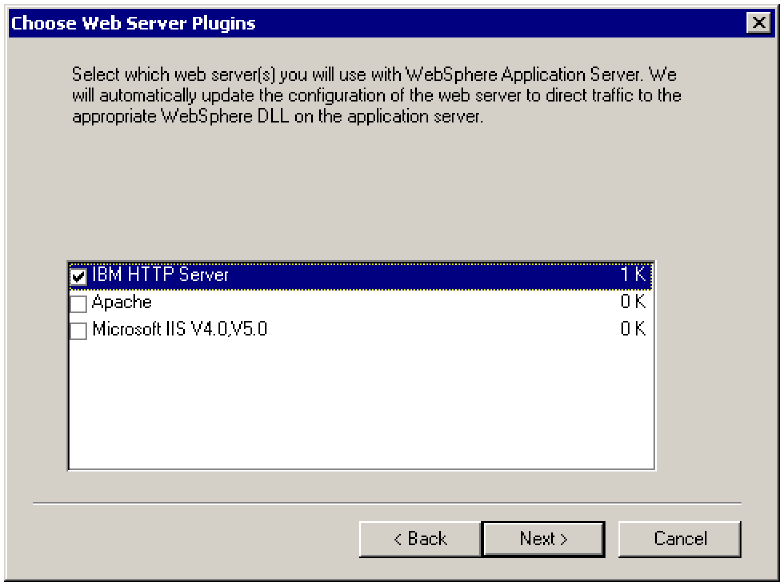
To stop the IBM HTTP Server processes, issue the command:

```
D:\> net stop "IBM HTTP Server"
```

9.9.2 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 9-25 on page 256, select only the **IBM HTTP Server** option and then click **Next**.


Choose Web Server Plugins

Select which web server(s) you will use with WebSphere Application Server. We will automatically update the configuration of the web server to direct traffic to the appropriate WebSphere DLL on the application server.

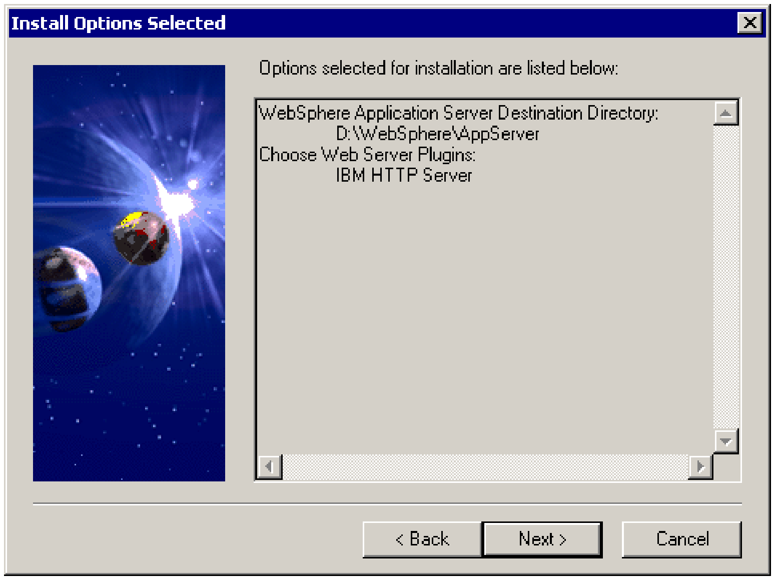
✓ IBM HTTP Server	1 K
☐ Apache	0 K
☐ Microsoft IIS V4.0,V5.0	0 K

Figure 9-25 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 9-26 on page 257, check that the selected components are correct. If yes, click **Next** to start the installation process.



Options selected for installation are listed below:

```
WebSphere Application Server Destination Directory:
            D:\WebSphere\AppServer
Choose Web Server Plugins:
            IBM HTTP Server
```

Figure 9-26 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**.

9.9.3 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.

9.9.4 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

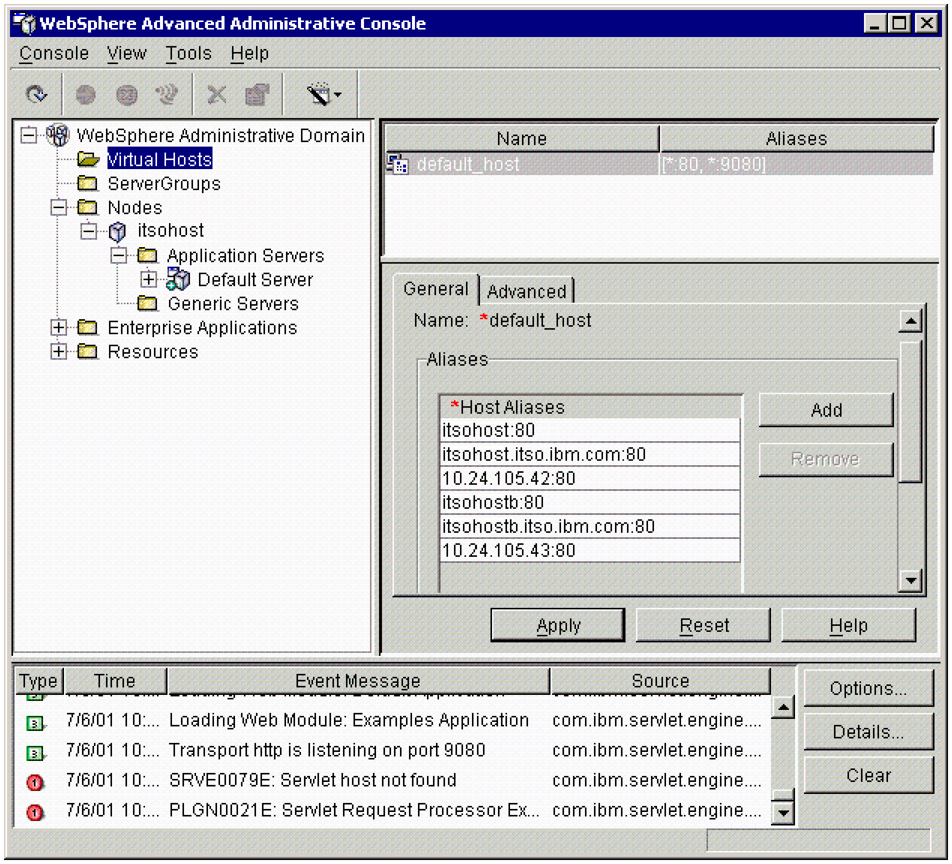
WebSphere uses virtual hosts to map HTTP requests for a particular host name and port number to selected Web applications. To map requests from the remote Web server to the required virtual host, complete the following steps:

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 9-27:

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



Name	Aliases
default_host	[*:80, *:9080]

Name: *default_host

Aliases

*Host Aliases
itsohost:80
itsohost.itso.ibm.com:80
10.24.105.42:80
itsohostb:80
itsohostb.itso.ibm.com:80
10.24.105.43:80

Type	Time	Event Message	Source
	7/6/01 10:...	Loading Web Module: Examples Application	com.ibm.servlet.engine....
	7/6/01 10:...	Transport http is listening on port 9080	com.ibm.servlet.engine....
	7/6/01 10:...	SRVE0079E: Servlet host not found	com.ibm.servlet.engine....
	7/6/01 10:...	PLGN0021E: Servlet Request Processor Ex...	com.ibm.servlet.engine....

Figure 9-27 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"
```

9.9.5 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.

screen capture here....

9.10 Add a new WebSphere node into existing domain

9.11 Configure WebSphere HTTP trasport 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.

9.11.1 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

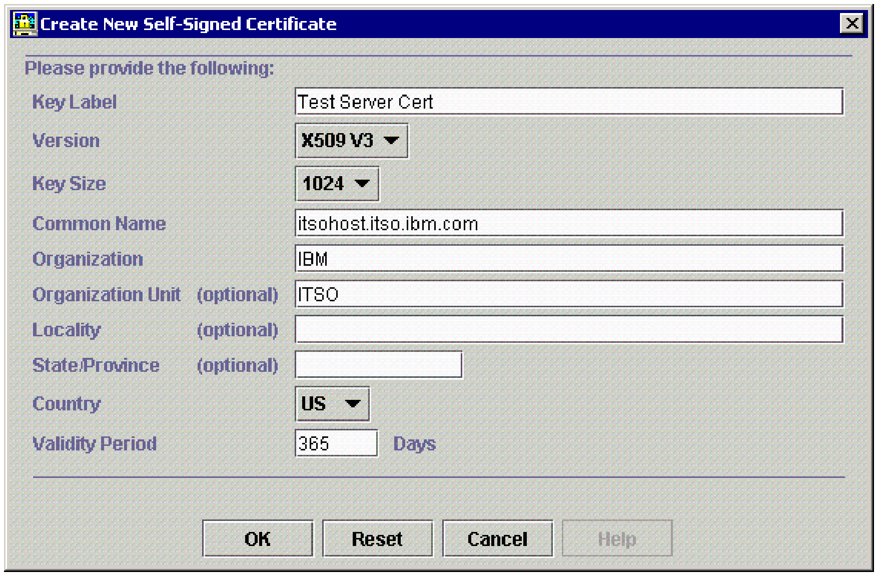
To create a new self-signed certificate, perform the following steps on the WebSphere server machine:

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 9-28 on page 264, 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`



Please provide the following:

Key Label		Test Server Cert
Version		X509 V3
Key Size		1024
Common Name		itsohost.itso.ibm.com
Organization		IBM
Organization Unit	(optional)	ITSO
Locality	(optional)	
State/Province	(optional)	
Country		US
Validity Period		365 Days

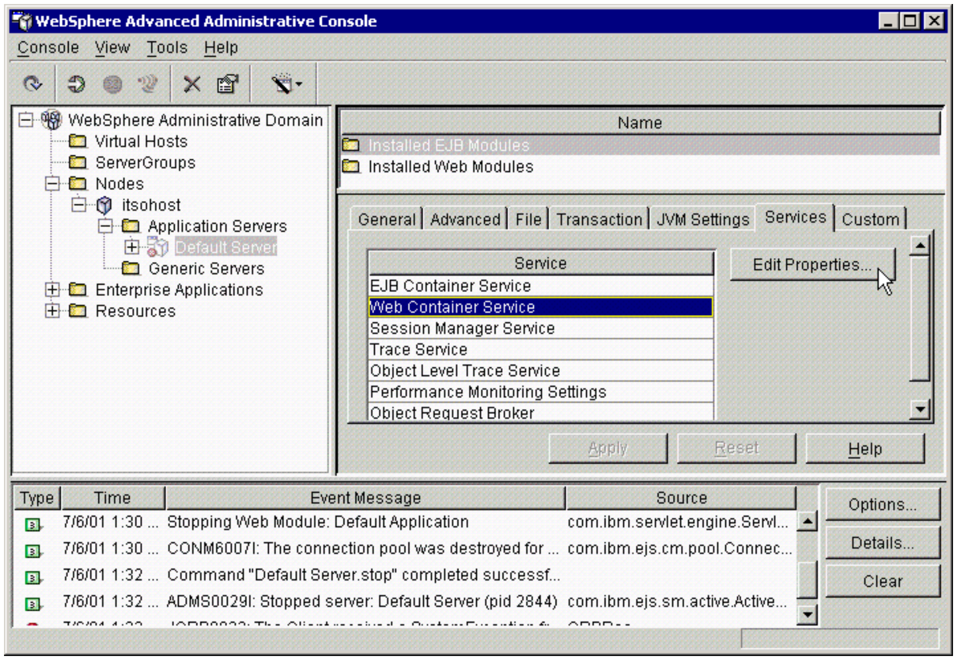
Figure 9-28 Self-signed certificate settings

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 9-29 on page 265.

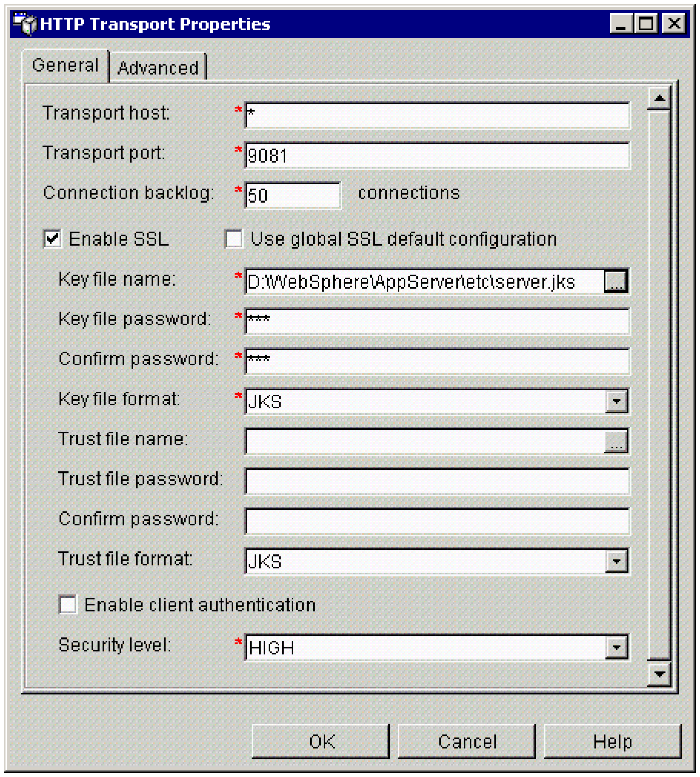


Service
EJB Container Service
Web Container Service
Session Manager Service
Trace Service
Object Level Trace Service
Performance Monitoring Settings
Object Request Broker

Type	Time	Event Message	Source
	7/6/01 1:30 ...	Stopping Web Module: Default Application	com.ibm.servlet.engine.Servl...
	7/6/01 1:30 ...	CONM6007I: The connection pool was destroyed for ...	com.ibm.ejs.cm.pool.Connec...
	7/6/01 1:32 ...	Command "Default Server.stop" completed successf...	
	7/6/01 1:32 ...	ADMS0029I: Stopped server: Default Server (pid 2844)	com.ibm.ejs.sm.active.Active...

Figure 9-29 Select Web Container service settings

6. In the Web Container Service window, shown in Figure 9-30 on page 266, 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`



Transport host: * *

Transport port: * 9081

Connection backlog: * 50 connections

[✓] Enable SSL

[ ] Use global SSL default configuration

Key file name: * D:\WebSphere\AppServer\etc\server.jks

Key file password: * ***

Confirm password: * ***

Key file format: * JKS

Trust file name:

Trust file password:

Confirm password:

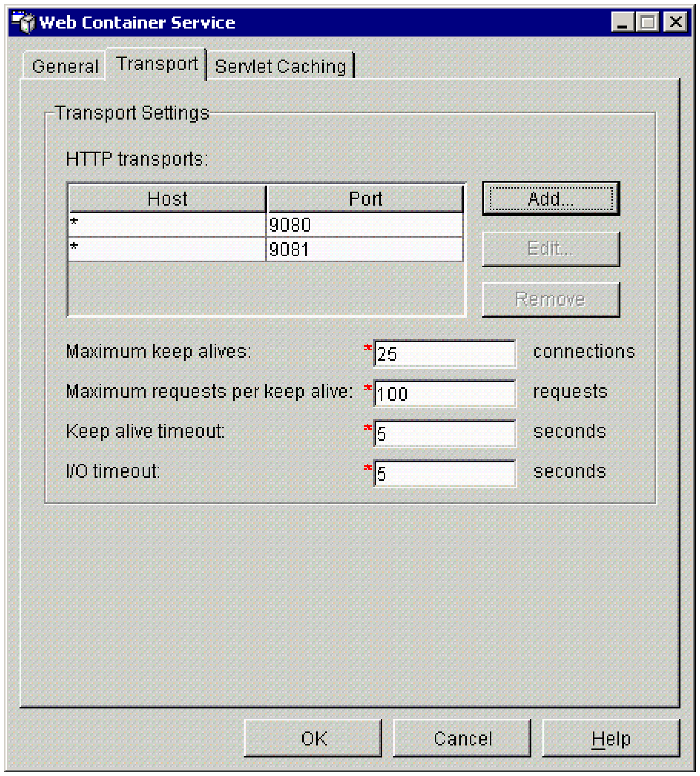
Trust file format: JKS

[ ] Enable client authentication

Security level: * HIGH

Figure 9-30 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 9-31 on page 267. Click **OK**.



Transport Settings

HTTP transports:

Host	Port
*	9080
*	9081

Maximum keep alives: * 25 connections

Maximum requests per keep alive: * 100 requests

Keep alive timeout: * 5 seconds

I/O timeout: * 5 seconds

Figure 9-31 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 9-32 on page 267 should be displayed.

Figure 9-32 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**:

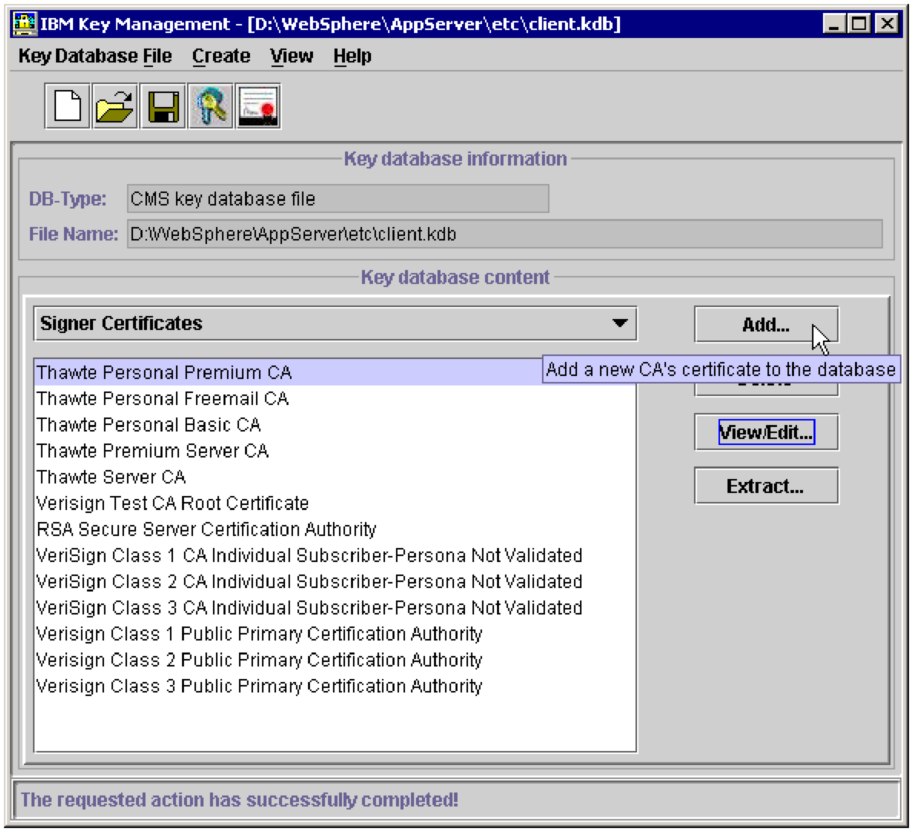
   ```
   The password has been encrypted and saved in the file:
   <plugin_install_path>\etc\client.sth
   ```

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 9-33.



DB-Type: CMS key database file
File Name: D:\WebSphere\AppServer\etc\client.kdb

Add a new CA's certificate to the database

```
Thawte Personal Premium CA
Thawte Personal Freemail CA
Thawte Personal Basic CA
Thawte Premium Server CA
Thawte Server CA
Verisign Test CA Root Certificate
RSA Secure Server Certification Authority
VeriSign Class 1 CA Individual Subscriber-Persona Not Validated
VeriSign Class 2 CA Individual Subscriber-Persona Not Validated
VeriSign Class 3 CA Individual Subscriber-Persona Not Validated
Verisign Class 1 Public Primary Certification Authority
Verisign Class 2 Public Primary Certification Authority
Verisign Class 3 Public Primary Certification Authority
```

The requested action has successfully completed!

Figure 9-33 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.

9.11.2 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 263 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 269 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 9-34, 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 9-34 SSL settings for client authentication enabled mode

9.11.3 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"
```

9.12 Install WebSphere Application Server in 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 9-4 on page 273.
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 9-4 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`

Setup option	Parameter description

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=`

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=`

Setup option	Parameter description

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=`

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`

Setup option	Parameter description

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=`

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.

9.13 Troubleshooting

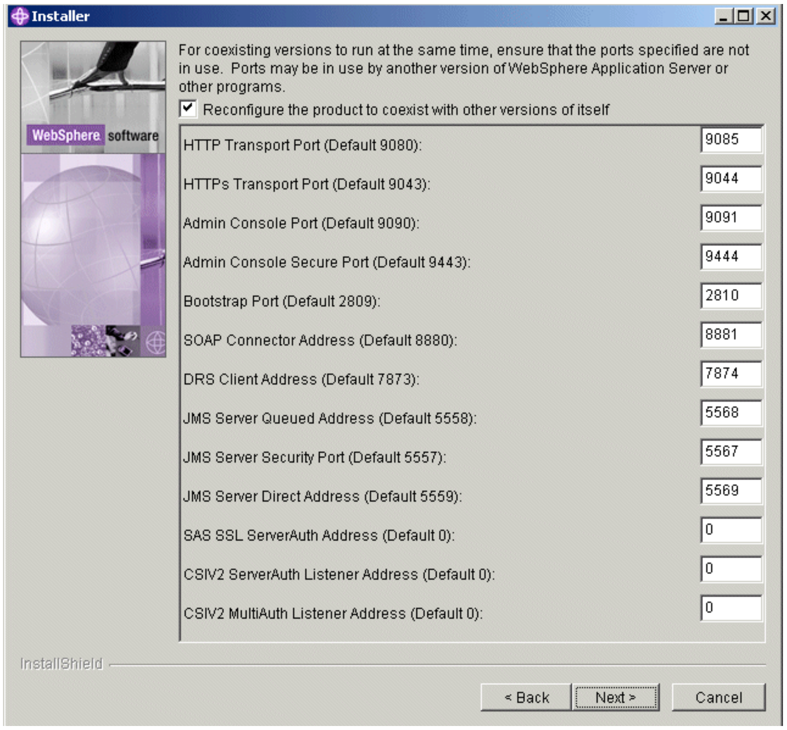
9.13.1 Problems with the web server

9.13.2 Problems with IBM WebSphere Application Server, base

9.13.3 Problems with IBM WebSphere Application Server Network Deployment

9.14 Multiple WebSphere Installations

Author Comment: Below is a screen capture that will diplay if there is an exiting ND installtion on a machine, and you are attempting a second ND installation.



For coexisting versions to run at the same time, ensure that the ports specified are not in use. Ports may be in use by another version of WebSphere Application Server or other programs.

[✓] Reconfigure the product to coexist with other versions of itself

HTTP Transport Port (Default 9080):	9085
HTTPs Transport Port (Default 9043):	9044
Admin Console Port (Default 9090):	9091
Admin Console Secure Port (Default 9443):	9444
Bootstrap Port (Default 2809):	2810
SOAP Connector Address (Default 8880):	8881
DRS Client Address (Default 7873):	7874
JMS Server Queued Address (Default 5558):	5568
JMS Server Security Port (Default 5557):	5567
JMS Server Direct Address (Default 5559):	5569
SAS SSL ServerAuth Address (Default 0):	0
CSIV2 ServerAuth Listener Address (Default 0):	0
CSIV2 MultiAuth Listener Address (Default 0):	0

Figure 9-35

10

AIX installation steps

Per Jeff .. need to mention mode (64 bit / 32 bit compatible) .. no info available

This chapter provides detailed procedures for installing, configuring, and verifying a number of the scenarios described in Chapter 8, “Installation approach” on page 177, for an environment consisting of the following components:

- Operating system - AIX 5.1
- Web server - IBM HTTP Server
- Application server - WebSphere Application Server
- Deployment manager - WebSphere Application Server Network Deployment

10.1 Planning

The procedures described in this chapter 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
- Installing AIX
- Installing a Web browser
- Installing the Web server
- Installing WebSphere Application Server V5.0
- Installing the WebSphere plug-in on a remote Web server
- Installing WebSphere Application Server Network Deployment V5.0
- Adding a WebSphere node into an existing cell
- Configuring the WebSphere HTTP transport for SSL
- Installing WebSphere Application Server in silent mode
- Installing WebSphere Application Server Network Deployment in silent mode

Author Comment (for carla): May need to customize this for our environment. I think we are not supposed to give hard prereqs, but point to the shipped info and then the Web site for the latest. ... will check with Tricia

10.1.1 Hardware and software prerequisites

At the time of this book, IBM WebSphere Application Server had the following hardware and software requirements.

Hardware

- RS/6000 or RS/6000 SP workstation running AIX.
- 604e RS/6000 at 375 MHz, or faster
- Support for an appropriate network interface
- CD-ROM drive
- 384 MB memory (minimum), 512 MB recommended
- 200 MB diskspace (minimum) for WebSphere Application Server
- 100 MB diskspace (minimum) for WebSphere Application Server Network Deployment
- 50 MB diskspace (minimum) for IBM HTTP Server

Software

- AIX 4.3.3.09 (APAR IY19277) or V5.1 Maintenance Level 1
- IBM WebSphere Application Server V5.0

- IBM IBM WebSphere Application Server Network Deployment V5.0 (optional)
- IBM HTTP Server 1.3.26
- IBM GSKit 5.0.4.79

For the latest in hardware and software requirements: When planning for installation, be sure to check the following for the latest in requirements:

- <installation directory>/waspc/prereqChecker.xml
- find website here ..

10.1.2 Software used in our test environment

We used the following software in our test environment:

- AIX 5.1 Maintenance Level 2
- IBM WebSphere Application Server V5.0
- IBM WebSphere Application Server Network Deployment V5.0
- IBM HTTP Server 1.3.26
- IBM GSKit 5.0.4.79 (included in IBM HTTP Server package)
- IBM JDK 1.3.1 (included in the WebSphere Application Server package)

Note: Other Web servers may be used, as documented in the Product Installation Guide. .. need ref here ..

Product installation roots

The variables listed in Table 10-1 on page 281 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_BASE_HOME>	/usr/WebSphere/AppServer	WebSphere Application Server
<WAS_ND_HOME>	/usr/WebSphere/DeploymentManager	WebSphere Application Server Network Deployment
<http_server_install_path>	/usr/IBMHttpServer	IBM HTTP Server

Author Comment (Arihiro): the table above lists a network deployment plug-in and an app server plug-in. Are they two different things?
=> Since these variables are not used in this chapter, they are removed.

Note: Both WAS_BASE_HOME and WAS_ND_HOME are defined as WAS_HOME in setupCmdLine.sh under bin directory of their installation directories. In order to distinguish the directory, different variable names are assigned in this document.

10.1.3 Hardware used in our test environment

This section describes the hardware used within our test WebSphere V5.0 environment on AIX.

- Server 1
  - IBM RS/6000 44P Model 170
  - 1 450 MHz CPU
  - 2 GB RAM
  - 18 GB Hard Disk
  - 1 IBM 10/100 Mbps Ethernet PCI Adapter
- Server 2
  - IBM RS/6000 44P Model 170
  - 1 450 MHz CPU
  - 2 GB RAM
  - 18 GB Hard Disk
  - 1 IBM 10/100 Mbps Ethernet PCI Adapter

10.2 Install AIX

Prior to installing any of the WebSphere components, the proper level of the operating system must be installed.

- AIX 5.1 maintenance level
- AIX 5.1 additional filesets required by WebSphere V5.0
- Extra tools

10.2.1 AIX 5.1 maintenance level

The WebSphere V5.0 minimum supported level of AIX 5.1 is maintenance level 1.

1. Determine whether the server currently has the required maintenance release (or newer) installed by issuing the following command:

```
# oslevel -r
```

2. The command will generate the following output.:

```
5100-02
```

The last two digits, 02 in this example, are the maintenance level of the installed AIX. In our test environment, the maintenance level 2 is used. If the digits are less than the required maintenance level, AIX should be upgraded to the required level. 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 site:

`http://techsupport.services.ibm.com/server/nav?fetch=fdca`

or the following IBM FTP site:

`ftp://ftp.software.ibm.com/aix/fixes/51/ml`

10.2.2 AIX 5.1 additional filesets required by WebSphere V5.0

Basically, there are no additional filesets required for the installation of WebSphere Application Server. However, the following fileset is required for Japanese character display.

- X11.fnt.ucs.ttf

Determine whether the fileset is currently installed by using the following command:

```
# lslpp -L | grep <fileset>
```

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.

Note: The required fileset information comes from the prereqChecker.xml tool. AIX filesets can be obtained from its product CDs or the following IBM site:

`http://techsupport.services.ibm.com/server/nav?fetch=fdca`

10.2.3 Extra tools

Table 10-5 lists a utility that may be required during the install of the software components but which is not installed by default on AIX.

Table 10-2 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 the Web server and plugin

Note: This section assumes that you are installing the IBM HTTP Server on a machine separate from the WebSphere Application Server. If you are installing both the IBM HTTP Server and WebSphere Application Server on the same machine, continue to 10.4, “Install WebSphere Application Server” on page 295.

This section provides detailed instructions for installing, configuring, and verifying IBM HTTP Server V1.3.26 for AIX. The section is organized into the following tasks:

- Preinstallation tasks.
- Installing IBM HTTP Server.
- Configuring IBM HTTP Server.
- Verifying the IBM HTTP Server installation.
- Enabling SSL encryption for requests (optional).

Important: In this chapter, the configuration and verification of the HTTP Administration Server are mentioned. The HTTP Administration Server is a powerful GUI tool used to administer the IBM HTTP Server. However, you may also perform administration by directly editing the httpd.conf configuration file. If this is the preferred method, you can skipt the sections referring to the HTTP Administration Server.

10.3.1 Preinstallation tasks (check IP ports)

Prior to installing IBM HTTP Server V1.3.26, you should verify that the following default IP ports are not in use by another program:

- 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 and plugin

Author Comment: According to the InfoCenter, an installation script named InstallIHS.sh will be provided. Since there is no such script included in beta 4, this section should be written after the script is provided. In order to write the following section, the same steps written in “Install WebSphere Application Server” on page 297 with only selecting IBM HTTP Server is used.

Author Comment (Arihiro): This section should be written using the launchpad.bat. You need both IBM HTTP Server and the plug-in so you might as well do both now. => Done

IBM HTTP Server can be installed by the installation program offered by WebSphere Application Server. To install IBM HTTP Server using the GUI installer interface of WebSphere Applicatrion Server, 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 V5.0 CD-ROM into the CD-ROM drive and mount the CD.

   ```
   # mount -r -v cdrfs /dev/cd0 /mnt
   ```

Author Comment: (Carla) verify the CD name

4. Change the directory to the installation root.
5. Ensure the DISPLAY and TERM environment variables are properly set. They can be set by the following commands:

   ```
   export DISPLAY=<IP address of host>:0.0
   export TERM=vt100
   ```

6. Run the LaunchPad.sh installation script:

   ```
   # ./LaunchPad.sh
   ```

7. In the Select a Language window, select a language and click **OK**. In this example, `English` is selected.
8. In the LaunchPad main window, click **Install the product**.

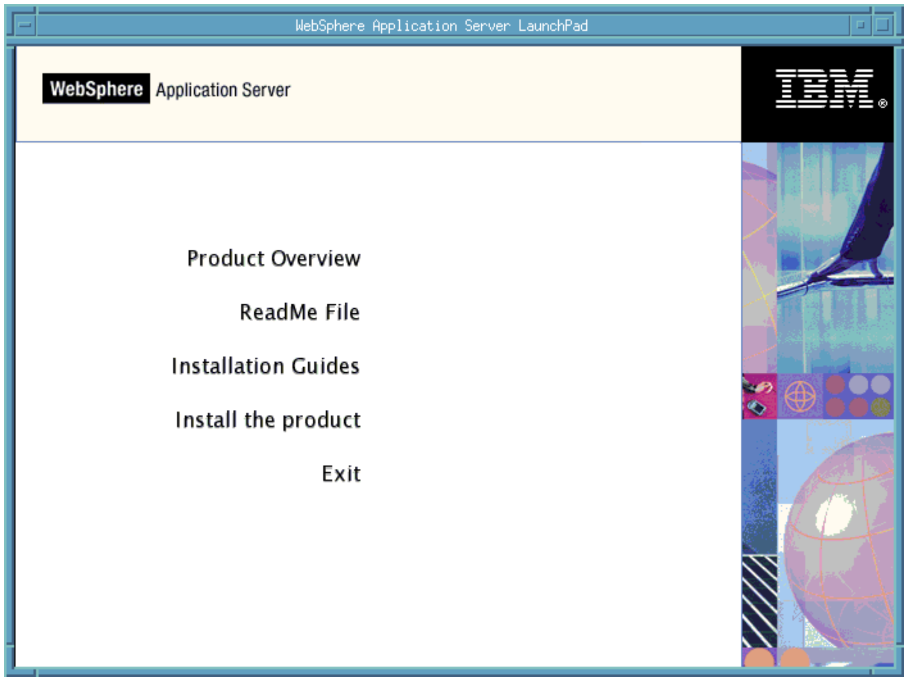


Figure 10-1 LaunchPad main window (IHS)

9. In the Installation Wizard window, select a language and click **OK**. In this example, `English` is selected.
10. In the Welcome window, click **Next**.
11. In the Software License Agreement window, read the licence agreement, select **Accept**, and click **Next**.
12. In the Operating System Level Check window, Click **Next**.
13. IIn the Installation Options window Select **Custom** and click **Next**. The next window will allow you to select the features to install. Select IBM HTTP Server and its HTTP plugins.

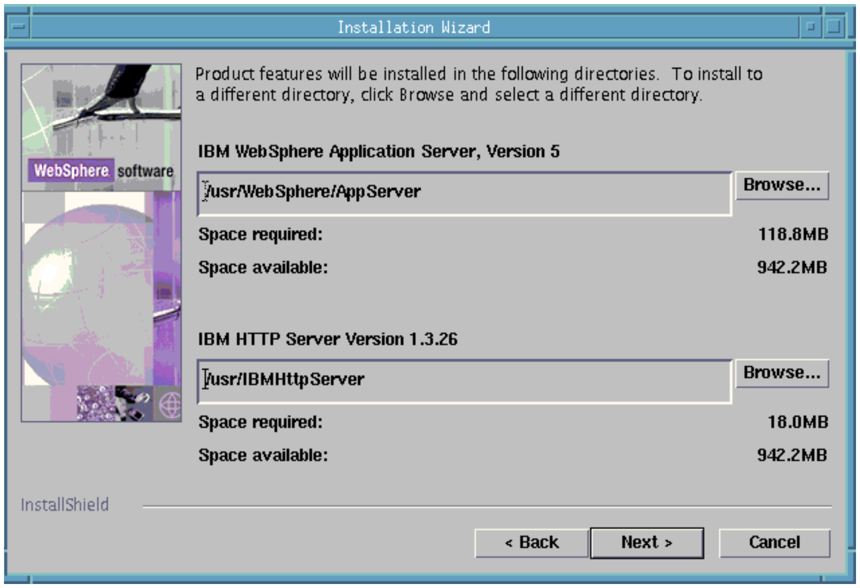
Note: Although not listed in the window, the IBM JDK 1.3.1 is automatically installed under the WebSphere installation directory. There is no need to separately install a JDK for use by the Web server plug-ins.

14. Click **Next**.

15. In the Install directory window, enter the directories for WebSphere Application Server and IBM HTTP Server. In this example, the default installation directories listed in Table 10-5 are used. Click **Next**:

Table 10-3 default installation directory

component	installation directory
WebSphere Application Server	/usr/WebSphere/AppServer
IBM HTTP Server	/usr/IBMHttpServer



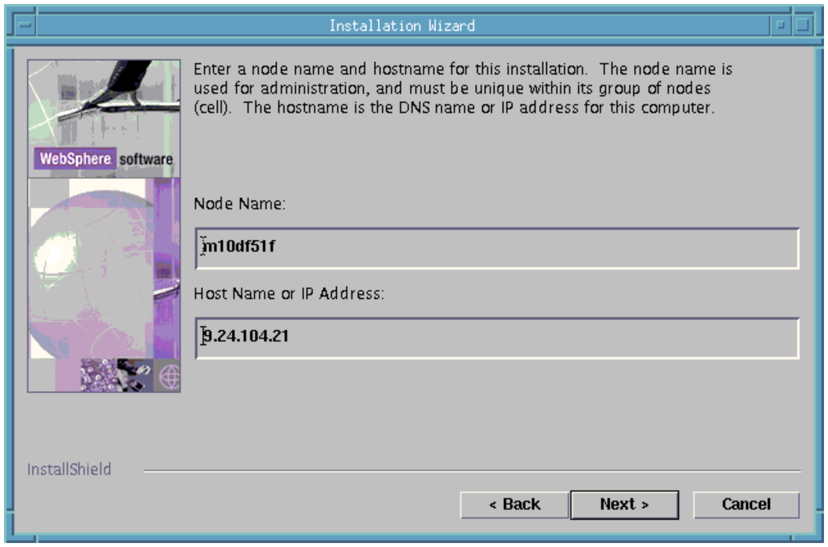
Product features will be installed in the following directories. To install to a different directory, click Browse and select a different directory.

IBM WebSphere Application Server, Version 5
/usr/WebSphere/AppServer
Space required: 118.8MB
Space available: 942.2MB

IBM HTTP Server Version 1.3.26
/usr/IBMHttpServer
Space required: 18.0MB
Space available: 942.2MB

Figure 10-2 Install directory window (IHS)

16. In the node name and hostname window, enter a node name and host name or IP Address. The values will automatically default to the host name and IP address for your system. Click **Next**.



Enter a node name and hostname for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer.

Node Name:

m10df51f

Host Name or IP Address:

9.24.104.21

Figure 10-3 Node name and hostname window

17. In the confirmation window, confirm the selected features will be installed, and click **Next**.
18. If you would like to register the product at this time, proceed through the registration panels. If not, deselect **Register** and click **Next**. Registration can be done later using the First Steps window.
19. In the Finish window, click **Finish**. The installation of the WebSphere Application Server is now complete. The installation program automatically starts First Steps Program.

10.3.3 Configure the IBM HTTP Server

After the installation of IBM HTTP Server V1.3.26, 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 -c -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: Both user ID and password for HTTP Administration Server are created here. They are used only in order to login to the HTTP Administration Server. The user ID created here is not AIX user ID.

Author Comment (Arihiro): Does this create a user or only update the password ?? => Done. See Note above.

Update httpd.conf

The IBM HTTP Server configuration file httpd.conf must be updated to reflect the fully qualified host name of the server. 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 10-4 on page 289.

Table 10-4 httpd.conf required settings

Setting	Required value...
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 configuration changes for httpd.conf file 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 stop
# ./apachectl start
```

Restart the HTTP Administration Server

The HTTP Administration Server must be restarted in order to recognize the configuration changes of httpd.conf:

Author Comment (Arihiro): should this say "must be restarted before the administration server recognizes the changes to the httpd.conf file? This sounds as if the httpd.conf change is not effective until the admin server is restarted.
=> Done.

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <http_server_install_path>/bin
# ./adminctl stop
# ./adminctl start
```

10.3.4 Verify the IBM HTTP Server

In order to verify the IBM HTTP Server V1.3.26 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
   ```
3. Check that the HTTP Administration Server is registered to listen on port 8008 and is therefore ready to handle requests:

```
# netstat -an | grep LISTEN | grep 8008
```

Check request handling by HTTP Server

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 10-4 on page 291 will be displayed if the IBM HTTP Server has been installed and configured correctly.

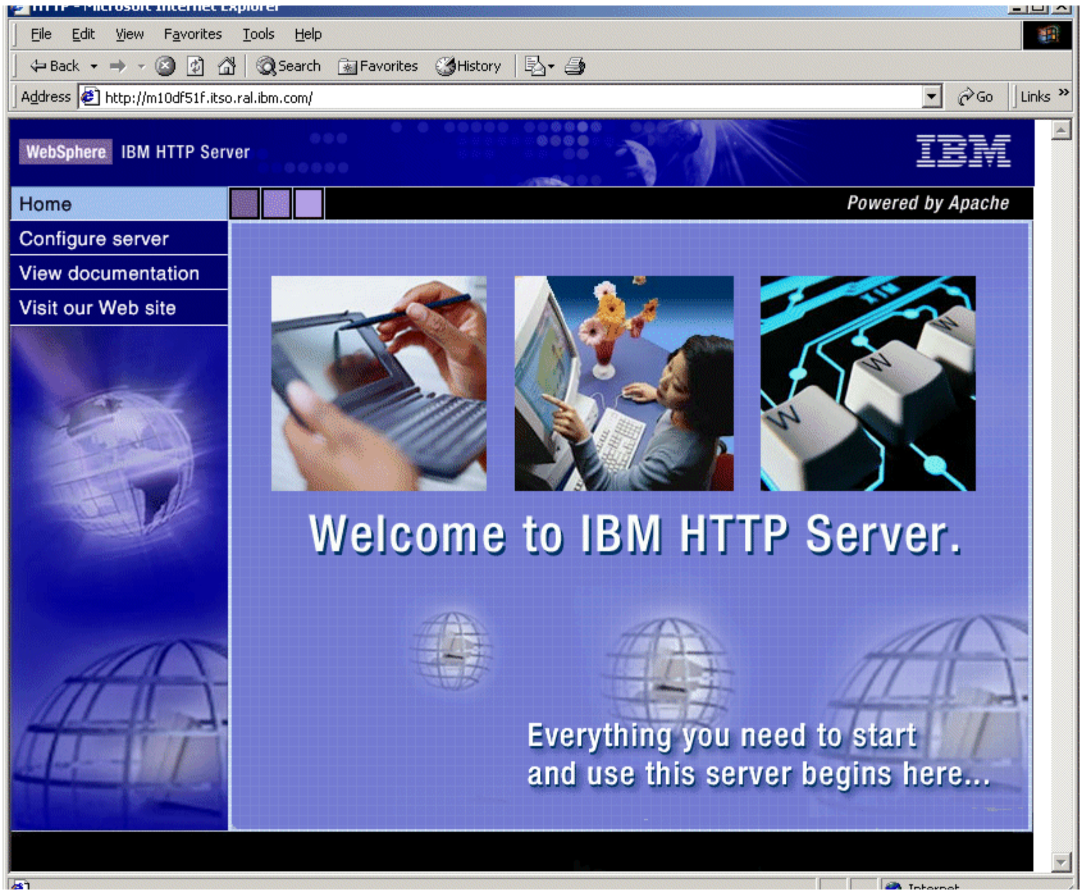

File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com/
Go Links
WebSphere IBM HTTP Server
IBM
Home
Configure server
View documentation
Visit our Web site
Powered by Apache
Welcome to IBM HTTP Server.
Everything you need to start and use this server begins here...
Internet

Figure 10-4 Home page request handled by IBM HTTP Server

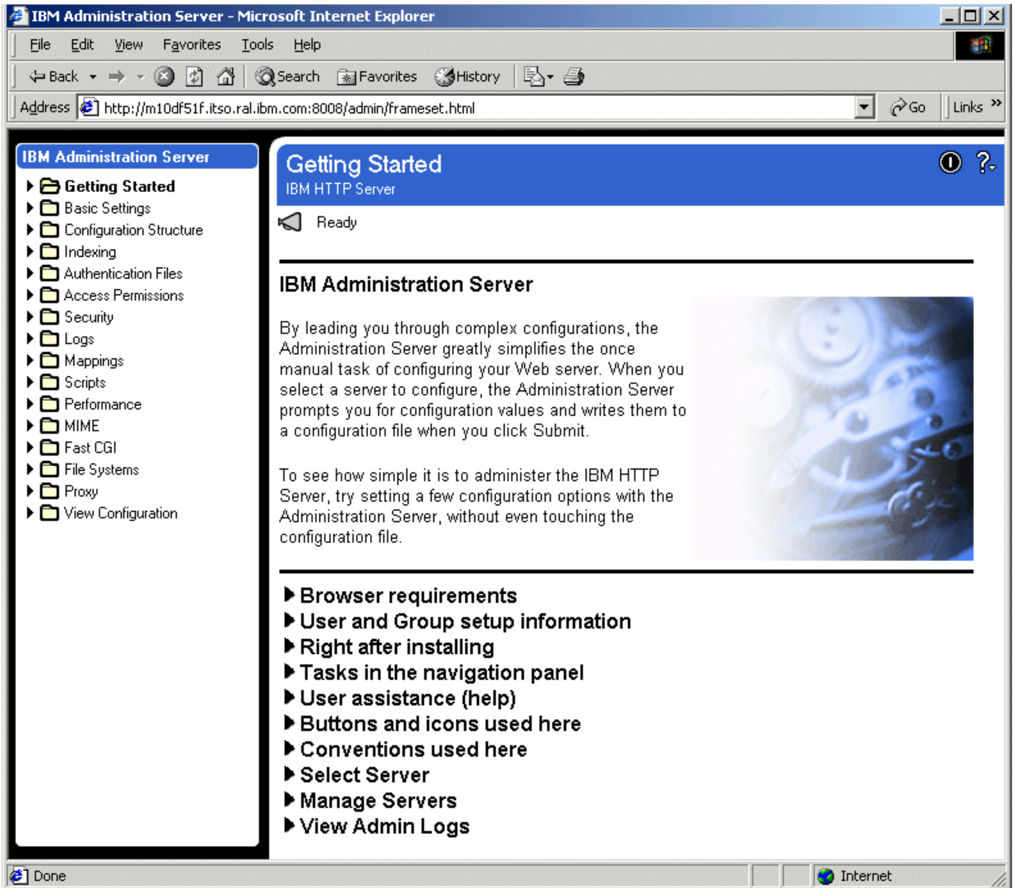
Check request handling by HTTP Administration Server

To check HTTP Administration Server request handling, perform the following steps:

1. Using a Web browser, request the following URL representing the HTTP Administration Server home page:

   `http://<hostname.domain.com>:8008/`

2. In the User Name and Password dialog box, enter user name and password specified in “Create the HTTP server admin account” on page 288. The window shown in Figure 10-5 on page 292 will be displayed if the HTTP Administration Server has been configured correctly.



By leading you through complex configurations, the Administration Server greatly simplifies the once manual task of configuring your Web server. When you select a server to configure, the Administration Server prompts you for configuration values and writes them to a configuration file when you click Submit.

To see how simple it is to administer the IBM HTTP Server, try setting a few configuration options with the Administration Server, without even touching the configuration file.

Figure 10-5 Home page request handled by HTTP Administration Server

10.3.5 Enable SSL encryption for requests (optional)

Author Comment (Carla): If we end up with a security chapter we should point to this section.

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 multi-step 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. A sample configuration file called httpd.conf.sample is shipped with the product and includes the entries required for enabling SSL. You can copy the lines from the sample configuration file into httpd.conf or, if you have not started configuring your server, you can simply use the sample file as your httpd.conf:
   - Change to the <http_server_install_path>/conf directory.
   - Back up the existing httpd.conf file by renaming it to httpd.conf.bak.
   - Rename httpd.conf.sample to httpd.conf.

2. The following configuration file entries are used to enable SSL. 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.

   ```
   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”. The method used is the same for all platforms, except for the following:

1. To run the Web server IBM Key Management Utility, issue the following commands:

   ```
   # gsk5ikm
   ```

Author Comment (security resident): Properly point to the “Create a new keystore (certificate trust database”. ????

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:

   ```
   # gsk5ikm
   ```

Author Comment (security resident): Needs to update “Create a new keystore (certificate trust) database” on page 199

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.

Author Comment (security resident): Needs to update “Verify SSL configuration” on page 202

10.4 Install WebSphere Application Server

This section provides detailed instructions for installing, configuring, and verifying WebSphere Application Server V5.0 for AIX.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Installing WebSphere.
3. Verifying the WebSphere installation.

10.4.1 Preinstallation tasks

Prior to installing WebSphere Application Server, 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:
   - 2809 (bootstrap port)
   - 8879 (SOAP connector address for JMX)
   - 9080 (sample applications)
   - 9443 (sample applications for SSL access)
   - 9090 (adminconsole)
   - 9043 (adminconsole for SSL access)
   - 80 (standard HTTP port) - this is only necessary if you are also installing the IBM HTTP Server on this system.

We suggest using the following command for this task:

```
# netstat -an | grep LISTEN
```

Important: Port 9090 might be already used by the Web-based system manager of AIX V5.1. In this case, the port number for admin console should be changed after the installation is completed. Please refer to “Change the port number of admin console” on page 301 for the detailed steps.

Stop the Web server processes

If you have already installed the IBM HTTP Server process, you should stop it while WebSphere is being 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
```

10.4.2 Install WebSphere Application Server

To install WebSphere Application Server using the GUI installer interface, complete the following steps on the WebSphere server machine:

Tip: The WebSphere installation script (install.sh) also provides a non-GUI scripted or “silent” mode of operation. See “Install WebSphere Application Server - silent mode” on page 334 for details.

1. Log in as root.
2. Start a terminal session.
3. Load the IBM WebSphere Application Server V5.0 CD-ROM into the CD-ROM drive and mount the CD.

```
# mount -r -v cdrfs /dev/cd0 /mnt
```

Author Comment: (Carla) verify the CD name

4. Change the directory to the installation root.
5. Ensure the DISPLAY and TERM environment variables are properly set. They can be set by the following commands:

```
export DISPLAY=<IP address of host>:0.0
export TERM=vt100
```

Author Comment (Arihiro): how? to what?
=> Done

6. Run the LaunchPad.sh installation script:

```
# ./LaunchPad.sh
```

7. In the Select a Language window, select a language and click **OK**. In this example, `English` is selected.
8. In the LaunchPad main window, click **Install the product**.

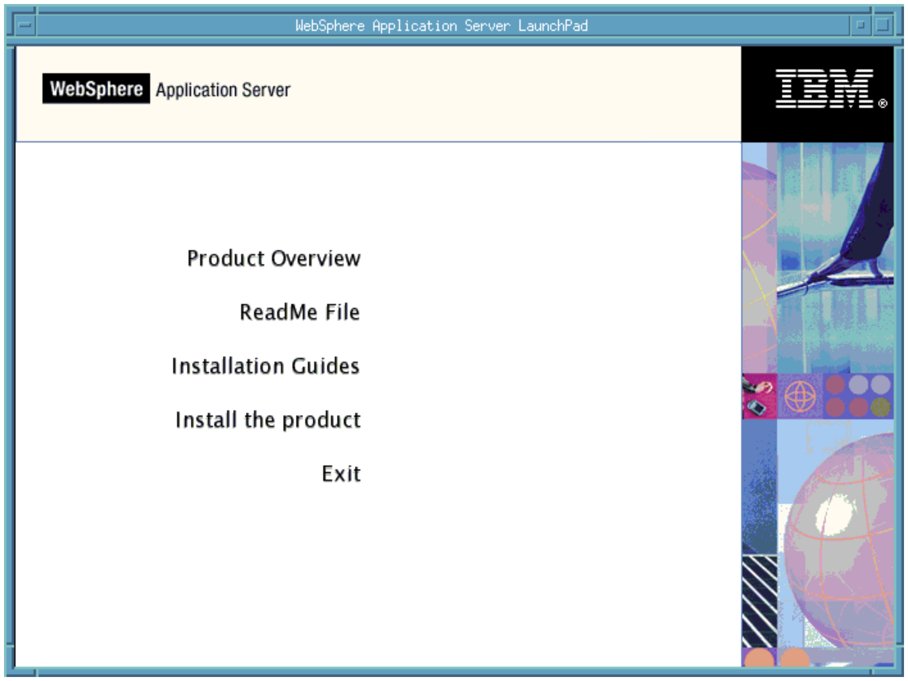


Figure 10-6 LaunchPad main window

9. In the Installation Wizard window, select a language and click **OK**. In this example, `English` is selected.
10. In the Welcome window, click **Next**.
11. In the Software License Agreement window, read the licence agreement, select **Accept**, and click **Next**.
12. In the Detect WebSphere copy window, click **Next**. If neither IBM HTTP Server nor some of WebSphere Components have already been installed, this window will not appear.
13. In the Operating System Level Check window, Click **Next**.
14. IIn the Installation Options window:
    a. If you are installing the IBM HTTP Server with this installation, select Typical and click **Next**.
    b. If you are not (you are using a separate Web server, or one other than the IBM HTTP Server) select **Custom** and click **Next**. The next window will allow you to select the features to install.

Deselect IBM HTTP Server and if you have no other Web servers on this machine, delect the HTTP plugins. If you are planning to use another Web server on this machine with this installation, select the appropriate plug-in.

Note: Although not listed in the window, the IBM JDK 1.3.1 is automatically installed under the WebSphere installation directory. There is no need to separately install a JDK for use by WebSphere Application Server or the Web server plug-ins.

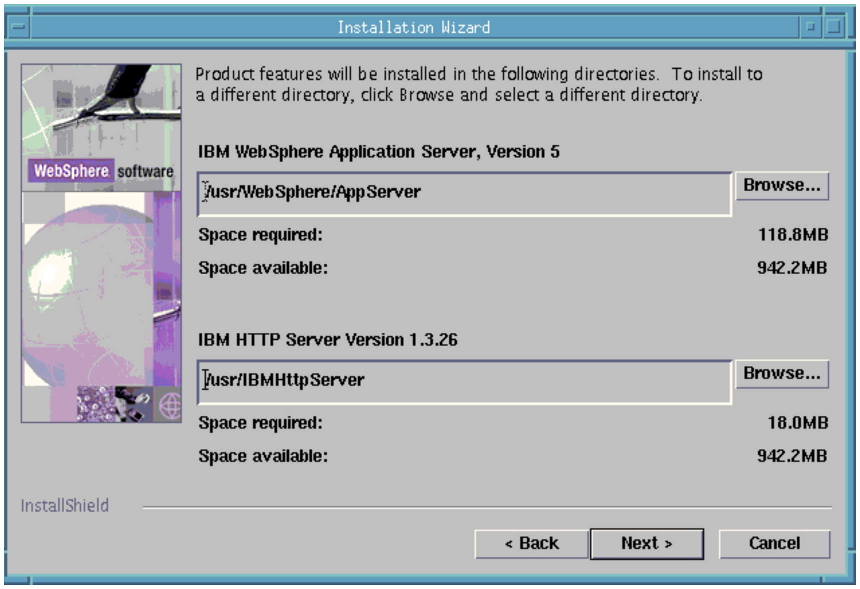
Click **Next**.

Note: The embedded messaging option will automatically create a user called `mqm`.

15. In the Install directory window, enter the directories for WebSphere and IBM HTTP Server. In this example, the default installation directories listed in Table 10-5 are used. Click **Next**:

Table 10-5 default installation directory

component	**installation directory**
WebSphere Application Server	/usr/WebSphere/AppServer
IBM HTTP Server	/usr/IBMHttpServer



Product features will be installed in the following directories. To install to a different directory, click Browse and select a different directory.

IBM WebSphere Application Server, Version 5

/usr/WebSphere/AppServer

Space required: 118.8MB

Space available: 942.2MB

IBM HTTP Server Version 1.3.26

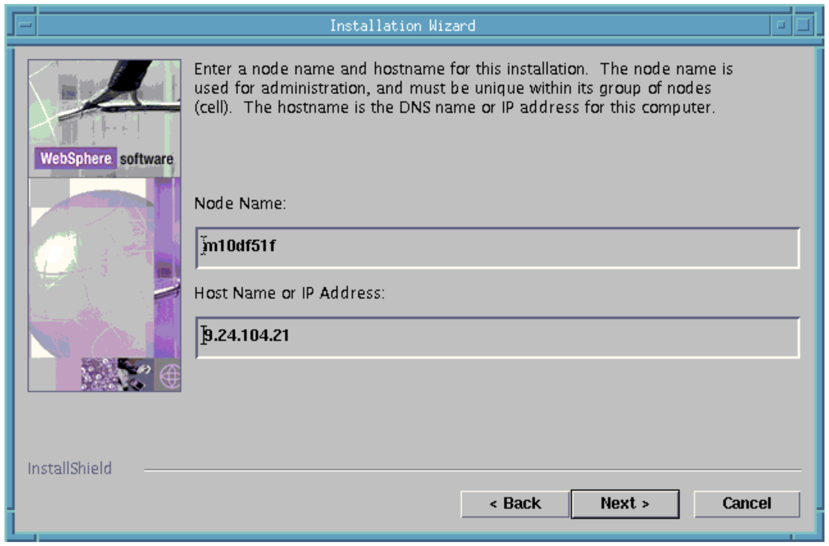
/usr/IBMHttpServer

Space required: 18.0MB

Space available: 942.2MB

Figure 10-7 Install directory window

16. In the node name and hostname window, enter a node name and host name or IP Address. The values will automatically default to the host name and IP address for your system. Click **Next**.



Enter a node name and hostname for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer.

Node Name:

m10df51f

Host Name or IP Address:

9.24.104.21

Figure 10-8 Node name and hostname window

17. In the confirmation window, confirm the selected features will be installed, and click **Next**.
18. If you would like to register the product at this time, proceed through the registration panels. If not, deselect **Register** and click **Next**. Registration can be done later using the First Steps window.
19. In the Finish window, click **Finish**. The installation of the WebSphere Application Server is now complete. The installation program automatically starts First Steps Program.

Change the port number of admin console

Port 9090 might be already used by the Web-based system manager of AIX V5.1. In this case, the port number for the admin console can be changed. The following files have to be changed:

Table 10-6 Files to be changed

Setting	**File**
Virtual Host	<WAS_BASE_HOME>/config/cells/<cellname>/virtualhosts.xml
HTTP Transport	<WAS_BASE_HOME>/config/cells/<cellname>/nodes/<nodename>/servers/server1/server.xml
`<cellname>` is the host name of the machine and `<nodename>` is a value which is specified during the installation.	

To change the virtual host setting, modify the following lines in the virtualhosts.xml file. In this example, the port number is changed from 9090 to 9091:

```
<host:VirtualHost xmi:id="VirtualHost_2" name="admin_host">
........
   <aliases xmi:id="HostAlias_4" hostname="*" port="9091"/>
   <aliases xmi:id="HostAlias_5" hostname="*" port="9043"/>
</host:VirtualHost>
```

To change the HTTP transport setting, modify the following part in the server.xml file. In this example, the port number is changed from 9090 to 9091:

```
<transports xmi:type="applicationserver.webcontainer:HTTPTransport"
xmi:id="HTTPTransport_3" sslEnabled="false">
   <address xmi:id="EndPoint_3" host="" port="9091"/>
</transports>
```

Note: If the port number of the admin console is changed, starting the admin console from First Steps Program will fail to connect to admin console because it uses default port, which is 9090.

10.4.3 Verify the WebSphere installation

Note: If you installed the IBM HTTP Server, see 10.3.3, “Configure the IBM HTTP Server” on page 288 for information on setting up and installation verification.

In order to very the installation of WebSphere Application Server, the following tasks must be completed in order:

1. Use the Installation Verification Test.
2. Check the installation log.
3. Check the Web server configuration file changes
4. Start the application server.
5. Start the Web server.

Use the Installation Verification Test

The Installation Verification Test is a tool which scans the product log files for errors and verifies core functionality of the product installation. This tool performs the followings:

1. Starts the application server (server1) if it has not started yet and verify its status.
2. Verifies the servlet engine status
3. Verifies the JSP status
4. Verifies the EJB status

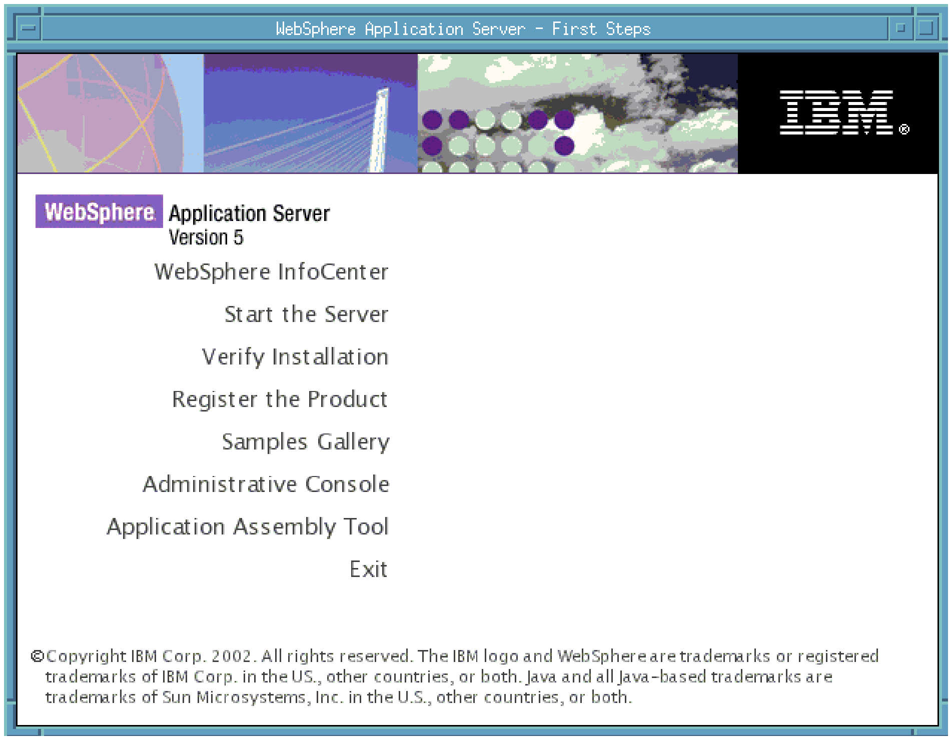
The installation program automatically starts the First Steps program. It can be started by issuing the following commands:

```
# cd <WAS_BASE_HOME>/bin
# ./firststeps.sh
```

Author Comment (Arihiro): how can you start first steps after the installation? for example, if you close the window?
=> Done

To start the Installation Verification Test from the First Step program:

1. Click **Verify Installation**.
2. The Installation Verification Test starts the verification. After the verification is completed, press **Enter**.



©Copyright IBM Corp. 2002. All rights reserved. The IBM logo and WebSphere are trademarks or registered trademarks of IBM Corp. in the US., other countries, or both. Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the U.S., other countries, or both.

Figure 10-9 First Steps Main Window

The Installation Verification Test can also be started by the following steps:

1. Log in as root on the WebSphere machine.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <WAS_BASE_HOME>/bin
# ./ivt.sh
```

Check the log file, <WAS_BASE_HOME>/logs/ivt.log, for the result.

Check the installation log

If you find that the IVT does not complete correctly, you can check the following log files under <WAS_BASE_HOME>/logs directory for installation errors.

Author Comment (Carla): move the http server to another table .. or to the end? it isn't under was_base_home

Table 10-7 Installation log files

Component	File
WebSphere Application Server	log.txt
IBM HTTP Server	ihs_log.txt
Default Application	installDefaultApplication.log
Sample Application	installSamples.log
Admin Console	installAdminConsole.log
MDB Samples Application	installMessagingSamples.log
Pet Store Application	installPetStore.log
Plants By WebSphere Application	installPlantsByWeb.log
Technology Samples Application	installTechSamples.log
IVT Application	installIVTApp.log

The installation wizard also creates the optional_log.txt file in the $TEMP directory. Check this file if necessary.

Check the Web server configuration file changes

Note: If you did not install the Web server and WebSphere plug-in on the same system as WebSphere, skip this step. “Install the WebSphere plug-in on a remote Web server” on page 309 discusses installing and generating the WebSphere plug-in on a remote machine.

To check the Web server configuration file changes, complete the following steps:

Author Comment (Arihiro): what if the http server is not on the same machine?
=> Done

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:

```
Alias /WSsamples /usr/WebSphere/AppServer/WSsamples
Alias /IBMWebAS/ /usr/WebSphere/AppServer/web/
LoadModule ibm_app_server_http_module
/usr/WebSphere/AppServer/bin/mod_ibm_app_server_http.so
WebSpherePluginConfig /usr/WebSphere/AppServer/config/plugin-cfg.xml
```

If not, manually add the above lines to the end of the httpd.conf file and save the changes.

Start the application server

The Installation Verification Test starts the application server but for future reference, the server can be started by either of the following ways:

- Start from First Steps window by clicking **Start the Server**
- Or, from a terminal session
  a. Log in as root on the WebSphere machine.
  b. Start a terminal session.
  c. Issue the following commands:

```
# cd <WAS_BASE_HOME>/bin
# ./startServer.sh server1
```

Author Comment (Arihiro): can we still use tail -f ../logfile to see messages?
=> Done

Check the log files, SystemOut.log and SystemErr.log, under <WAS_BASE_HOME>/logs/server1 directory. Issue the following commands in order to check the SystemOut.log.

```
# cd <WAS_BASE_HOME>/logs
# tail -f SystemOut.log
```

The following message should be appeared in SystemOut.log for its successful start.

```
[9/12/02 17:14:40:617 EDT] 425dc76b WsServer      A WSVR0001I: Server
server1 open for e-business
```

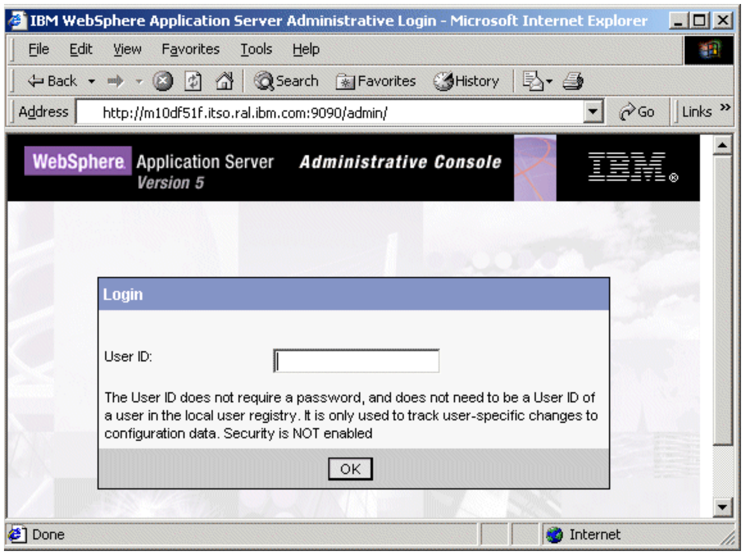
To access the admin console:

1. Using a Web browser, request the following URL:

   `http://<hostname>:9090/admin`

   A window similar to the one shown in Figure 10-10 on page 306 is displayed in the browser.

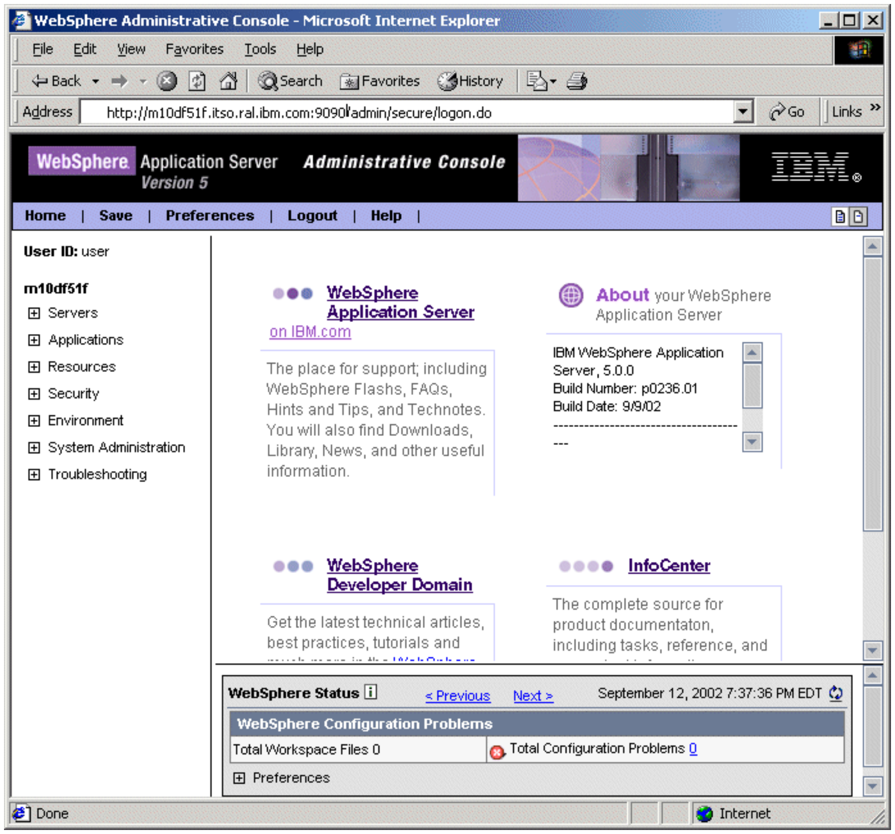
Note: If the port number of the admin console is changed, the correct port number should be specified.


IBM WebSphere Application Server Administrative Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9090/admin/
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Login
User ID:
OK
Done
Internet

The User ID does not require a password, and does not need to be a User ID of a user in the local user registry. It is only used to track user-specific changes to configuration data. Security is NOT enabled

Figure 10-10 Admin console

2. Enter User ID to login. It is for tracking purpose only, so any user ID is acceptable. In this example, **user** is used as User ID. The following window will appear in the browser.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
User ID: user
m10df51f
Servers
Applications
Resources
Security
Environment
System Administration
Troubleshooting
WebSphere Application Server on IBM.com
About your WebSphere Application Server
WebSphere Developer Domain
InfoCenter
WebSphere Status < Previous Next >
Preferences
Done
Internet

The place for support; including WebSphere Flashs, FAQs, Hints and Tips, and Technotes. You will also find Downloads, Library, News, and other useful information.

IBM WebSphere Application Server, 5.0.0
Build Number: p0236.01
Build Date: 9/9/02

Get the latest technical articles, best practices, tutorials and

The complete source for product documentaton, including tasks, reference, and

September 12, 2002 7:37:36 PM EDT

WebSphere Configuration Problems	
Total Workspace Files 0	Total Configuration Problems 0

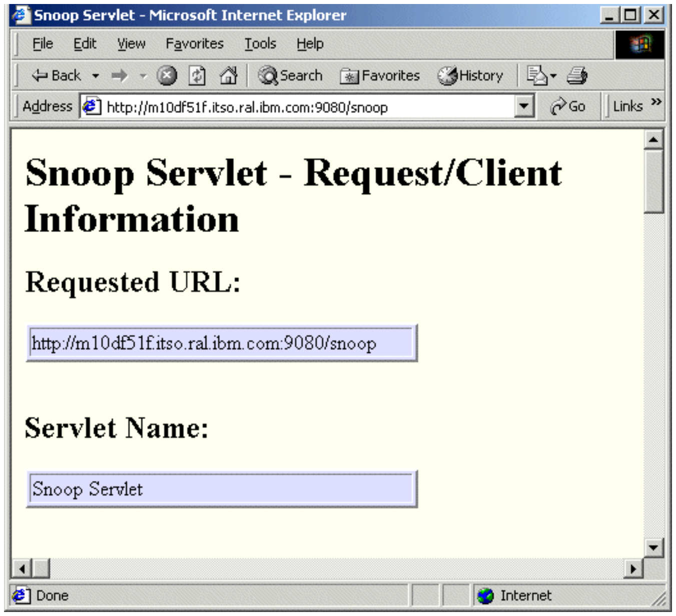
Figure 10-11 Admin console first page

To perform one final test, access the snoop application:

1. Using a Web browser, request the following URL:

```
http://<hostname>:9080/snoop
```

   A window similar to the one shown in Figure 10-13 is displayed in the browser.


Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9080/snoop
Go Links
Done
Internet

Snoop Servlet - Request/Client Information

Requested URL:

http://m10df51f.itso.ral.ibm.com:9080/snoop

Servlet Name:

Snoop Servlet

Figure 10-12 Snoop servlet

Start the Web server

Note: If you installed the IBM HTTP Server on this system, you can proceed with this section. If not,

To verify Web server plug-in configuration, the Web server must be started.

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands to start the IBM HTTP Server:

```
# cd <http_server_install_path>/bin
# ./apachectl start
```

Note: If you are using another local Web server, refer to the product instructions for starting it.

The Web server plug-in configuration can be verified by requesting a servlet through the Web server that has already successfully requested through the Web container's embedded Web server:

Using a Web browser, request the following URL:

```
http://<hostname>/snoop
```

A window similar to the one shown in Figure 10-12 on page 308 is displayed in the browser.

10.5 Install the WebSphere plug-in on a remote Web server

Author Comment: It seems like this whole section should be integrated into 10.3, “Install the Web server and plugin” on page 284. If you are installing the IBM HTTP Server, why not install it and the plug-ins at the same time using the launchpad?

Note: If you have installed the IBM HTTP Server or another supported Web server on a machine separate from WebSphere Application Server, proceed into this section. These instructions assume the IBM HTTP Server is used, but the process would be similar for other Web servers.

This section provides detailed instructions for installing, configuring, and verifying the WebSphere HTTP plug-in on a remote Web server. It is assumed that IBM HTTP Server has already installed.

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.

10.5.1 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
```

10.5.2 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 V5.0 CD-ROM into the CD-ROM drive and mount the CD.

```
# mount -r -v cdrfs /dev/cd0 /mnt
```

Author Comment (Carla): check the CDROM name

4. Change the directory to the installation root.
5. Ensure the DISPLAY and TERM environment variables are properly set. They can be set by the following commands:

```
export DISPLAY=<IP address of host>:0.0
export TERM=vt100
```

6. Run the LaunchPad.sh installation script:

```
# ./LaunchPad.sh
```

7. In the Select a Language window, select a language and click **OK**. In this example, `English` is selected.
8. In the LaunchPad main window, click **Install the product**.

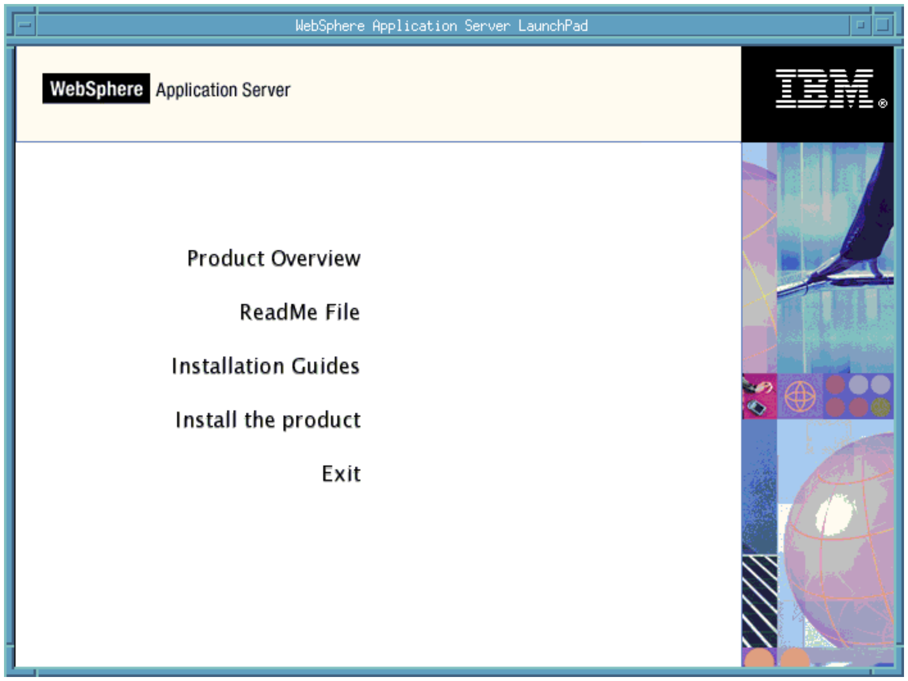
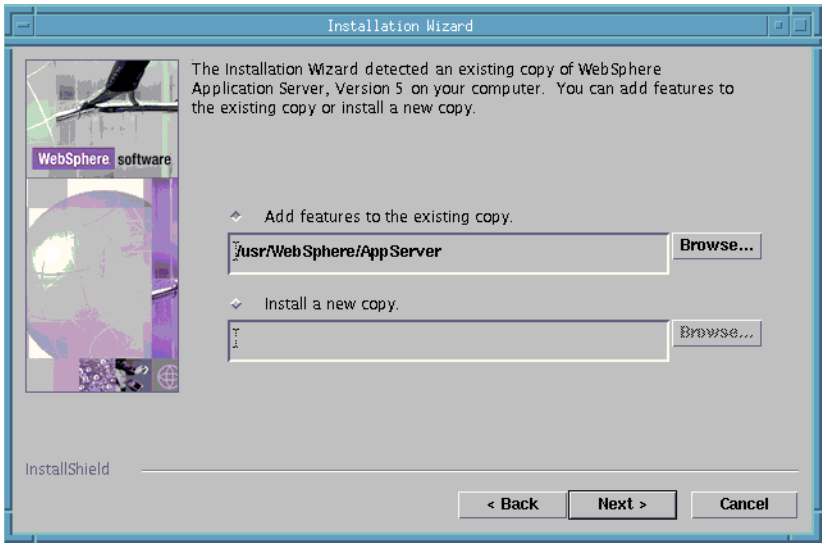


Figure 10-13 LaunchPad main window (Plugin)

9. In the Installation Wizard window, select a language and click **OK**. In this example, `English` is selected.
10. In the Welcome window, click **Next**.
11. In the Detect WebSphere copy window, click **Next**.


Installation Wizard

The Installation Wizard detected an existing copy of WebSphere Application Server, Version 5 on your computer. You can add features to the existing copy or install a new copy.

Add features to the existing copy.

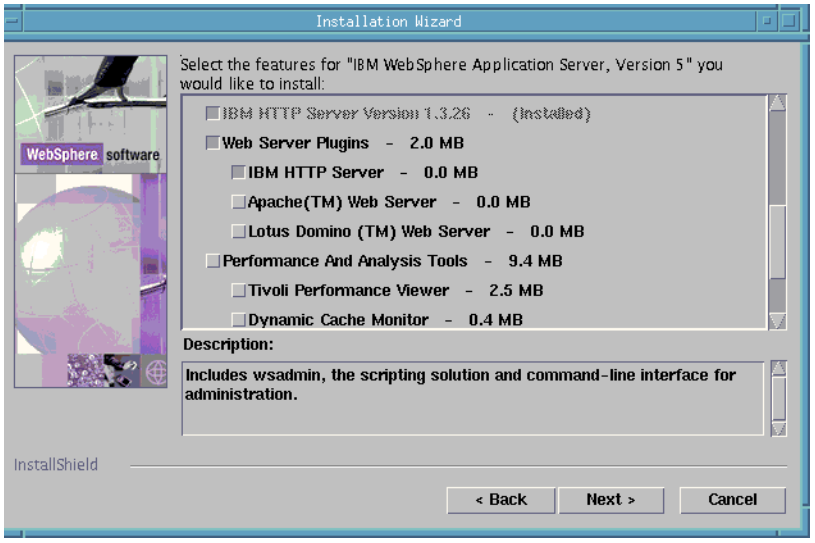
/usr/WebSphere/AppServer

Install a new copy.

Figure 10-14 Detect WebSphere Copy (plugin)

Author Comment (??): In this chapter, IBM HTTP Server is installed by install.sh instead of installIHS.sh because the shell script is not available now. If the behavior of both shell scripts are different, some modifications are required.

12. In the Software License Agreement window, read the license agreement, select **Accept**, and click **Next**.
13. In the Operating System Level Check window, Click **Next**.
14. In the Installation Options window, select Custom and click **Next**.
15. In the Select the features window, choose Web server plugin of IBM HTTP Server, click **Next**.



Select the features for "IBM WebSphere Application Server, Version 5" you would like to install:

- IBM HTTP Server Version 1.3.26 - (Installed)
- Web Server Plugins - 2.0 MB
  - IBM HTTP Server - 0.0 MB
  - Apache(TM) Web Server - 0.0 MB
  - Lotus Domino (TM) Web Server - 0.0 MB
- Performance And Analysis Tools - 9.4 MB
  - Tivoli Performance Viewer - 2.5 MB
  - Dynamic Cache Monitor - 0.4 MB

Description:

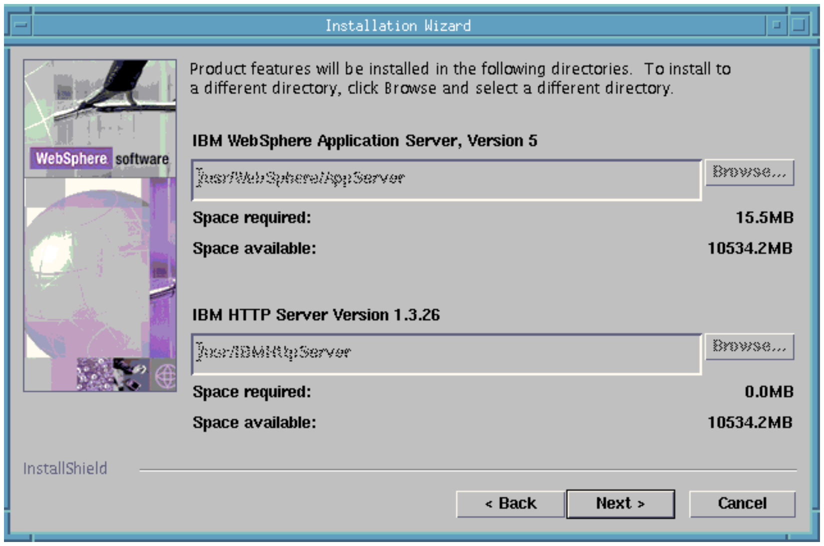
Includes wsadmin, the scripting solution and command-line interface for administration.

Figure 10-15 Select the features window (plugin)

Attention: Since IBM HTTP Server has already installed, it is shaded and can not be selected.

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.

16. In the Install directory window, the installation directories are shaded and are not able to be entered because some components of WebSphere and IBM HTTP Server has been already installed. Click **Next**:


Installation Wizard

Product features will be installed in the following directories. To install to a different directory, click Browse and select a different directory.

IBM WebSphere Application Server, Version 5

/usr/WebSphere/AppServer

Space required: 15.5MB

Space available: 10534.2MB

IBM HTTP Server Version 1.3.26

/usr/IBMHttpServer

Space required: 0.0MB

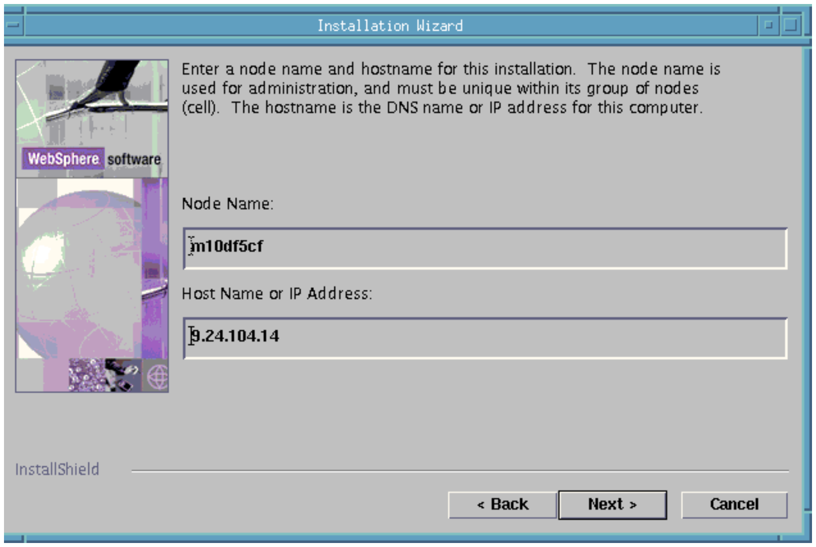
Space available: 10534.2MB

Figure 10-16 Install directory window (plugin)

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 1.3.1.

17. In the node name and hostname window, enter a node name, and host name or IP Address. Click **Next**. In this example, the default values are used.



Enter a node name and hostname for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer.

Node Name: m10df5cf

Host Name or IP Address: 9.24.104.14

Figure 10-17 Node name and hostname window (plugin)

18. In the confirmation window, confirm the selected features will be installed, and click **Next**.
19. If you would like to register the product at this time, proceed through the registration panels. If not, deselect **Register** and click **Next**.
20. In the Finish window, click **Finish**. The installation of the plug-in is now complete.
21. Click **Exit** in the LaunchPad window.

10.5.3 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 315.

10.5.4 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. Regenerate the Web server plug-in settings.
2. Copy the plug-in settings to the remote server.
3. Restart the Web server.

Author Comment (Carla): In this section, “Configure the WebSphere virtual host” as written in V4.0 Red Book is not included because it works correctly without this configuration. If this information is necessary, same steps in the V4.0 Red Book should be included here.

Regenerate the Web server plug-in settings

To regenerate the Web server plug-in settings:

1. Follow the steps described in Chapter 16, “Configuring the Web server interface” on page 549.

Author Comment (Carla): Need a more specific reference here or copy the steps inline. The next section is probaby also a repeat.

Copy the plug-in settings to the remote Web server

Next the regenerated Web server plug-in settings must be copied to the remote Web server:

1. Copy the <WAS_BASE_HOME>/config/plugin-cfg.xml file from the WebSphere server machine to the <WAS_BASE_HOME>/config directory on the remote Web server machine.

Author Comment (Arihiro): do we copy both if ND or just the second? .. also, we haven’t installed ND yet so this may be out of place.
=> Done. Description about ND is not necessary here.

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
```

10.5.5 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="Error" Name="<plugin_install_path>/logs/http_plugin.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:

```
[Mon Sep 16 14:40:55 2002] 000040de 00000001 - PLUGIN: Plugins loaded.
[Mon Sep 16 14:40:55 2002] 000040de 00000001 - PLUGIN:
--------------------System Information-----------------------
[Mon Sep 16 14:40:55 2002] 000040de 00000001 - PLUGIN: Bld date: Sep  8
2002, 23:56:52
[Mon Sep 16 14:40:55 2002] 000040de 00000001 - PLUGIN: Webserver:
IBM_HTTP_SERVER/1.3.26  Apache/1.3.26 (Unix)
[Mon Sep 16 14:40:55 2002] 000040de 00000001 - PLUGIN: Hostname = m10df5cf
[Mon Sep 16 14:40:55 2002] 000040de 00000001 - PLUGIN: NOFILES = hard:
INFINITE, soft: 2000
```

```
[Mon Sep 16 14:40:55 2002] 000040de 00000001 - PLUGIN: MAX COREFILE SZ =
hard: INFINITE, soft: 1073741312
[Mon Sep 16 14:40:55 2002] 000040de 00000001 - PLUGIN: DATA = hard:
INFINITE, soft: 134217728
[Mon Sep 16 14:40:55 2002] 000040de 00000001 - PLUGIN:
--------------------------------------------------------------
```

Note: See Appendix C, “The plugin-cfg.xml file definitions” on page 1071 for more details.

Author Comment (Carla): The cross-reference to the plugin-cfg.xml should be updated later. Where is this content going to be??

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>/snoop
   ```

If both the Web server and WebSphere have been correctly configured to support remote HTTP plug-in access, then a window similar to Figure 10-12 on page 308 will be obtained.

10.6 Install WebSphere Application Server Network Deployment

This section provides detailed instructions for installing, configuring, and verifying WebSphere Application Server Network Deployment for AIX.

The section is organized into the following tasks:

1. Preinstallation tasks.
2. Install WebSphere Network Deployment.
3. Verify the WebSphere Network Deployment installation.

10.6.1 Preinstallation tasks (check IP port usage)

Prior to installing WebSphere Application Server Network Deployment, you will need to check that the following IP ports are unused:

- 2809 (bootstrap port)
- 8879 (SOAP connector address for JMX)
- 9090 (adminconsole)
- 9043 (adminconsole for SSL access)

We suggest using the following command for this task:

```
# netstat -an | grep LISTEN
```

Important: Port 9090 might be already used by the Web-based system manager of AIX V5.1. In this case, the port number for admin console should be changed after the installation is completed. Please refer to “Change the port number of admin console” on page 301 for the detailed steps.

10.6.2 Install the Network Deployment

To install WebSphere Application Server Network Deployment using the GUI installer interface, complete the following steps on the WebSphere server machine:

Tip: The WebSphere installation script (install.sh) also provides a non-GUI scripted or “silent” mode of operation. See “Install Network Deployment - silent mode” on page 338 for details.

1. Log in as root.
2. Start a terminal session.
3. Load the IBM WebSphere Application Server V5.0 Network Deployment CD-ROM into the CD-ROM drive and mount the CD.

```
# mount -r -v cdrfs /dev/cd0 /mnt
```

Author Comment (Carla): verify CDROM name

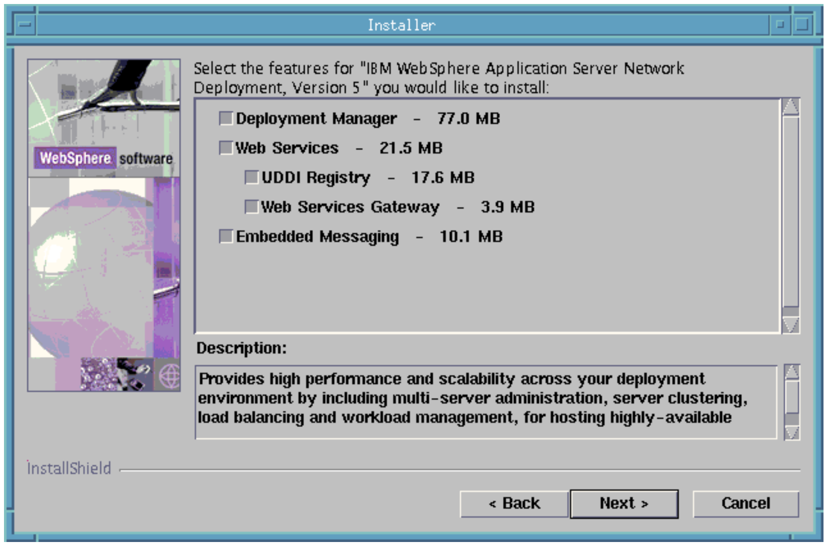
4. Change the directory to the installation root.
5. Ensure the DISPLAY and TERM environment variables are properly set. They can be set by the following commands:

```
export DISPLAY=<IP address of host>:0.0
export TERM=vt100
```

6. Run the LaunchPad.sh installation script:

```
# ./LaunchPad.sh
```

7. In the Select a Language window, select a language and click **OK**. In this example, `English` is selected.
8. In the LaunchPad main window, click **Install the product**.
9. In the Installation Wizard window, select a language, English for example, and click **OK**.
10. In the Welcome window, click **Next**.
11. In the Software License Agreement window, read the license agreement, select **Accept**, and click **Next**.
12. In the Operating System Level Check window, Click **Next**.
13. In the Installation Options window, select **Custom** and click **Next**.
14. In the Select the features window, choose all options, and click **Next**.



Select the features for "IBM WebSphere Application Server Network Deployment, Version 5" you would like to install:

- Deployment Manager - 77.0 MB
- Web Services - 21.5 MB
  - UDDI Registry - 17.6 MB
  - Web Services Gateway - 3.9 MB
- Embedded Messaging - 10.1 MB

Description:

Provides high performance and scalability across your deployment environment by including multi-server administration, server clustering, load balancing and workload management, for hosting highly-available

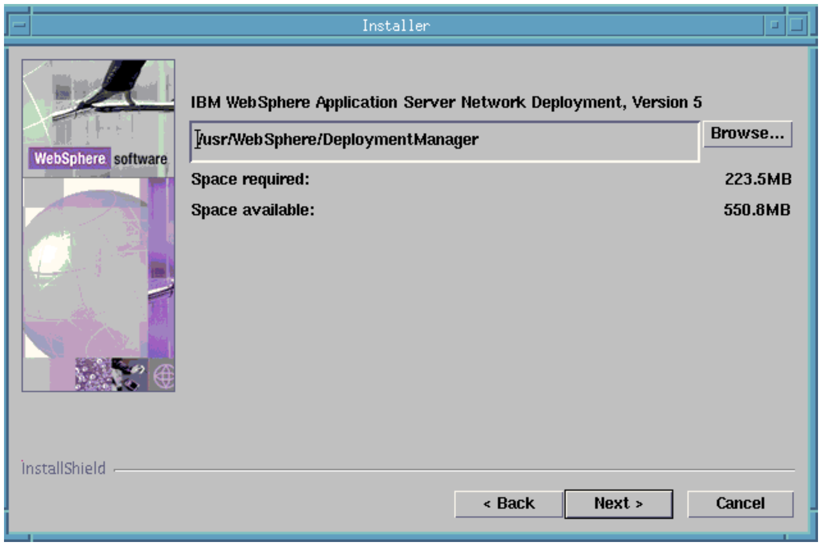
Figure 10-18 Select the features window (ND)

Important: Although not listed in the window, the IBM JDK 1.3.1 is automatically installed under the WebSphere Network Deployment installation directory. There is no need to separately install a JDK for used by WebSphere Application Server Network Deployment.

15. In the Install directory window, enter the directories for WebSphere Network Deployment. In this example, the default installation directories listed in Table 10-8 are used. Click **Next**:

Table 10-8 default installation directory (ND)

component	installation directory
WebSphere Application Server Network Deployment	/usr/WebSphere/DeploymentManager



IBM WebSphere Application Server Network Deployment, Version 5

/usr/WebSphere/DeploymentManager

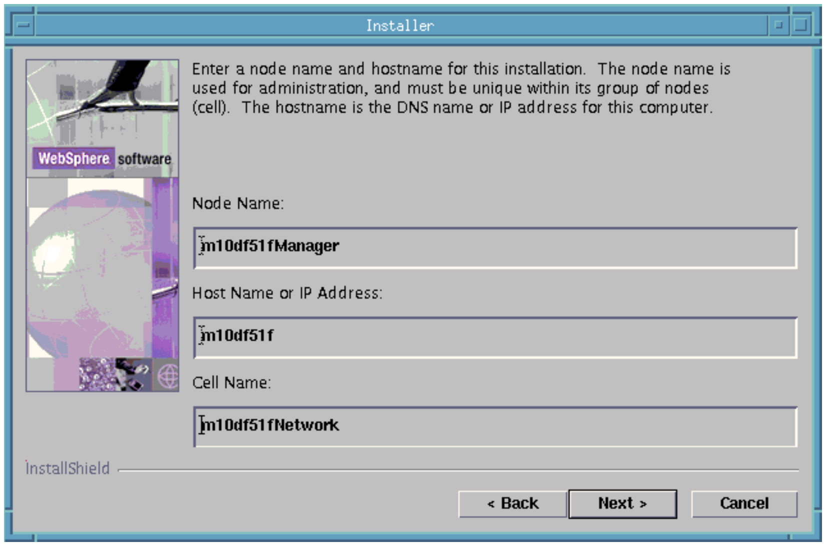
Space required: 223.5MB

Space available: 550.8MB

Figure 10-19 Install directory window (ND)

16. In the node name, hostname, and cell name window, the default values will probably be sufficient, but you have the option of entering new values. Click **Next**.

Author Comment (Noelle): Noelle changed the node name and had a problem .. do we need a note here?


Enter a node name and hostname for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer.

Node Name: m10df51fManager

Host Name or IP Address: m10df51f

Cell Name: m10df51fNetwork

Figure 10-20 Node name, hostname and cell name window (ND)

17. In the confirmation window, confirm the selected features will be installed, and click **Next**.
18. If you would like to register the product at this time, proceed through the registration panels. If not, deselect **Register** and click **Next**.

In the Finish window, click **Finish**. The installation of the WebSphere Application Server Network Deployment is now complete. The installation program automatically starts First Steps Program. It can be started by issuing the following commands:

```
# cd <WAS_ND_HOME>/bin
# ./firststeps.sh
```

Change the port number of admin console

Port 9090 might be already used by the Web-based system manager of AIX V5.1. In this case, the port number for admin console should be changed. The following files have to be changed:

Table 10-9 Files to be changed

Setting	File
Virtual Host	<WAS_ND_HOME>/config/cells/<cellname>/virtualhosts.xml

Setting	File
HTTP Transport	<WAS_ND_HOME>/config/cells/<cellname>/nodes/<nodename>/servers/dmgr/server.xml

Where both <cellname> and <nodename> are values which are specified during the installation.

To change the virtual host setting, modify the following part in the virtualhosts.xml file. In this example, the port number is changed from 9090 to 9092.:

```
<host:VirtualHost xmi:id="VirtualHost_2" name="admin_host">
   <aliases xmi:id="HostAlias_4" hostname="*" port="9092"/>
   <aliases xmi:id="HostAlias_5" hostname="*" port="9043"/>
</host:VirtualHost>
```

To change the HTTP transport setting, modify the following part in the server.xml file. In this example, the port number is changed from 9090 to 9092.:

```
<transports xmi:type="applicationserver.webcontainer:HTTPTransport"
xmi:id="HTTPTransport_1" sslEnabled="false">
   <address xmi:id="EndPoint_1" host="" port="9092"/>
</transports>
```

Note: If the port number of the admin console is changed, starting admin console from First Steps Program will fail to connect to admin console because it uses default port, which is 9090.

10.6.3 Verify the Network Deployment installation

In order to ensure the installation of WebSphere Application Server Network Deployment was successful, perform the following tasks:

1. Use the Installation Verification Test.
2. Check the installation log.
3. Start the deployment manager.

Use the Installation Verification Test

The Installation Verification Test is a tool which scans the product log files for errors and verifies core functionality of the product installation. This tool starts the deployment manager and verifies its status. The installation program automatically starts the First Steps program.

To start the Installation Verification Test from the First Step program:

1. Click **Verify Installation**.

2. The Installation Verification Test starts the verification. After the verification is completed, press **Enter**.

Alternately, the Installation Verification Test can be started without the First Steps windows by doing the followings:

1. Log in as root on the WebSphere machine.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <WAS_ND_HOME>/bin
# ./ivt.sh
```

Check the log file, <WAS_ND_HOME>/logs/ivt.log, for the result.

Check the installation log

If the IVT is not successful, check the following installation log files under <WAS_ND_HOME>/logs directory for errors.

Table 10-10 Installation log files (ND)

Component	**File**
Network Deployment	log.txt
Admin Console	installAdminConsole.log
File Transfer Application	installFiletransfer.log

Start the deployment manager

The Installation Verification Test starts the deployment manager. But if for some reason, the deployment manager is not running, start it using one of the following ways:

- Start from First Steps window and click **Start the Deployment Manager**
- Or, Start from a terminal session
  a. Log in as root on the WebSphere machine.
  b. Start a terminal session.
  c. Issue the following commands:

```
# cd <WAS_ND_HOME>/bin
# ./startManager.sh
```

Check the log files, SystemOut.log and SystemErr.log, under <WAS_ND_HOME>/logs/dmgr directory. Issue the following commands in order to check the SystemOut.log.

```
# cd <WAS_ND_HOME>/logs
# tail -f SystemOut.log
```

The following message should be appeared in SystemOut.log for its successful start.

Author Comment (Arihiro): not tail -f to a log file??
=> Done

```
[9/13/02 14:06:18:513 EDT] 75c55974 WsServer      A WSVR0001I: Server dmgr
open for e-business
```

To access the admin console:

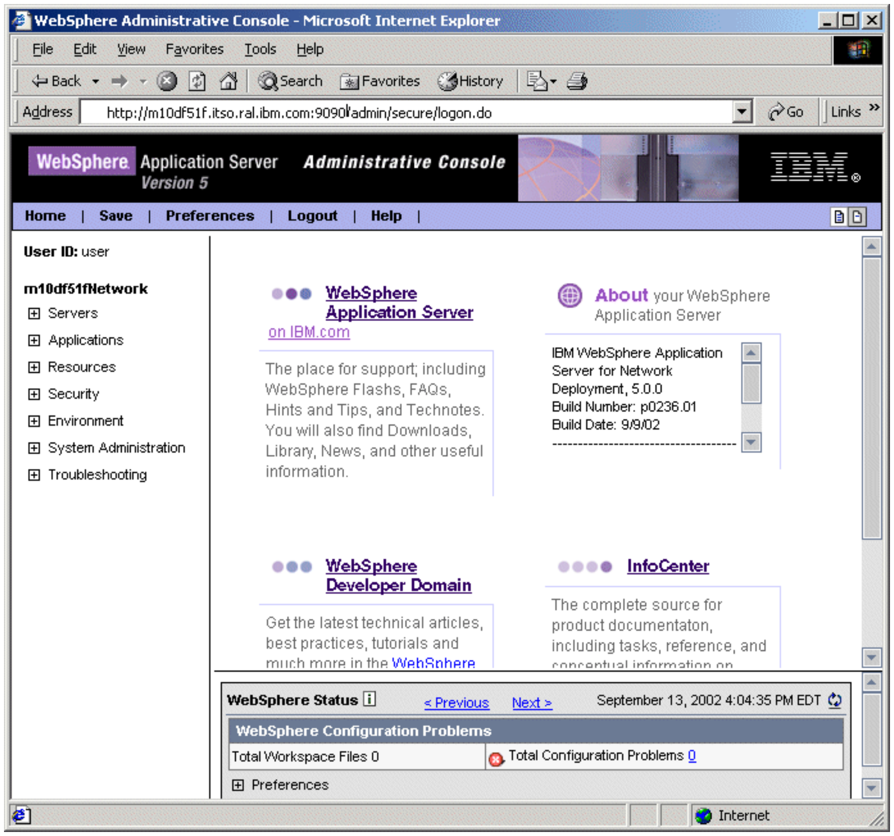
1. Using a Web browser, request the following URL:

   `http://<hostname>:9090/admin`

   A window similar to the one shown in Figure 10-10 on page 306 is displayed in the browser.

Note: If the port number of the admin console is changed, the correct port number should be specified.

2. Enter User ID to login. It is for tracking purpose only, so any user id is acceptable. In this example, **user** is used as User ID. The following window will appear in the browser.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Servers
Applications
Resources
Security
Environment
System Administration
Troubleshooting
Preferences
Internet

User ID: user

m10df51fNetwork

WebSphere Application Server on IBM.com

The place for support; including WebSphere Flashs, FAQs, Hints and Tips, and Technotes. You will also find Downloads, Library, News, and other useful information.

About your WebSphere Application Server

IBM WebSphere Application Server for Network Deployment, 5.0.0
Build Number: p0236.01
Build Date: 9/9/02

WebSphere Developer Domain

Get the latest technical articles, best practices, tutorials and much more in the WebSphere

InfoCenter

The complete source for product documentaton, including tasks, reference, and conceptual information on

WebSphere Status < Previous Next > September 13, 2002 4:04:35 PM EDT

WebSphere Configuration Problems

Total Workspace Files 0 | Total Configuration Problems 0

Figure 10-21 Network Deployment admin console

10.7 Add a WebSphere node into an existing cell

This section provides detailed instructions for configuring and verifying a WebSphere node into an existing WebSphere V5.0 cell.

This section is organized into the following tasks:

1. Preparation tasks
2. Configure the cell
3. Verify the cell

A cell is an aggregation of nodes. A cell is created during the WebSphere Network Deployment installation.

A node is a set of managed servers on a physical machine. A node is created during the WebSphere Application Server installation.

A deployment manager controls and communicates with all node agents that are part of the cell.

See “System Management” on page 405 for the basic concept of the System Management and “WebSphere administration basics” on page 449 for the detail of the administration.

10.7.1 Preparation tasks

Prior to configuring the cell, the installation of both WebSphere Application Server and WebSphere Application Server Network Deployment should be completed.

10.7.2 Configure the cell

There are two steps to configure the cell as shown in the Table 10-11.

Table 10-11 Steps to configure the cell

	WebSphere Application Server machine	**WebSphere Application Server Network Deployment machine**
1		Start the deployment manager
2	Add a node into an existing cell	

Start the deployment manager

Important: These steps must be performed on the Network Deployment server machine.

To start the deployment manager, complete the following steps on the Network Deployment machine.

1. Login as root.
2. Start terminal session.
3. Run the following command:

```
# cd <WAS_ND_HOME>/bin
# ./startManager.sh
```

4. Check the log files, SystemOut.log and SystemErr.log, under <WAS_ND_HOME>/logs/dmgr directory. The following message should be appeared in SystemOut.log for its successful start.

   ```
   [9/13/02 14:06:18:513 EDT] 75c55974 WsServer      A WSVR0001I: Server dmgr
   open for e-business
   ```

Add a node into an existing cell

Important: These steps must be performed on the WebSphere Application Server machine.

To add a node into an existing cell, complete the following steps on the WebSphere Application Server machine.

1. Login as root.
2. Start a terminal session.
3. Run the following command to connect to the deployment manager process and to add this node into the existing cell:

   ```
   # cd <WAS_BASE_HOME>/bin
   # ./addNode.sh <dmgr host> <dmgr port> -includeapps
   ```

   Where `<dmgr host>` is a hostname of the Network Deployment server machine and `<dmgr port>` is the port number to which the deployment manager process listens. The default value of the port number is 8879. If you have installed Network Deployment and WebSphere Application Server on the same machine use “localhost” as the hostname.

Note: The applications in a node will not be included into the cell by default. If it is needed, `-includeapps` has to be specified as an option of the addNode.sh. It is also specified in this example. In the later sections in this chapter, it is assumed that this option is specified.

4. When the process has completed, the following message will be shown on the screen.

   ```
   ADMU0003I: Node <node name> has been successfully federated.
   ```

   Where `<node name>` is a node name of the WebSphere Application Server. The addNode.sh also starts the node agent process.

10.7.3 Verify the cell

Currently, a deployment manager process is started on the Network Deployment machine and a node agent process is started on the WebSphere Application Server machine.

To confirm that the node was successfully added into the cell:

1. Access the admin console on the Network Deployment machine.
2. Start the application server on the WebSphere Application Server machine.

This section also illustrates that how an application server on the WebSphere Application Server machine is started from the admin console on the Network Deployment machine.

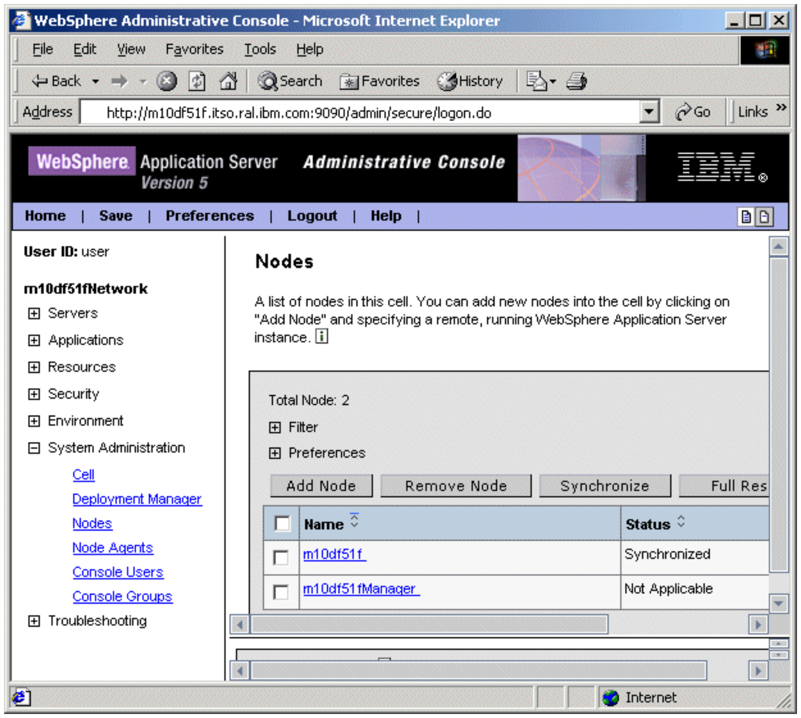
Access the network deployment admin console

To access to the admin console:

1. Using a Web browser, request the following URL:

   ```
   http://<dmgr hostname>:9090/admin
   ```

2. Enter a user ID to login.
3. Click **System Administration -> Nodes**. A window similar to the one shown in Figure 10-22 is displayed in the browser. The node name which was previously added is displayed in the window. In this example, name `m10df51f` is the node name to be added.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: user
m10df51fNetwork
Servers
Applications
Resources
Security
Environment
System Administration
Cell
Deployment Manager
Nodes
Node Agents
Console Users
Console Groups
Troubleshooting
Add Node
Remove Node
Synchronize
Full Res
Internet

Nodes

A list of nodes in this cell. You can add new nodes into the cell by clicking on "Add Node" and specifying a remote, running WebSphere Application Server instance.

Total Node: 2

Filter

Preferences

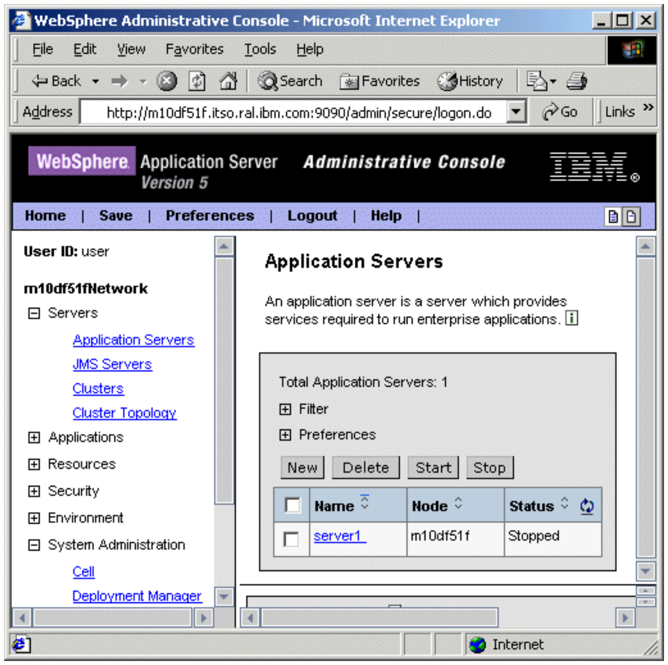
	Name	Status
☐	m10df51f	Synchronized
☐	m10df51fManager	Not Applicable

Figure 10-22 Node in Cell

Start the application server

To start the application server, the following steps should be performed after the steps shown in “Access the network deployment admin console” on page 329.

1. Click **Servers -> Application Servers**. A window similar to the one shown in Figure 10-23 on page 331 is displayed in the browser. The server name which was previously added is displayed in the window. Confirm that the status of `server1` is `Stopped`.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9090/admin/secure/logon.do Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
User ID: user
m10df51fNetwork
Servers
Application Servers
JMS Servers
Clusters
Cluster Topology
Applications
Resources
Security
Environment
System Administration
Cell
Deployment Manager
Internet

Application Servers

An application server is a server which provides services required to run enterprise applications.

Total Application Servers: 1

Filter

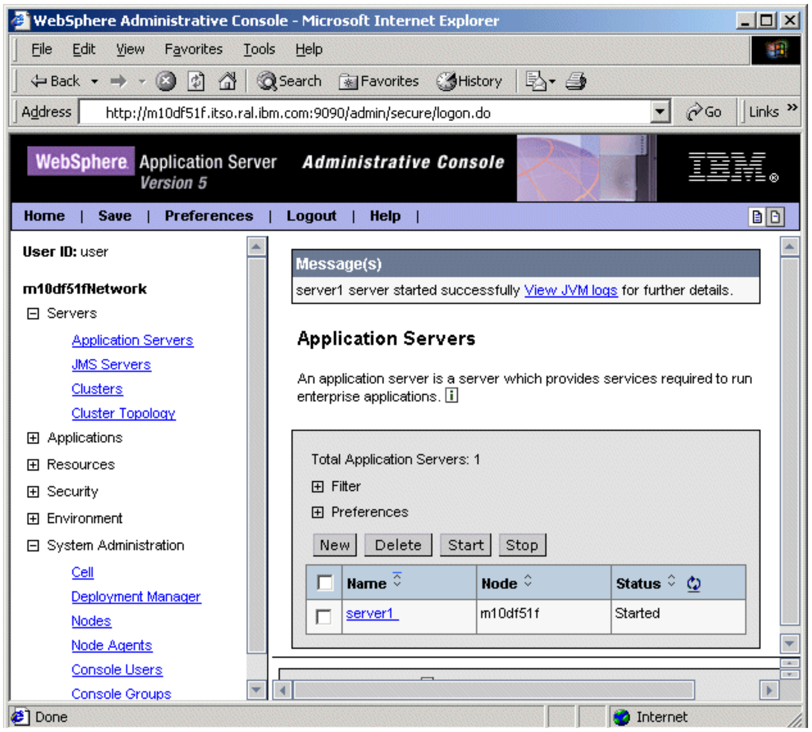
Preferences

New Delete Start Stop

	Name	Node	Status
☐	server1	m10df51f	Stopped

Figure 10-23 Application servers

2. Select `server1` and click **Start**.
3. Confirm that the status of `server1` is changed from `Stopped` to `Started`.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
User ID: user
m10df51fNetwork
Servers
Application Servers
JMS Servers
Clusters
Cluster Topology
Applications
Resources
Security
Environment
System Administration
Cell
Deployment Manager
Nodes
Node Agents
Console Users
Console Groups
Filter
Preferences
New Delete Start Stop
Done
Internet

Message(s)

server1 server started successfully View JVM logs for further details.

Application Servers

An application server is a server which provides services required to run enterprise applications.

Total Application Servers: 1

	Name	Node	Status
	server1	m10df51f	Started

Figure 10-24 Application server started

4. To verify that the server was started, use a Web browser to access the snoop servlet on the application server:

   http://<hostname>:9080/snoop

   Where `<hostname>` is a hostname of the WebSphere Application Server machine.

Update the Web server plug-in (Optional)

Author Comment: This section is added because this information is important for users. If this is not a good place for this section, move to anywhere!

Note: This section might be useful if you want to do more than just adding a single node to a cell.

Generally, the Web server plug-in file should be updated when add/remove nodes, add/remove/change virtual hosts, add/remove host aliases of HTTP transport, install/uninstall applications, and so on. Updating the Web server plug-in file from admin console of the WebSphere Depolyment Manager updates only the plug-in file in <WAS_ND_HOME>/config directory in the WebSphere Depolyment Manager server machine. To reflect the updates to the plug-in file in the cell, the following actions are required:

1. Copy the <WAS_ND_HOME>/config/plugin-cfg.xml file from the WebSphere Network Deployment server machine to the <WAS_BASE_HOME>/config directory on the Web server machine.
2. Change all the occurrence of <WAS_ND_HOME> directory with <WAS_BASE_HOME> directory in the plug-in file.
3. Restart the Web server in the Web server machine.
4. Restart the application servers in the WebSphere server machine.

10.8 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.

Author Comment (Arihiro): I think this is a repeat .. from somewhere in this chapter, possibly 10.3.5, “Enable SSL encryption for requests (optional)” on page 293
=> I don’t think this is a repeat. The configuration of SSL between plugin and App Server appeared for the first time in this chapter. To execute the ikeyman of WebSphere also appeared for the first time here. I think only to start IHS’s ikeyman is repeat. Or SSL setting should be put into one place.

Author Comment (security resident): “Configure WebSphere HTTP transport for SSL” in Windows installation chapter should be written and will be pointed here.

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_BASE_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 command:

```
# gsk5ikm
```

10.9 Install WebSphere Application Server - silent mode

This section describes how to install WebSphere Application Server using the non-interactive, or silent, mode. To complete a silent installation, you will create a customized response file from the default one, then execute the installation script for WebSphere Application Server, supplying the response file as a command-line parameter.

In this example, the instructions assume the following:

- Your machine has sufficient memory and disk space for your installation.
- 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 “Migration” on page 581.

Author Comment (Carla): There is no migration chapter ..

- 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.

10.9.1 Default response file

A default response file, named responsefile.txt, is supplied with WebSphere Application Server. You can use this default response file to install WebSphere Application Server as a template for creating a customized response file.

With default options, the following software and other resources are installed:

- IBM Java 2 Software Developer's Kit (SDK) 1.3.1
- IBM HTTP Server 1.3.26
- IBM WebSphere Application Server V5.0
- Embedded Messaging
- Performance And Analysis Tools
- 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/IBMHttpServer. In addition, WebSphere Application Server is configured for use with IBM HTTP Server when you use the default response file.

10.9.2 Customized response file

The default response file is used as a template for creating a customized response file. The default response file can be edited to select the components of WebSphere Application Server 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.

The following lines are the example which lines should be updated at least for the basic installation.

```
-P wasBean.installLocation="/usr/WebSphere/AppServer"
-P ihsFeatureBean.installLocation="/usr/IBMHttpServer"
-W nodeNameBean.nodeName="m10df51f"
-W nodeNameBean.hostName="9.24.104.21"
-W defaultIHSConfigFileLocationBean.value="/usr/IBMHttpServer/conf/httpd.conf"
```

In this example, the following items are specified in the response file respectively.

- WebSphere installation directory
- IBM HTTP Server installation directory
- node name
- host name or IP address
- configuration file of IBM HTTP Server

Important: All values should be enclosed in double quotes (““).

10.9.3 Perform 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 1.3.26 as part of the WebSphere Application Server installation and you have a level of IBM HTTP Server prior to 1.3.26 on your system, you must uninstall it for the WebSphere Application Server installation program to install IBM HTTP Server 1.3.26.
3. Ensure that the DISPLAY and TERM environment variables set correctly. They can be set by the following commands:

```
export DISPLAY=<IP address of host>:0.0
export TERM=vt100
```

Author Comment (Arihiro): how and what should they be set to?
=> Done.

4. Load the IBM WebSphere Application Server V5.0 CD-ROM into the CD-ROM drive and mount the CD.

```
# mount -r -v cdrfs /dev/cd0 /mnt
```

Author Comment (Arihiro): what is the command?
=> Done

5. Navigate to the /mnt directory.
6. 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 responsefile.txt <new_responsefile.txt>
```

In this command, `<new_responsefile.txt>` represents the full path name of the copy of the default response file you are creating (for example, `/tmp/my`_responsefile.txt).

7. To create a customized response file, perform the following steps:
   a. Use a text editor to open your copy of the default response file, <new_responsefile.txt>.
   b. Use the detailed comments throughout the file to help you select the appropriate options for your WebSphere Application Server installation. See "Customized response file" on page 335 for minimum changes of the response file.
   c. Save the changes that you have made to the customized response file.
8. Run the installation script by using the following commands.

```
# ./install.sh -options <new_responsefile.txt>
```

   The install.sh script uses the response file to install the components and options that you have selected. The variable <new_responsefile.txt> 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_responsefile.txt).
9. After installation is completed, refer to the log file named log.txt located in the /tmp directory to determine if the silent installation was successful. A copy of this file also exists in the directory <WAS_BASE_HOME>/logs.
10. Unmount the CD-ROM before removing it from the CD-ROM drive by using the **`umount`** command, as follows:

```
# umount /mnt
```

11. 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 commands:

```
# cd <http_server_install_path>/bin
# ./apachectl start
```

    b. Start a Web browser and type the name of the host machine as the URL (http://<host_machine> where host_name is a hostname of WebSphere Application Server server machine). If you see the Welcome to the IBM HTTP Server Web page in Figure 10-4 on page 291, the server has been installed correctly.

See “Change the port number of admin console” on page 301 to avoid a port number conflict. After that, see “Verify the WebSphere installation” on page 302 for verification of the installation.

10.10 Install Network Deployment - silent mode

This section describes how to install WebSphere Application Server Network Deployment using the non-interactive, or silent, mode. To complete a silent installation, you will create a customized response file from the default one, then execute the installation script for WebSphere Application Server, supplying the response file as a command-line parameter.

These instructions assume that your machine has sufficient memory and disk space for your installation.

Author Comment (Carla): where can I see the amount of space that will be needed? Is that in a readme and how can I see the readme before installation? The answer to these questions should be at the beginning of each Install section.

10.10.1 Default response file

A default response file, named responsefile.txt, is supplied with WebSphere Application Server Network Deployment. You can use this default response file to install WebSphere Application Server as a template for creating a customized response file.

With default options, the following software and other resources are installed:

- IBM Java 2 Software Developer's Kit (SDK) 1.3.1
- WebSphere Application Server Network Deployment
- Embedded Messaging
- Web Services

10.10.2 Customized response file

The default response file is used as a template for creating a customized response file. The default response file can be edited to select the components of WebSphere Application Server Network Deployment 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.

The following lines are the example which lines should be updated at least for the basic installation.

```
-P wasBean.installLocation="/usr/WebSphere/DeploymentManager"
-W nodeNameBean.nodeName="m10df51fManager"
-W nodeNameBean.cellName="m10df51fNetwork"
-W nodeNameBean.hostName="m10df51f"
```

Author Comment (Carla): there is probably a naming convention implied here that we should probably have introduced somewhere.

In the example, the following items are specified in the response file respectively.

- Network Deployment installation directory
- Node name
- Cell name
- Host name or IP address

Important: All values should be enclosed in double quotes (““).

10.10.3 Perform a silent Installation

Perform the following steps to create a customized response file (if desired) to install WebSphere Application Server Network Deployment. 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. Ensure that the DISPLAY and TERM environment variables set correctly. They can be set by the following commands:

   ```
   export DISPLAY=<IP address of host>:0.0
   export TERM=vt100
   ```

Author Comment (Arihiro/Carla): how and what to? check the following instruction to call the CD by the correct name.
=> Changed to use commands.

3. Load the IBM WebSphere Application Server Network Deployment V5.0 CD-ROM into the CD-ROM drive and mount the CD.

   ```
   # mount -r -v cdrfs /dev/cd0 /mnt
   ```

4. Navigate to the /mnt directory.

5. 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 responsefile.txt <new_responsefile.txt>
   ```

   In this command, `<new_responsefile.txt>` represents the full path name of the copy of the default response file you are creating (for example, `/tmp/my`_responsefile.txt).

6. To create a customized response file, perform the following steps:

   a. Use a text editor to open your copy of the default response file, <new_responsefile.txt>.

   b. Use the detailed comments throughout the file to help you select the appropriate options for your WebSphere Application Server Network Deployment installation. See “Customized response file” on page 338 for minimum changes of the response file.

   c. Save the changes that you have made to the customized response file.

7. 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_responsefile.txt> 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_responsefile.txt).

   ```
   # ./install.sh -options <new_responsefile.txt>
   ```

8. After installation is completed, refer to the log file named log.txt located in the /tmp directory to determine if the silent installation was successful. A copy of this file also exists in the directory <WAS_ND_HOME>/logs.

9. Unmount the CD-ROM before removing it from the CD-ROM drive by using the **`umount`** command, as follows:

   ```
   # umount /mnt
   ```

11

Solaris installation steps

This chapter provides detailed procedures for installing, configuring, and verifying the components in the scenarios described in Chapter 8, “Installation approach” on page 127. 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:

- Installation planning
- Operating system considerations
- Installing Sun ONE Web Server, Enterprise Edition (formerly iPlanet Web Server) 6.0 SP1
- Installing IBM WebSphere Application Server
- Installing IBM WebSphere Application Server Network Deployment
- Installing the WebSphere HTTP plug-in on the remote Web server
- Installing IBM WebSphere Application Server in silent mode

11.1 Planning

This section defines the hardware and software used within the Solaris environment to test a number of different IBM WebSphere Application Server configuration scenarios.

11.1.1 Hardware and software prerequisites

The current hardware and software requirements can be found at
`http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html`

Author Comment: Remove and refer to product doc on web site. The requirements are also listed in the class material.

WebSphere has the following hardware and software requirements.

Hardware

- Sparc workstation at 440MHz or faster
- Minimum 384MB memory, 512 MB recommended
- 350 MB disk space (minimum) for WebSphere Application Server
- 130 MB disk space (minimum) for Sun ONE Web Server
- At least 400MB swap space, or twice the physical RAM size, whichever is greater.

Software

- Any workstation running Solaris 8 at a maintenance level of October 2000.
- IBM WebSphere Application Server 5.0 base and Network Deployment
- Netscape Communicator 4.7.9
- Sun ONE Web Server, Enterprise Edition (formerly iPlanet Web Server) 6.0 SP1
- IBM HTTP Server 1.3.24
- 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. iPlanet Web Server 6.0, Enterprise Edition Installation Guide:

   `http://docs.sun.com`

11.1.2 Software used in our test environment

We used the following software in our test environment:

- Solaris 8 + prerequisites
- Netscape iPlanet Web Server, Enterprise Edition V4.1 SP7 (4.1.7)

Author Comment(shafkat): .. did we use this or the new sun one v6?

- Netscape Communicator 4.76 for Solaris
- Sun JDK 1.3.0.3 (included in WebSphere Application Server package)

Product installation root variables

The variables listed in are used frequently throughout this documentation to represent the root installation directories of the software components.

Table 11-1 Variables

variable	Value (for Solaris systems)	Represents:
<WAS_ROOT>	/opt/WebSphere/AppServer	WebSphere
<plugin_install_path	/opt/WebSphere/AppServer	WebSphere HTTP plugin
<http_server_install_path>	/opt/sunOne/server6	Netscape IPlanet

11.1.3 Hardware used in our test environment

This section describes the hardware used within our test IBM WebSphere Application Server environment on Solaris.

Server 1 (“sun1”)

This server hosts the iPlanet Web Server. It’s configuration is as follows:

- 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 (“sun1”)

This server hosts the IBM WebSphere Application Server and IBM WebSphere Application Server Network Deployment installations. It’s configuration is as follows:

- Sun SPARC Ultra 60 (600-6486-01)
- 450 MHZ UltraSPARC II CPU
- 1 GB RAM
- 17 GB hard disk
- 1 Ethernet adapter (built-in)

11.2 Operating system considerations

We are not going to cover the actual installation of Solaris, but if you are interested, you can see the steps we used outlined in Appendix A of *WebSphere Commerce V5.4 Handbook* , SG24-6567*.*

The intent of this section is to make sure that the operating system is prepared for the WebSphere installation.

11.2.1 Checking the operating system level

The first thing to check is the level of the Solaris operating system to make sure it is at the optimum level. You can check the operating system level with the **`uname-a`** command as shown in Example 11-1. Note that the numbering convention for the Solaris operating system is unique. Solaris 8 appears as release 5.8 in the command results.

Example 11-1 checking the operating system level

```
# uname -a
SunOS sun2 5.8 Generic_108528-16 sun4u sparc SUNW,Ultra-60
```

You may find a second command, **`showrev`**, useful. It displays both the operating system and the hardware platform. The output from this command is shown in Example 11-2.

Example 11-2 Output of the showrev command

```
# showrev
Hostname: sun2
Hostid: 80e5beae
Release: 5.8
Kernel architecture: sun4u
Application architecture: sparc
Hardware provider: Sun_Microsystems
Domain:
Kernel version: SunOS 5.8 Generic 108528-16 August 2002
```

Setting up the .profile file

The .profile file contains the necessary variables and commands for an operating environment. The file is located in the home directory of the user. Since we install Websphere as root, the .profile file is located in the / directory. The following should be done in the .profile:

1. Include a command to start the Korn shell.

   ```
   /usr/bin/ksh
   ```

2. Set up the PATH variable so the necessary commands can be found by the installation program.
3. Another set of variable is included in the .profile file is required for successful installation of embedded messaging service of WebSphere.

Author Comment (shafkat): Need to point out the specifics for items 2&3. What is it in the path statement below that we need (highlight)? What variables? They aren't in the .profile on sun1 or 2 .. is this referring to the kernal settings?

You can see our .profile file set up in Example 11-3.

Example 11-3 .profile

```
PATH=/usr:/usr/bin:/usr/sbin:/usr/openwin/bin:/usr/dt/bin:/usr/openwin/lib:/usr/local/X11/lib:.
LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/usr/local/X11/lib:/usr/dt/lib:/usr/openwin/lib
export LD_LIBRARY_PATH
/usr/bin/ksh
set -o vi
stty erase ^H
```

```
 export PS1=‘hostname‘ ’$LOGNAME:$PWD$ ’
~
```

Checking the TCP/IP configuration

The Solaris system must reside on and be configured properly for a TCP/IP network. Some of the ways to test this are:

- Hostname resolution
  - Display the host name with the **`hostname`** command. Make sure this is the host name you expect.
- Run **ifconfig -a** to see the configured network interface
- Ping the IP address for the system from another system to make sure you can reach it.
- If DNS is configured make sure name resolution works using the **`nslookup`** command.
- Make sure you can telnet to the machine from a remote system.
- Make sure the following ports are not used by any other program in /etc/services file
  - Port 8888 - iPlanet Web Server administration port
  - Port 9090 - WebSphere administration console port
- If you are using xterm or a CDE terminal, make sure you can export the display to your desktop machine from the Sun box. This check is important if you plan to install WebSphere remotely instead of from the local console attached to the Sun machine.

  ```
  export DISPLAY=<ip_addr of host>:0.0
  ```

Note: In our case we use remote console for installation and thereby we exported the display from the Sun system. To get the display on your machine you need to have a xterminal program running on your desktop. In our case we had Hummingbird Exceed running. However we found certain versions of Exceed didn’t work properly with the installation program and generated core dump. We found Hummingbird Exceed V7.0 and V6.2 worked but any version lower than 6.2 or even version 8 caused us problems.

Checking the Solaris patch levels

In addition to Solaris 8, there are also patch level requirements. There are two ways to find the prerequisite patches:

- The following URL can be checked for up-to-date patch level information required by WebSphere:

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

- The second way is use the prereqChecker.xml file included on the installation media. This file contains prerequisite information for all platforms.

Important: The information contained in the prereqChecker.xml file will not be updated so you should always check the Web site for new information.

At the time of the writing of this book, the patches recommended were as listed in Table 11-2. You can bypass these prereqs during installation but the results are unpredictable.

Table 11-2 Patches required

Solaris 8 patch ID	Description
108940-14	Motif 2.1 patch
108652-27	X11 6.4.1 Xsun patch
108921-11	CDE 1.4 dtwm patch
109147-16	Linker patch

Check if patch is installed

The **showrev** command can be used to check if a particular patch is installed or not. For example, Example 11-4 shows the command to check for the 108940 patch and the results.

Example 11-4 Checking a installed patch

```
# showrev -p | grep 108940
Patch: 108940-09 Obsoletes:  Requires:  Incompatibles:  Packages: SUNWmfrun,
SUN
Wdtbax
Patch: 108940-25 Obsoletes:  Requires: 108652-25 Incompatibles:  Packages:
SUNWm
frun, SUNWdtbax
#
```

In this case you can see 108940-25 has been installed (superceding 108940-14). If the patch is not installed, there will be no response to the showrev command.

Installation of new or updated patches

The simplest way to approach this may be to download the latest recommended Solaris 8 patch cluster and install it. The latest recommended Solaris 8 patch cluster can be found at the following site: `http://sunsolve.sun.com`.

To apply a patch cluster to the Solaris 8 operating system do the following:

1. Log in as root and start a terminal session
2. Download the recommended Solaris 8 patch cluster archive, 8_Recommended.zip file
3. Change the operating system to single user mode:

   ```
   # /usr/sbin/shutdown -iS -y -g0
   ```

4. Copy the archive to a temporary directory. This particular patch requires that the directory have 50 MB disk space free

   ```
   # cp 8_Recommended.zip /tmp
   ```

5. Unzip the archive file in the /tmp directory

   ```
   # cd /tmp
   # /usr/bin/unzip 8_Recommended.zip
   ```

6. Run the patch cluster installer script

   ```
   # ./install_cluster
   ```

   The patch installation takes few minutes depending on the size of the cluster and the speed of the machine.

7. Once install has finished, check the log file at /var/sadm/install_data/Solaris_8_Recommended_log to determine if the installation is successful. The log file may report a number of errors for patches that were not installed. This is common when there are patches that apply to components that are not installed.
8. After reviewing the log file it is necessary to reboot the system:

   ```
   # /usr/bin/shutdown -i6 -y -g0
   ```

Checking file systems

You will need to check the Solaris file systems to make sure there is enough free space in each of the file systems.

Author Comment: This list was created based on what we see in the install screens. Where can we check this before install?

The Table 11-3 shows the file system requirements. If you do not have this much free space available you will need to increase it before starting installation.

Table 11-3 Disk space requirement

Name of application	File system name	Disk space required (MB)
Websphere Application Server	/opt	450
Sun ONE Web Server Enterprise Edition 6.0 SP1	/opt	150

Table 11-4 shows the size the minimum size we found sufficient for our installation.

Table 11-4 Recommended files 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, Web server, Other programs	3000
/export/home	User home directory	Home directories	1000
/tmp	Temporary files		500

You can use **`df -k`** command to check the status of free space in file systems. You can see the output of the command in Example 11-5

Example 11-5 Checking File system spaces

```
# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t0d0s0     494235  263855  180957    60%    /
/dev/dsk/c0t0d0s6    2056211  705039 1289486    36%    /usr
/proc                      0       0       0     0%    /proc
fd                         0       0       0     0%    /dev/fd
mnttab                     0       0       0     0%    /etc/mnttab
swap                  155160      16  155144     1%    /var/run
swap                 1482328 1327184  155144    90%    /tmp
/dev/dsk/c0t0d0s5    6196234 1146876 4987396    19%    /opt
/dev/dsk/c0t0d0s7    7597873   54567 7467328     1%    /export/home
```

11.3 Installing the Sun ONE Web Server

This section provides detailed instruction for installing, configuring and verifying Sun ONE Web Server V6.0 SP1 on Solaris system. The section is organized into the following sections:

- Preinstallation tasks
- Install the Web server
- Configure the Web server
- Verify the Web server functionality

11.3.1 Preinstallation tasks

Prior to installing Sun ONE Web Server the following 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 Sun ONE Web Server refer to documentation at `http://docs.sun.com`.

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 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
```

11.3.2 Installing Sun ONE Web Server V6.0 SP1

To install the Web server complete the following steps:

1. Start the Solaris console and log in a root. Copy the Sun ONE Web Server tar file either from the CD or from the internet into a temporary directory on your system:

```
# mkdir -p /opt/ipv61
# cd /cdrom/solaris/enterprise
# cp enterprise-6[1].0SP1-domestic-us.sparc-sun-solaris2.6.tar.gz
/opt/ipv61
```

1. Uncompress and extract the files from the from the tar file as follows

```
# gunzip enterprise-6[1].0SP1-domestic-us.sparc-sun-solaris2.6.tar.gz
#
# tar -xvf  enterprise-6[1].0SP1-domestic-us.sparc-sun-solaris2.6.tar
```

2. Start the Web server installation from the installation directory with the following command:

```
# ./setup
```

3. 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.

4. When you are prompted with the message `Do you agree to license terms? [No]:` type Yes, and then press Enter.
5. When you are prompted with the message `Choose the installation type [2]:` press Enter to accept the default (Typical installation).
6. When you are prompted with the message `Install location [/usr/netscape/server4]:` enter the path `/opt/sunOne/server6`, then press Enter.
7. When you are prompted with the message `Specify the components you wish to install [All]:` press Enter.

8. 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 - Search and Indexing Support
> - 5 - SNMP support

9. When you are prompted with the message “`Computer name [your_hostname.domain]`**:”,** press Enter.
10. When you are prompted with the message “System User [nobody]:”, press Enter.
11. When you are prompted with the message “System Group [nobody]:”, press Enter.
12. When you are prompted with the message “Run iWS Administration Server as [root]:”, press Enter.
13. When you are prompted with the message “iWS Administration Server User Name [admin]:”, `type` **`root`** `and` press Enter. When prompted for a password, type it in and press Enter.
14. When you are prompted with the message “iWS Admin Server Port [8888]:”, press Enter.
15. When you are prompted with the message ”Web Server Port [80]:” press Enter.
16. When you are prompted with the message “Web Server Content Root [/opt/netscape/server4/docs]:”, press Enter.
17. When you are prompted with the message “Do you want to use your own JDK [No]:”, press Enter
18. When you are prompted with the message “Do you want to register this with an existing Directory Server [No]:”, press Enter.
19. Review messages to make sure everything was extracted and installed successfully. Press Enter to continue and return to the command prompt.

The iPlanet Web Server installation is complete.

11.3.3 Configure the Web server

After installation of Sun ONE Web Server, the following configuration tasks must be completed on the Web server machine:

1. Create a system startup script to automatically start the Web server.
2. SSL configuration (optional).

Create the 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 Sun ONE 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 11-6 provides a sample startup script.

Example 11-6 Sample iPlanet startup script

```
#!/bin/ksh
#
# Filename: iplanet
# Purpose: Cleanly startup/shutdown the iPlanet Web Server on system
# startup/shutdown respectively.

IPLANET_HOME=/opt/sunOne/server6
INSTANCES="sun2.itso.ral.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 SSL configuration

In this section we describe the procedure to configure SSL for the Sun ONE Web Server. This section is organized into following steps:

- Create a new Web server instance
- Create a certificate trust database
- Request a server certificate
- Install the server certificate on SSL server
- Turn on SSL encryption

Create a new Web server instance (port 443)

Each Web server port requires a Web server instance. By default, the port 80 Web server instance is created during the installation. To create a Web server instance for the SSL port, 443, complete the following steps:

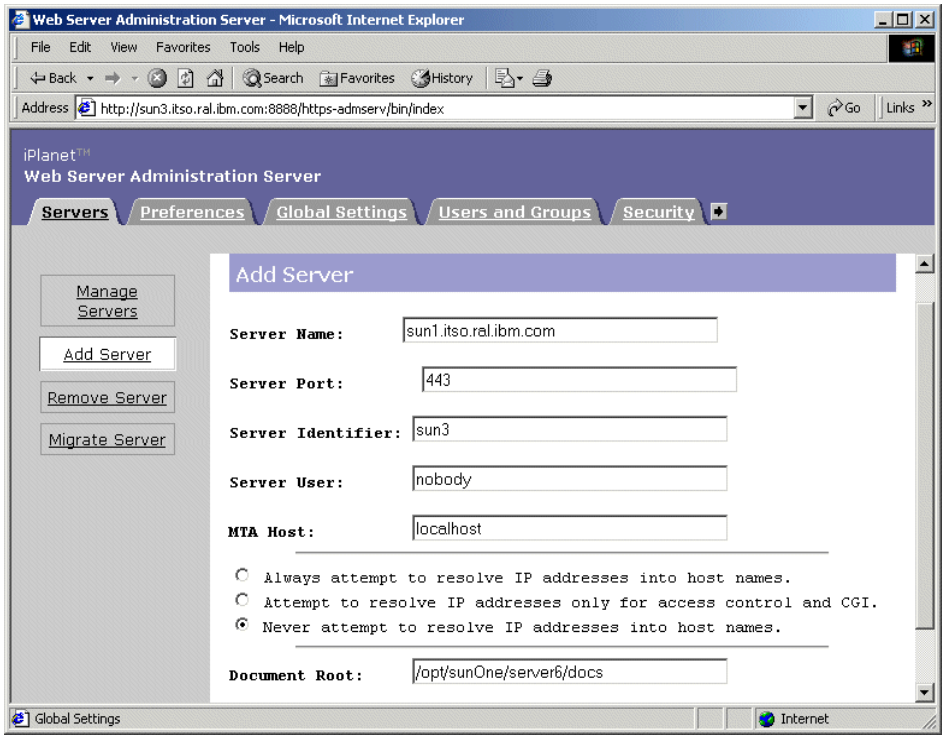
- Start the Sun ONE Web Server administration panel by requesting the following URL from your desktop browser:

  ```
  http://<fully qualified hostanme>:8888/
  ```

  In our case:

  ```
  http://sun1.itso.ral.ibm.com:8888
  ```

- Select the Servers tab.
- Click **Add Server** in the left navigation pane.
- When the Add Server window appears, enter the requested information and then click **OK**. The values we used are shown in Figure 11-1.


Web Server Administration Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://sun3.itso.ral.ibm.com:8888/https-admserv/bin/index
Go Links
iPlanet™
Web Server Administration Server
Servers Preferences Global Settings Users and Groups Security
Manage Servers
Add Server
Remove Server
Migrate Server
Add Server
Global Settings
Internet

Server Name: sun1.itso.ral.ibm.com

Server Port: 443

Server Identifier: sun3

Server User: nobody

MTA Host: localhost

( ) Always attempt to resolve IP addresses into host names.
( ) Attempt to resolve IP addresses only for access control and CGI.
[✓] Never attempt to resolve IP addresses into host names.

Document Root: /opt/sunOne/server6/docs

Figure 11-1 Create a server instance

Tip: When the Web server is added it generates the following path for the Web server:

```
/usr/netscape/server/https-<identifier>
```

For example, we created the following for host name sun1:

```
/usr/netscape/server4/https-sun1-443
```

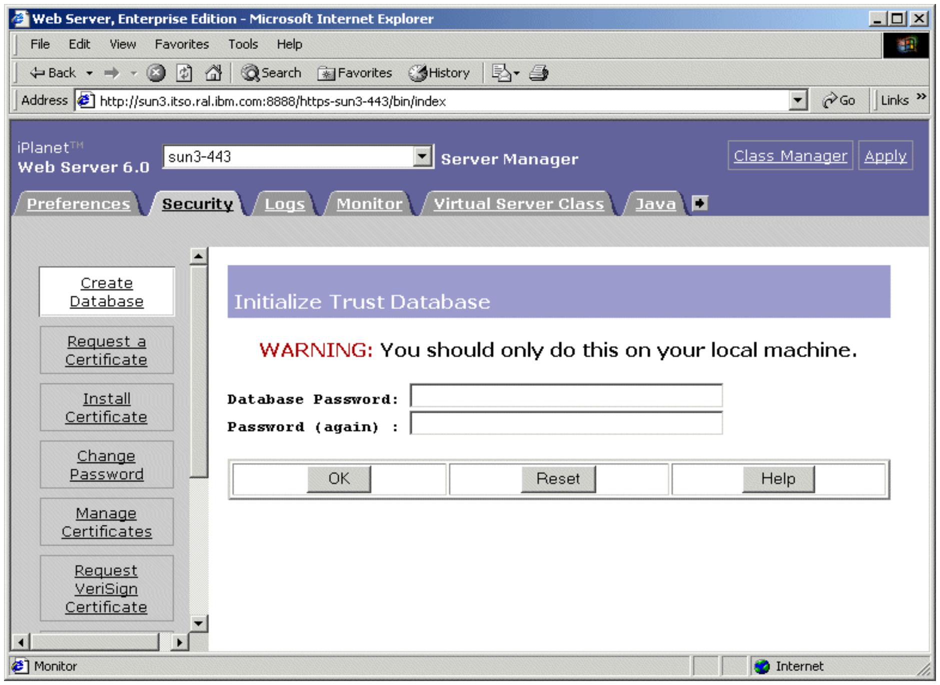
Create a certificate trust database

Certificates key pairs are stored in a database installed on the local host. Each Web server instance has its own certificate key pair, referred to as a trust database. In our scenario, we will need to create one certificate trust database (sun1-443). To create the certificate trust database needed for SSL enabled servers (443), complete the following steps:

- From the Manage Server window, select the sun1-443 server from the Select a Server pull-down, and then click **Manage**.

- Click the Security tab from the top of the page.
- Click the **Create Database** link from the left-hand side of the page.
- Enter the database password as shown in Figure 11-2 on page 356. Click **OK**.

A pop-up window should appear, indicating that a database was successfully created.


Web Server, Enterprise Edition - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://sun3.itso.ral.ibm.com:8888/https-sun3-443/bin/index
Go Links
iPlanet™ Web Server 6.0
sun3-443
Server Manager
Class Manager
Apply
Preferences Security Logs Monitor Virtual Server Class Java
Create Database
Request a Certificate
Install Certificate
Change Password
Manage Certificates
Request VeriSign Certificate
OK Reset Help
Monitor
Internet

Initialize Trust Database

WARNING: You should only do this on your local machine.

Database Password:

Password (again) :

Figure 11-2 Create a trusted database for Web server instance sun1-443

Request a server certificate

For each SSL server you have to request a server certificate. Perform the following steps to generate a certificate request:

- From the Administration Server, select the **Security** tab.
- Click the **Request a Certificate** link in the left navigation pane.
- When the Request a Server certificate window appears, do the following:
  – Select the **New Certificate** radio button.
  – In the CA-email address field, enter your e-mail address.

- Select **Internal (software)** from the Cryptographic module pull-down.
- In the Key Pair File Password field, enter your Certificate Trust database password as previously created.
- In the Common Name field, enter the URL by which your host will be known. This is normally the fully qualified hostname, for example sun1.itso.ral.ibm.com .
- In the remaining fields, fully enter your contact information.
- Click **OK**.

- The next window will inform you that your request has been sent. Take note of the file name specified by the message, A copy of the certificate request has been saved in the file:<file_name>. The certificate will be e-mailed to you.
- Proceed to a Certificate authority Web site to submit your request. Follow the instructions provided there for the completion of your request.

Note: Most certificate sites will allow you to generate a test certificate for use on non-production servers, for example, a VeriSign Free Trial SSL ID can be found at: `http://www.verisign.com/products/site/index.html`.

Shortly after submitting the request using the request file created in the previous step you should receive a test certificate from VeriSign. Be aware that the test certificate is valid for 14 days only.

When you receive the certificate, cut and paste the lines beginning and ending (including those statements) with the following text to a file and copy it to the Web Server in preparation for installing the certificate.

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

You will see a whole bunch of characters in between the begin and end certificate. This is what makes the certificate unique.

Install the server certificate on each SSL server

When you receive the certificate, you have to install it on each SSL server instance. Perform the following steps:

- From the Server Manager pull-down menu, select the server instance you want to configure (for example, sun1-443 or sun1-8000). Click **Manage**.
- Click the **Security** tab for the server you have requested the certificate.

- Click **Install Certificate**. When the Install a Server Certificate window appears, enter the following and then click `OK`.
  - Certificate For: select **This Server**
  - Cryptographic Module: select **Internal (software)**
  - Key Pair File Password: <certificate_trust_database_password>Certificate Name : leave this field blank
  - Select the **Message Text** radio button.
- Paste the certificate sent to you by the certificate authority into the Message Text Area.

  When the Add Server Certificate window appears, take note of the information provided (for example, Certificate Name: Server-Cert) and click **Add Server Certificate**.

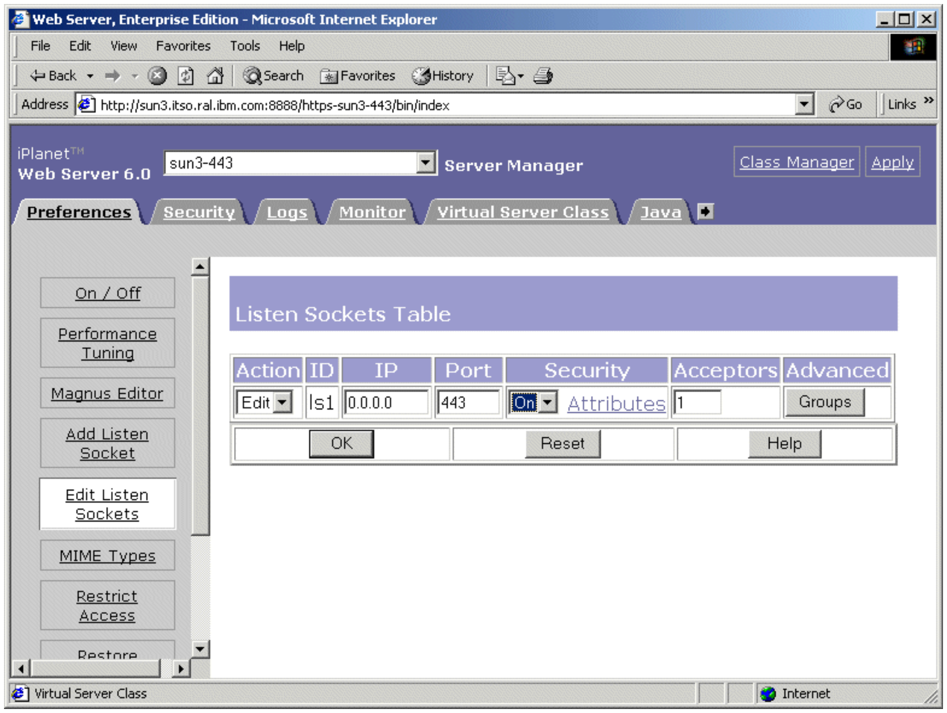
You should see a success window will pop up, as well as a window warning you that the server will need to be restarted for the security changes to take effect. Now you can turn on the SSL encryption.

Turn on SSL encryption

To enable or turn on SSL encryption, complete the following steps:

- From the Manage Servers window, select the server you want to configure (for example, sun1-443)
- Click the `preferences` tab.
- Click `Edit Listen Sockets` in the left navigation pane
- From the drop-down under the security column select `on` as shown in Figure 11-3
- Click **OK** to complete the operation

This completes the turning on SSL for a server instance.


Web Server, Enterprise Edition - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://sun3.itso.ral.ibm.com:8888/https-sun3-443/bin/index
Go Links
iPlanet™ Web Server 6.0
sun3-443
Server Manager
Class Manager Apply
Preferences Security Logs Monitor Virtual Server Class Java
On / Off
Performance Tuning
Magnus Editor
Add Listen Socket
Edit Listen Sockets
MIME Types
Restrict Access
Restore
Virtual Server Class
Internet

Listen Sockets Table

Action	ID	IP	Port	Security		Acceptors	Advanced
Edit	ls1	0.0.0.0	443	On	Attributes	1	Groups

Figure 11-3 Turning on SSL encryption for server instance sun1-443

11.3.5 Verify the installation

In order to verify the installation, perform the following checks on the Web Server machine:

1. Check the server status.
2. Check the process status.
3. Check request handling.

Check the server status

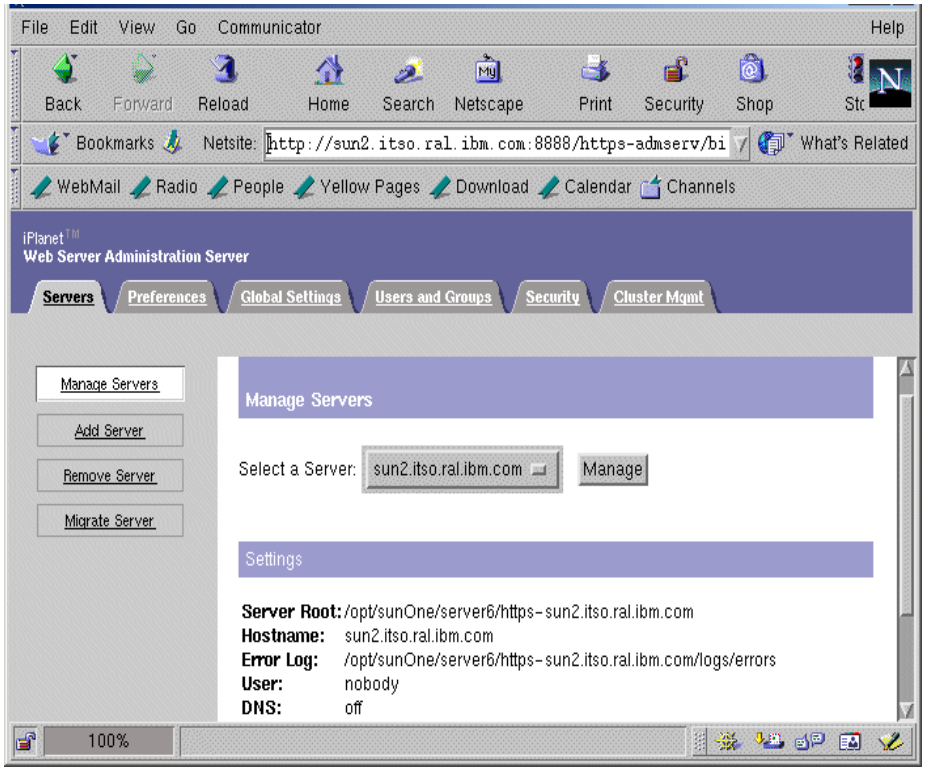
To check the Sun ONE Web Server status, perform the following steps:

1. Start the iPlanet Administration Server process and load its console:

```
# cd <http_server_install_path>
# ./startconsole
```

Note: If you use the **`startconsole`** command to start the Web server administration panel, the Netscape browser has to be installed on the Web server machine and your PATH variable must be set properly so the Netscape browser can be started from the command line. Alternatively you can access the administration panel from a Web browser using the following URL: http://<fully qualified host name>:8888/

2. In the Select a Server pull-down, select the Web server.

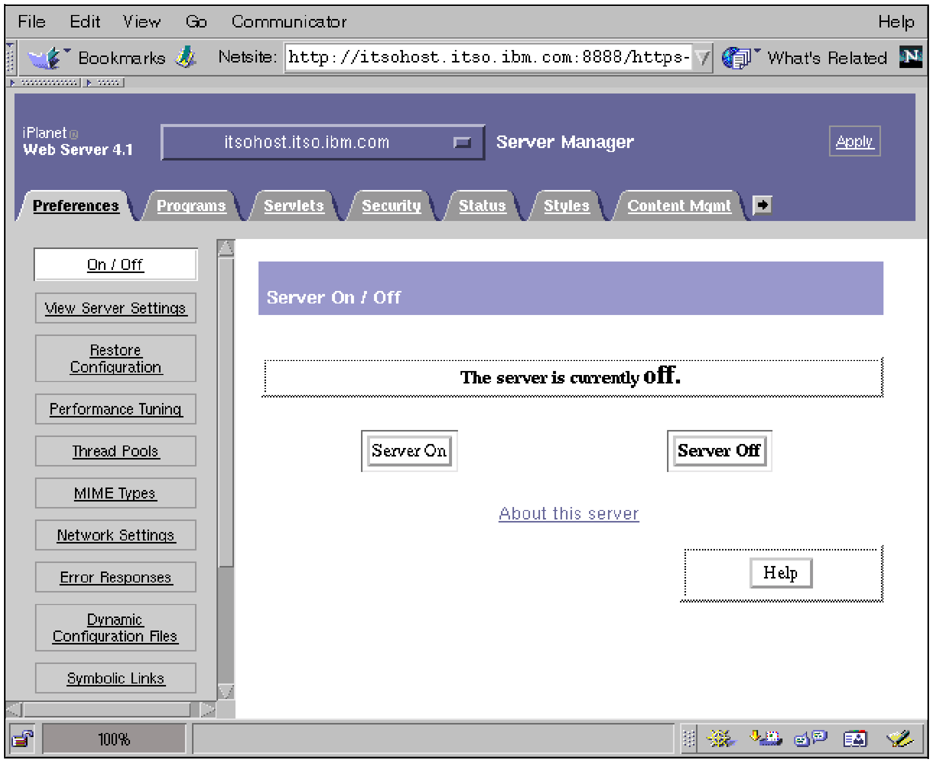

File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stc
Bookmarks Netsite: http://sun2.itso.ral.ibm.com:8888/https-admserv/bi What's Related
WebMail Radio People Yellow Pages Download Calendar Channels
iPlanet™ Web Server Administration Server
Servers Preferences Global Settings Users and Groups Security Cluster Mgmt
Manage Servers
Add Server
Remove Server
Migrate Server
Manage Servers
Manage
Settings
100%

Select a Server: sun2.itso.ral.ibm.com

Server Root: /opt/sunOne/server6/https-sun2.itso.ral.ibm.com
Hostname: sun2.itso.ral.ibm.com
Error Log: /opt/sunOne/server6/https-sun2.itso.ral.ibm.com/logs/errors
User: nobody
DNS: off

Figure 11-4 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 11-5. If the server is not running, click **Server On** to start the server.



The server is currently off.

Figure 11-5 Set server status to On

5. If the server starts successfully you should see the following message:

```
The server is currently on.
```

Check the process status

To check the Sun ONE Web Server process status, perform the following steps:

1. Check that the Web server processes are running by issuing the following command:

```
ps -ef | grep ns-httpd
```

   The output should list one process for the Web Server and one process for the administration server. For example:

Author Comment (shafkat): why are there two processes each?

Example 11-7

```
# ps -ef | grep ns-httpd
    root  3162  3160  0 14:03:23 pts/5    0:00 grep ns-httpd
    root  2528  2527  0   Sep 23 ?        0:01 ns-httpd -d
/opt/sunOne/server6/https-admserv/config
    root  2529  2528  0   Sep 23 ?        0:05 ns-httpd -d
/opt/sunOne/server6/https-admserv/config
    root  2596  2595  0   Sep 23 ?        0:03 ns-httpd -d
/opt/sunOne/server6/https-sun1.itso.ral.ibm.com/config
  nobody  2597  2596  0   Sep 23 ?        0:05 ns-httpd -d
/opt/sunOne/server6/https-sun1.itso.ral.ibm.com/config
    root  3133  3132  0 13:04:05 ?        0:02 ns-httpd -d
/opt/sunOne/server6/https-sun1-443/config
  nobody  3134  3133  0 13:04:08 ?        0:02 ns-httpd -d
/opt/sunOne/server6/https-sun1-443/config
```

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
   ```

   For example:

Example 11-8 Checking port 80

```
# netstat -a | grep LISTEN | grep 80
      *.80  *.*  0  0  24576  LISTEN
```

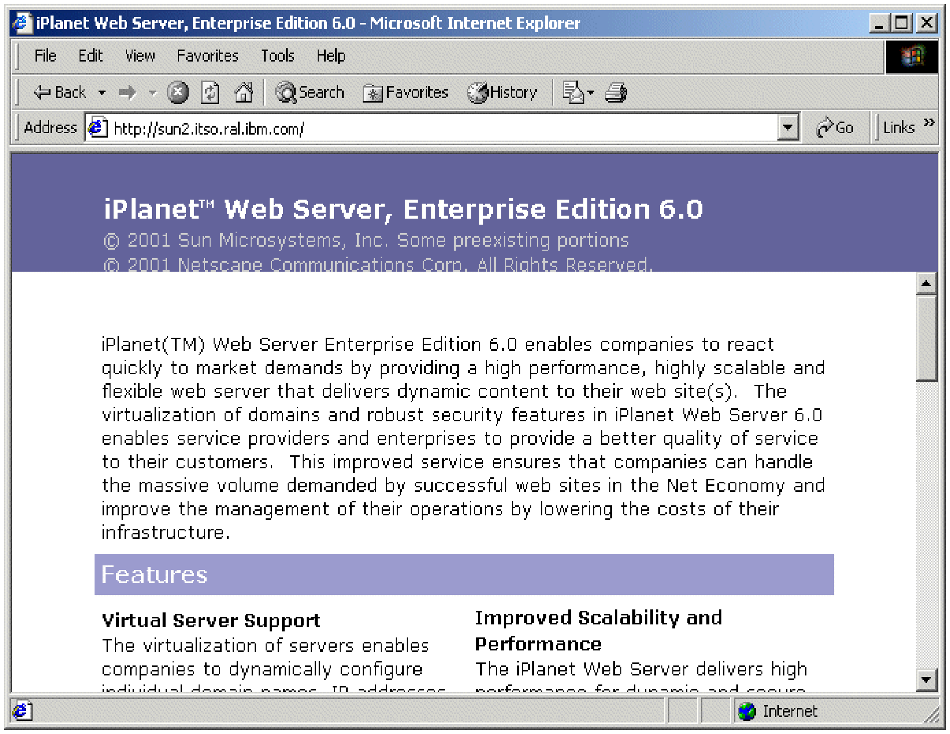
Check request handling

To check Sun ONE Web Server request handling, perform the following steps:

1. Using a Web browser, request the following URL representing the Web server home page:

   ```
   http://<hostname.domain.com>/
   ```

   The window shown in Figure 11-6 will be displayed if the Web server has been installed and configured correctly.


iPlanet Web Server, Enterprise Edition 6.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://sun2.itso.ral.ibm.com/
Go Links
Internet

iPlanet™ Web Server, Enterprise Edition 6.0
© 2001 Sun Microsystems, Inc. Some preexisting portions
© 2001 Netscape Communications Corp. All Rights Reserved.

iPlanet(TM) Web Server Enterprise Edition 6.0 enables companies to react quickly to market demands by providing a high performance, highly scalable and flexible web server that delivers dynamic content to their web site(s). The virtualization of domains and robust security features in iPlanet Web Server 6.0 enables service providers and enterprises to provide a better quality of service to their customers. This improved service ensures that companies can handle the massive volume demanded by successful web sites in the Net Economy and improve the management of their operations by lowering the costs of their infrastructure.

Features

Virtual Server Support
The virtualization of servers enables companies to dynamically configure

Improved Scalability and Performance
The iPlanet Web Server delivers high

Figure 11-6 Home page request handled by Web server

11.4 WebSphere Application Server

This section provides detailed instructions for installing, configuring, and verifying IBM WebSphere Application Server for Solaris.

The section is organized into the following tasks:

- Install IBM WebSphere Application Server
- Install IBM WebSphere Application Server Network Deployment
- Install the Websphere plug-in on the Web server host
- Install WebSphere in silent mode

11.4.1 Install IBM WebSphere Application Server

This section provides detailed instructions for installing, configuring, and verifying IBM WebSphere Application Server for Solaris. It is organized into the following tasks:

- Installing IBM WebSphere Application Server
- Verifying the installation.

Installing IBM WebSphere Application Server

To install IBM WebSphere Application Server using the GUI installer interface, complete the following steps:

1. Log in as root.
2. Start a terminal session.
3. Load the IBM WebSphere Application Server V5.0 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. Run the installation script:

   ```
   # ./LaunchPad.sh
   ```

6. In the Installation Options window, select a language to be used for installation and select **OK**.
7. The software license agreement dialog will appear. Read the license agreements and if you agree select the radio button next to “I agree with the terms of this license agreement and select **`Next`** to continue.
8. After accepting the license agreement, the installation will check for prerequisites on the installation machine.Once it has confirmed that all prerequisites have been met, the dialog with display the installation options.

   WebSphere provides two options. Select the **Custom** option and click **`Next`** to continue.

9. The next screen allows the user to select the installation options. In our case we selected everything EXCEPT the IBM HTTP Server, the Web server plug-ins, and the client only embedded messaging option.

   Note: You only need to select a Web server plug-in if you are installing or have installed a Web server on this machine.

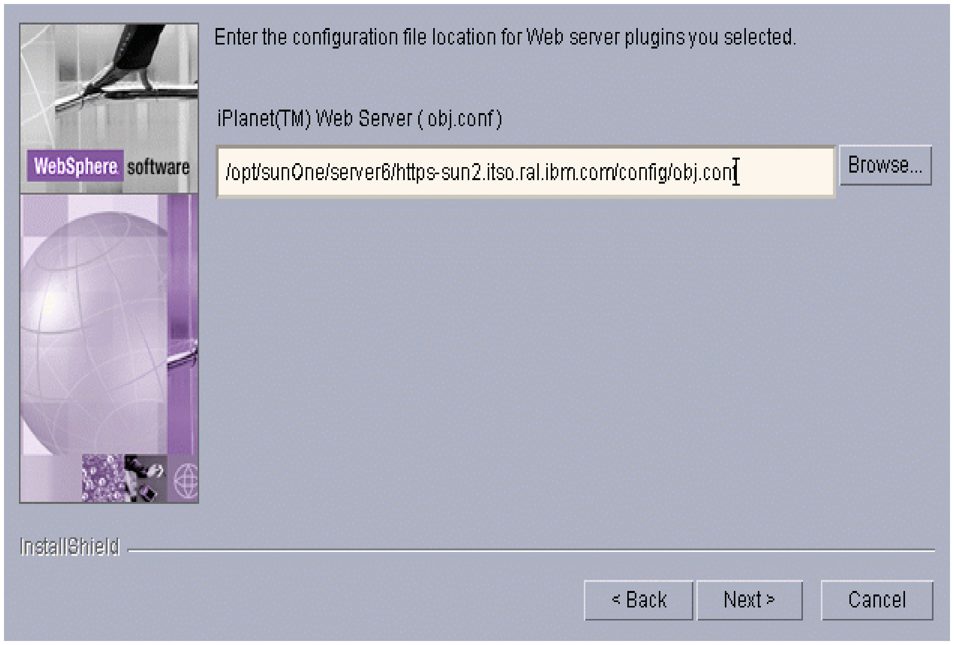
   You may want to consider installing the IBM HTTP Server on this machine if you are planning to implement SSL between the plug-in and the application server and will be using the kdb SSL key file format. It provides the ikeyman utility for this.

10. The next screen will enable the user to select the installation directories for WebSphere and the IBM HTTP Server. Ensure that directory path is set correctly. We recommend you install WebSphere in /opt/WebSphere/AppServer directory.

    We are not installing the IBM HTTP Server, so you if you are asked for the installation directory just take the default. Click **Next** to continue.

11. The next screen asks for the location of the iPlanet configuration file since we elected to install the plug-in file for iPlanet. Enter the location for the obj.conf Web server file. In our case the obj.conf file is located under the following directory:

    ```
    /opt/sunOne/server6/https-sun2.itso.ral.ibm.com/config/obj.conf
    ```


Enter the configuration file location for Web server plugins you selected.

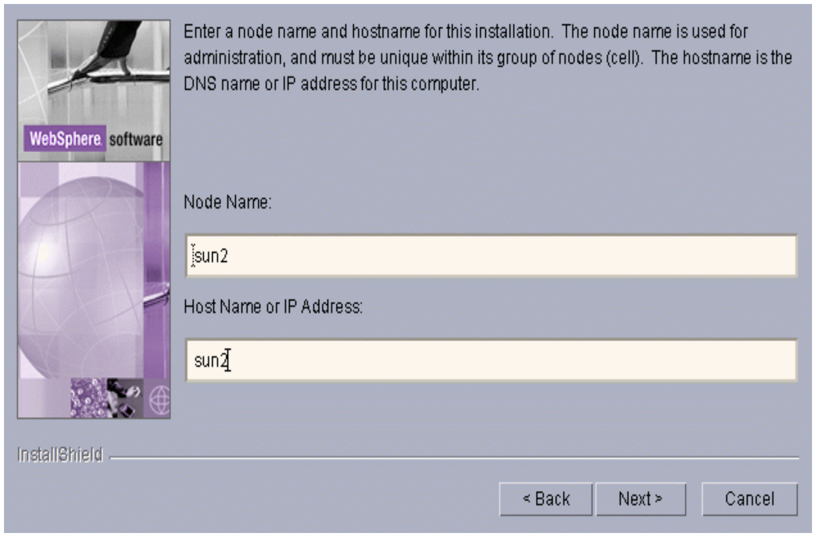
iPlanet(TM) Web Server (obj.conf)

/opt/sunOne/server6/https-sun2.itso.ral.ibm.com/config/obj.conf

Figure 11-7

12. The next dialog asks for the node name for your default node. We use the short host name of our machine as the node name. It is recommended that the node name be different than that of the full DNS name of the machine.

.


Enter a node name and hostname for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer.

Node Name:

sun2

Host Name or IP Address:

sun2

Figure 11-8 Providing Node name and Hostname

Click **Next** to continue

13. The next screen will show a summary of all of the choices that were made in the custom installation. Selecting Next will begin the copying of files to the specified directory structures. Once the selections have been reviewed, click **Next** to begin the installation.
14. A progress bar will indicate the status of installation. Once the installation is complete the installer will prompt the user to register. The user can choose to register the product immediately by checking the check box at the bottom of the dialog, or manually register at a later time by unchecking the checkbox and click **Next** to continue.

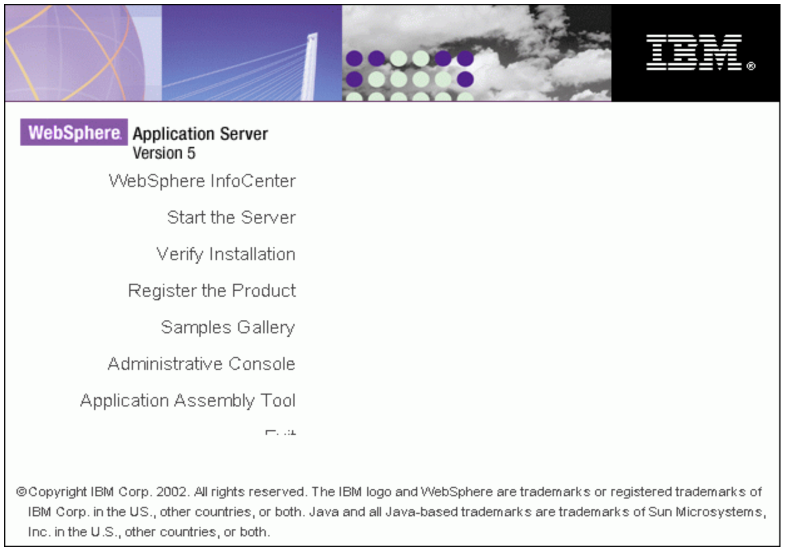
    The Confirmation screen will inform you that the installation was successful. Click **Finish** to complete the installation steps.

Verify WebSphere Installation

The First Steps screen, shown in Figure 11-9, will start automatically at the end of the installation.

Starting the first steps program: If you close the first steps window before completing all the tasks or want to open it sometime later, simple execute /opt/WebSphere/AppServer/bin/firststeps.sh

You can verify the installation by selecting by selecting **Verify Installation** from this screen.



©Copyright IBM Corp. 2002. All rights reserved. The IBM logo and WebSphere are trademarks or registered trademarks of IBM Corp. in the US., other countries, or both. Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the U.S., other countries, or both.

Figure 11-9 First Steps

The verification will do the following:

- Start the application server (server1) if it has not started yet and verify its status.
- Verify the servlet engine status
- Verify the JSP status
- Verify the EJB status

Note: The install verification test can also be started from the command line. To run the install verification from the command line please follow the steps:

- Change the directory to <WAS_ROOT>/bin directory

```
cd <WAS_ROOT>/bin
```

- Run the ivt.sh script

```
./ivt.sh
```

- Check the ivt.log file under <WAS_ROOT>/logs directory.

Other verification tasks

If you encountered problems during the installation for verification, there are some things you can do to check the status of the product:

- Check the installation log file
- Check changes of Web server configuration files
- Start application server to access administrative console

Check installation logs file

You can check the log files listed in Table 11-5 for more information about the installation. These files are available under <WAS_ROOT>/logs directory.

Table 11-5 Installation log files

Component	File Name
WebSphere Application Server	log.txt
Default Application	installDefaultApplication.log
Sample Application	installDefaultApplication.log
Administrative Console	installAdminConsole.log
MDB Samples Application	installMessagingSamples.log
IVT Application	installIVTApp.log

The installation wizard also creates the optional_log.txt file in the $TEMP directory. Check this file if necessary.

Check the Web server plug-in configuration changes

If you installed the HTTP plug-in on this system, you should see changes in two of the iPlanet configuration files:

- obj.conf

  The following line will be added to the obj.conf file in the <Object name=default> stanza

  ```
  Service fn="as_handler"
  ```

- magnus.conf

  The following lines will be added in the magnus.conf file:

  ```
  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"
  ```

Start the application server

The Installation Verification Test starts the default application server. It can also be started from the command line as follows:

- Change directory to <WAS_ROOT>/bin directory and run the startServer.sh script

  ```
  # cd <WAS_ROOT>/bin
  # ./startServer.sh server1
  ```

Check the log files SystemOut.log and SystemErr.log files under `<WAS_ROOT>/logs/server1` directory. The following messages should be appeared in the SystemOut.log file for successful start of the application server:

```
[9/16/02 18:29:29:650 EDT]  752a14 WsServer     A WSVR0001I: Server server1
open for e-business
```

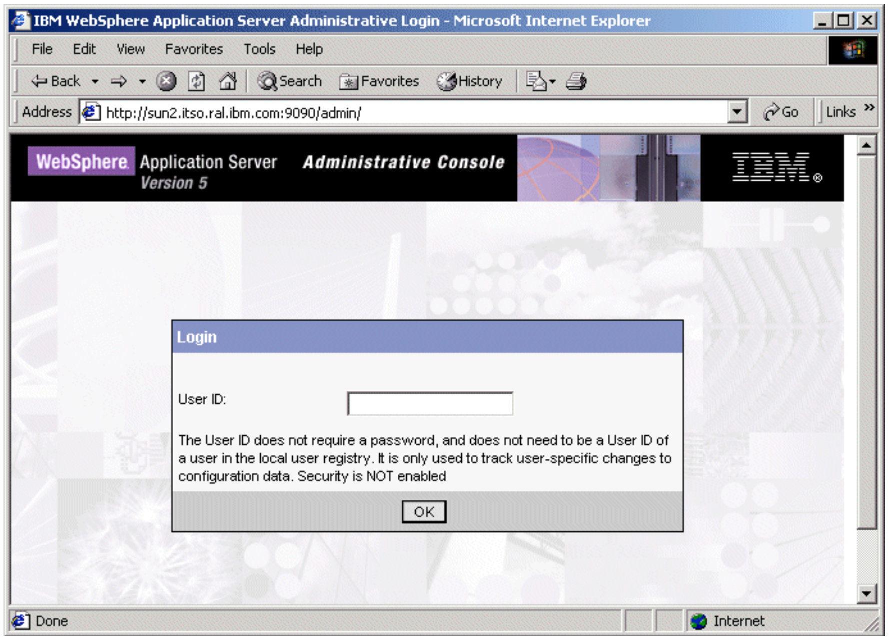
Open the administrative console

To access the administrative console for WebSphere:

- From a Web browser request the following URL:

  ```
  http://<hostname>:9090/admin
  ```

  A window similar to the one shown in Figure 11-10 is displayed in the browser.


IBM WebSphere Application Server Administrative Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://sun2.itso.ral.ibm.com:9090/admin/
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Done
Internet

Login

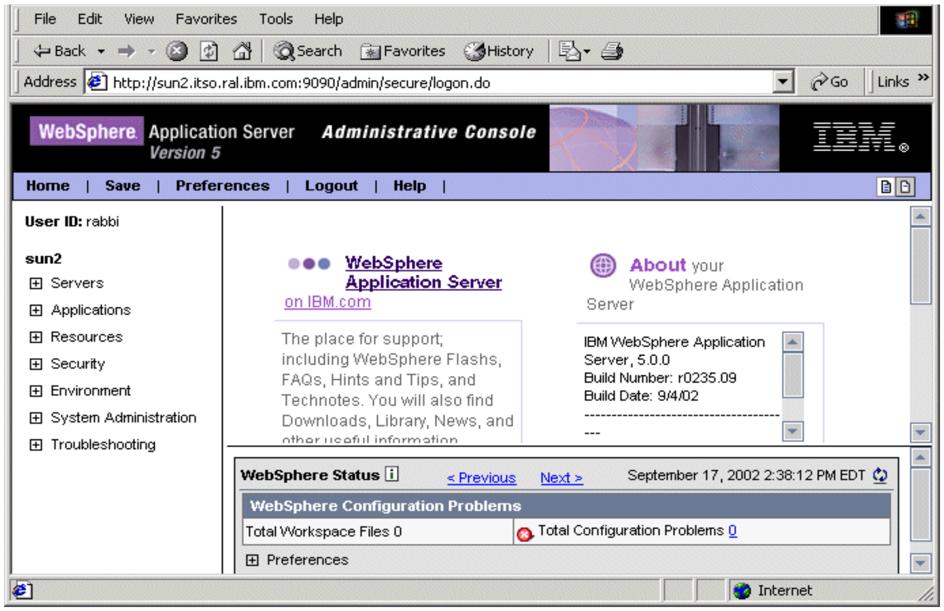
User ID:

The User ID does not require a password, and does not need to be a User ID of a user in the local user registry. It is only used to track user-specific changes to configuration data. Security is NOT enabled

OK

Figure 11-10 Administrative console

- Enter a user ID. At this stage we have not implemented any security. The user ID you enter here is used only for tracking purposes and is not authenticated.
- Click **OK** to continue. The administrative console will appear in the browser.


File Edit View Favorites Tools Help
Back Search Favorites History
Go Links
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
sun2
Servers
Applications
Resources
Security
Environment
System Administration
Troubleshooting
WebSphere Application Server on IBM.com
About your WebSphere Application Server
WebSphere Status
< Previous
Next >
WebSphere Configuration Problems
Preferences
Internet

Address http://sun2.itso.ral.ibm.com:9090/admin/secure/logon.do

User ID: rabbi

The place for support; including WebSphere Flashs, FAQs, Hints and Tips, and Technotes. You will also find Downloads, Library, News, and other useful information

IBM WebSphere Application Server, 5.0.0
Build Number: r0235.09
Build Date: 9/4/02

September 17, 2002 2:38:12 PM EDT

Total Workspace Files 0
Total Configuration Problems 0

Figure 11-11 Administrative console

Access the snoop servlet

The snoop servlet is provided with WebSphere and can be used to verify that the application server can be reached. For testing, you can use the WebSphere internal Web service (port 9080) to reach the application. You can access the application from a Web browser by using the following URL:

```
http://<websphere_hostname>:9080/snoop
```

A window similar to the one shown in Figure 11-21 on page 308 will appear in the browser. This confirms that application server started properly.

Generate and verify the WebSphere HTTP plug-in

Note: If you did not install the Web server and WebSphere plug-in on the same system as WebSphere, skip this step. 11.5, “Install the WebSphere HTTP plug-in on a remote node” on page 378 discusses installing and generating the WebSphere plug-in on a remote machine.

To verify that WebSphere plug-in is working properly with the Sun ONE Web Server do the following:

1. Make sure the application server is started. You can check this by doing the following:

   ```
   cd /opt/WebSphere/AppServer/bin
   ./serverStatus.sh -all
   ```

   If you need to start the server, issue the following command:

   ```
   ./startServer.sh server1
   ```

2. From a Web browser, access the administrative console using the following URL:

   ```
   http://<websphere_hostname>:9090/admin
   ```

3. Select **Environment** in the left navigation screen.
4. Click on the **Update Web Server Virtual Host** to regenerate the plug-in file. Click **OK.**
5. Try to access the snoop servlet (http://<web_server_hostname>/snoop). If the plug-in is configured properly you should see the snoop servlet.

11.4.2 Install IBM WebSphere Application Server Network Deployment

This section provides detailed instructions for installing, configuring, and verifying IBM WebSphere Application Server Network Deployment for AIX. The section is organized into the following tasks:

1. Preinstallation tasks.
2. Installing IBM WebSphere Application Server Network Deployment.
3. Verifying the installation.

11.4.3 Preinstallation tasks

Prior to installation you will need to make sure the IP ports required are free.

- 2809 (bootstrap port)
- 8879 (SOAP connector address for JMX)
- 9090 (administrative console)
- 9043 (secure administrative console)

Note: If you have installed WebSphere Application Server on this same machine, you should stop it before installation of IBM WebSphere Application Server Network Deployment.

We suggest using the following command for this task:

```
# netstat -an | grep LISTEN
```

11.4.4 Installing WebSphere Application Server Network Deployment

To install WebSphere Application Server Network Deployment using the GUI installer interface, complete the following steps:

1. Log in as root.
2. Start a terminal session.
3. Load the WebSphere Application Server Network Deployment CD-ROM into the CD-ROM drive and the CD will be automatically mounted by CDE.

Author Comment: need to check the CDROM packaging and names. Need to expand on steps 4 & 5. I think step 4 should be a drive on the CDROM?

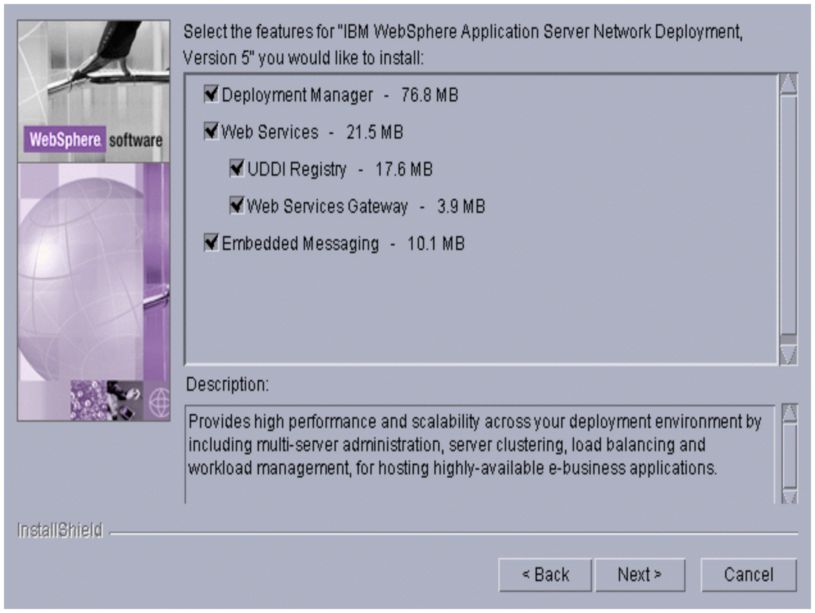
4. Change the directory to the installation root
5. Run the installation script:

   ```
   # ./LaunchPad.sh
   ```

6. In the Select a Language window, select a language and click **OK**. In this example, `English` is selected.
7. In the LaunchPad main window, click **Install the product**.
8. In the Installation Wizard window, select a language and click **OK**. In this example, `English` is selected.
9. In the Welcome window, click **Next**.
10. In the Software License Agreement window, read the license agreement, select **Accept**, and click **Next**.
11. In the Installation Options window, select **Custom** and click **Next**.

Note: You could select typical install here since we will be selecting everything. We just do this so you can see what components will be included in the install.

12. In the Select the features window, choose all options, and click **Next**.


Select the features for "IBM WebSphere Application Server Network Deployment, Version 5" you would like to install:

- ☑ Deployment Manager - 76.8 MB
- ☑ Web Services - 21.5 MB
  - ☑ UDDI Registry - 17.6 MB
  - ☑ Web Services Gateway - 3.9 MB
- ☑ Embedded Messaging - 10.1 MB

Description:

Provides high performance and scalability across your deployment environment by including multi-server administration, server clustering, load balancing and workload management, for hosting highly-available e-business applications.

Figure 11-12 Installation components selected (ND)

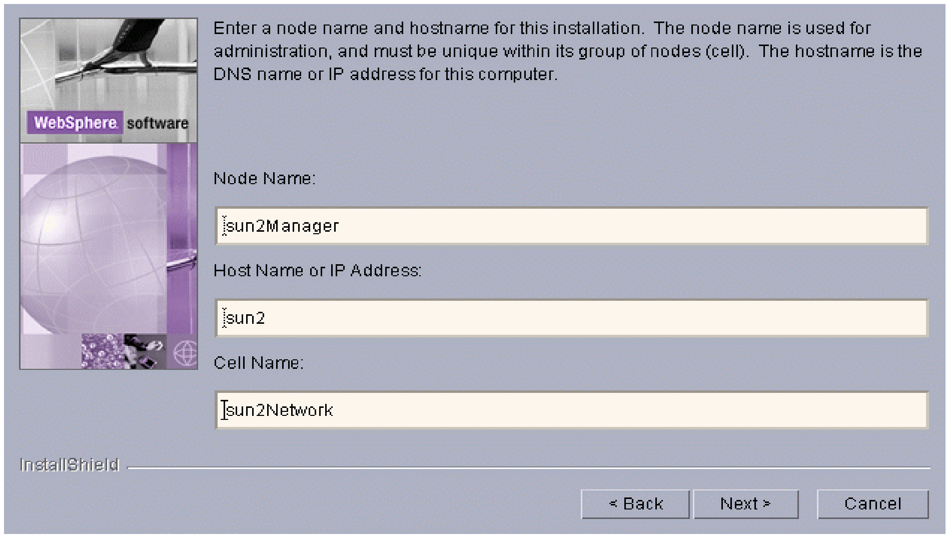
Important: Although not listed in the window, the IBM JDK 1.3.1 is automatically installed under the WebSphere Network Deployment installation directory. There is no need to separately install a JDK.

13. In the Install directory window, take the directory. We used the default:

    ```
    /opt/WebSphere/DeploymentManager
    ```

    Click **Next**:

14. In the next screen we take the default values for node name, host name, and cell name window.


Enter a node name and hostname for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer.

Node Name: sun2Manager

Host Name or IP Address: sun2

Cell Name: sun2Network

Figure 11-13 Node name, hostname and cell name window (ND)

Click **Next**.

15. In the confirmation window, confirm the selected features will be installed, and click **Next**.
16. In the registration window, select whether to register now. If you choose not to you can register later. Click **Next**.
17. In the Finish window, click **Finish**. The installation will take place and when done, will automatically start the First Steps program.

11.4.5 Verifying the installation

In order to very the installation of WebSphere Application Server Network Deployment, do the following:

1. Use the Installation Verification Test.
2. Check the installation log.
3. Start the deployment manager.

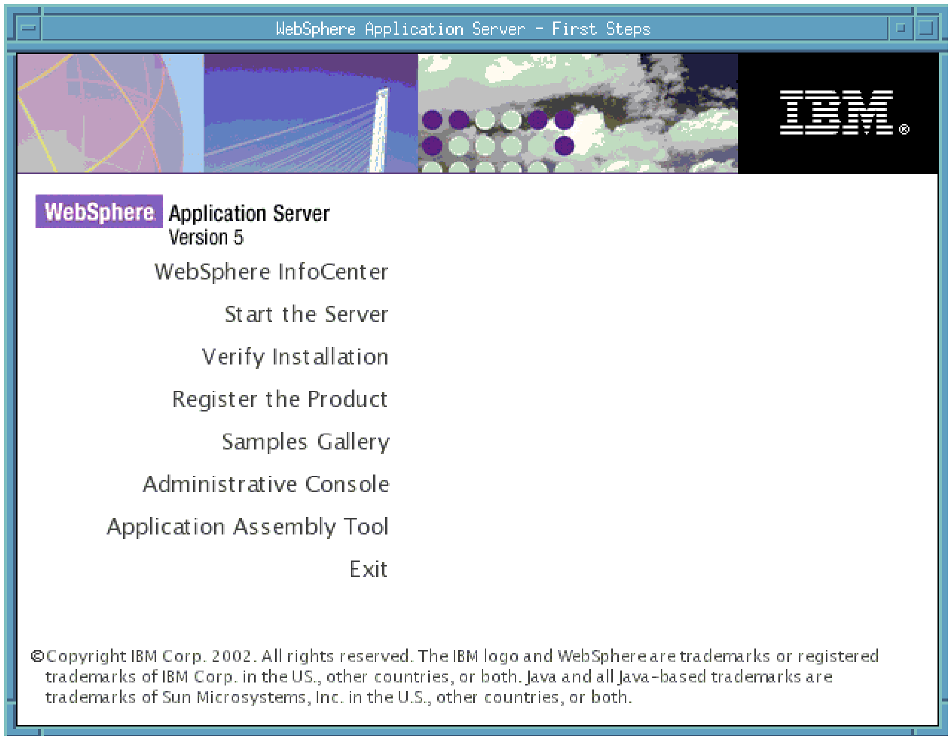
Installation Verification Test

The Installation Verification Test (IVT) is a tool which scans the product log files for errors and verifies core functionality of the product installation. It will attempt to start the deployment manager and verifies its status. The IVT can be started from the First Steps window (started after installation).

Note: If you don't have the First Steps window up:

```
cd /opt/WebSphere/AppServer/bin
./firststeps.sh
```

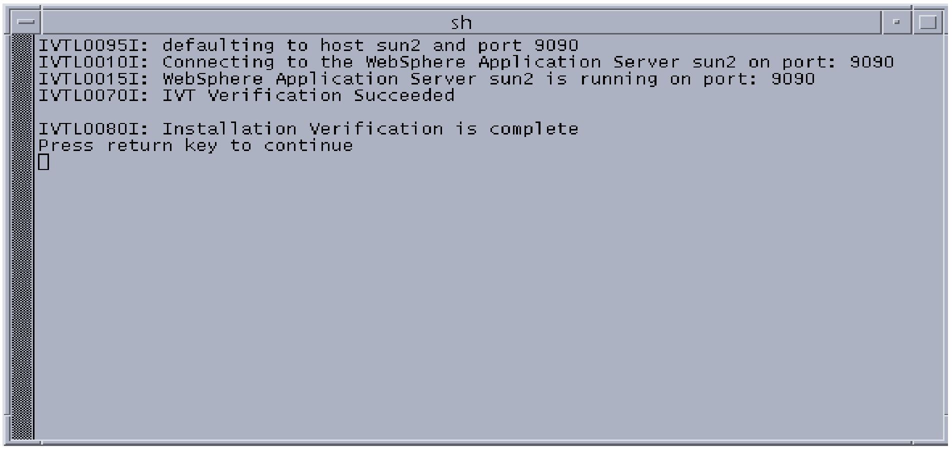
Select **Verify Installation** to start the IVT.



©Copyright IBM Corp. 2002. All rights reserved. The IBM logo and WebSphere are trademarks or registered trademarks of IBM Corp. in the US., other countries, or both. Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the U.S., other countries, or both.

Figure 11-14 First Steps Main Window

A shell is initiated and displays the verification test messages as it runs. The messages you should expect are shown in Figure 11-15.



```
IVTL0095I: defaulting to host sun2 and port 9090
IVTL0010I: Connecting to the WebSphere Application Server sun2 on port: 9090
IVTL0015I: WebSphere Application Server sun2 is running on port: 9090
IVTL0070I: IVT Verification Succeeded

IVTL0080I: Installation Verification is complete
Press return key to continue
```

Figure 11-15 Installation verification test messages

When the test is complete, press **Enter**.

Check the installation log

If you experience errors during the installation or verification, check the following installation log files under <WAS_ROOT>/logs directory for errors.

Table 11-6 Installation log files (ND)

Component	File
WebSphere Application Server Network Deployment	log.txt
Administrative console	installAdminConsole.log
File Transfer Application	installFiletransfer.log

Start the deployment manager

The IVT starts the deployment manager. You can also start the server in the following ways:

- Start from First Steps window click **Start the Server.**
- Or, from a terminal session:
  a. Log in as root on the WebSphere machine.
  b. Start a terminal session.
  c. Issue the following commands:

```
# cd <WAS_ND_HOME>/bin
# ./startManager.sh
```

Check the SystemOut.log and SystemErr.log log files under <WAS_ROOT>/logs/dmgr directory. The following message should be appeared in SystemOut.log for its successful start.

```
[9/13/02 14:06:18:513 EDT] 75c55974 WsServer      A WSVR0001I: Server dmgr
open for e-business
```

Open the administrative console

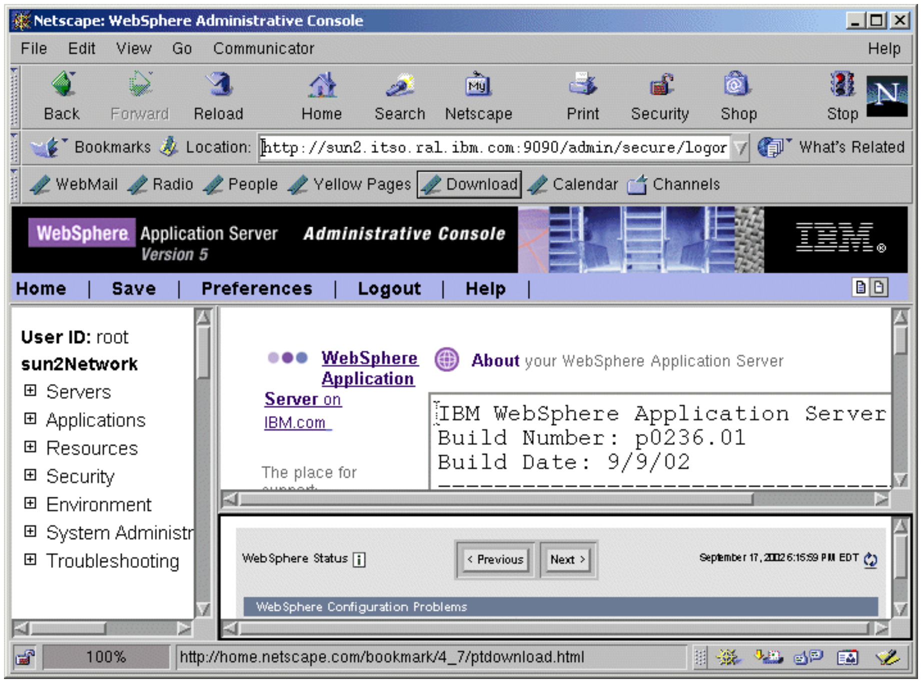
To access the administrative console:

1. Using a Web browser, request the following URL:

   ```
   http://<hostname>:9090/admin
   ```

   A window similar to the one shown in Figure 11-18 is displayed in the browser.

2. Enter a user ID for tracking purposes. Security has not been implemented yet so any name is acceptable. In this example, **root** is used as user ID. The following window will appear in the browser.


Netscape: WebSphere Administrative Console
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://sun2.itso.ral.ibm.com:9090/admin/secure/logor What's Related
WebMail Radio People Yellow Pages Download Calendar Channels
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: root
sun2Network
Servers
Applications
Resources
Security
Environment
System Administr
Troubleshooting
WebSphere Application Server on IBM.com
The place for support
About your WebSphere Application Server
WebSphere Status
< Previous
Next >
September 17, 2002 6:15:59 PM EDT
WebSphere Configuration Problems
100%
http://home.netscape.com/bookmark/4_7/ptdownload.html

```
IBM WebSphere Application Server
Build Number: p0236.01
Build Date: 9/9/02
```

Figure 11-16 Administrative console (ND)

The administrative console for Network Deployment looks similar to the base WebSphere Application Server. However, you will notice as you go through it that you have more configuration options.

11.4.6 Add a node to existing WebSphere cell

Now that you have a Network Deployment environment it makes sense to add a node to the cell. The cell gets configured while installing the WebSphere network Deployment. The steps for adding a node is independent of platform. You can follow the steps described in Section 10.7, "Add a WebSphere node into an existing cell" on page 259

11.5 Install the WebSphere HTTP plug-in on a remote node

Note: This section is to be used when you have the Web server and WebSphere installed on two separate machines. The WebSphere HTTP plug-in must be installed on the Web server machine.

This section describes how to install the WebSphere plug-in on the Web server machine. The assumption is that we have installed the Netscape iPlanet Web Server on a separate machine from WebSphere Application Server.

Preinstallation tasks

prior to installing the WebSphere plug-in onto the Web server machine the Web server has to be installed on the physical machines.

Install and configure 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 WebSphere Application Server CD-ROM into the CD-ROM drive and mount the CD.

Author Comment: check packaging and CDROM name.

4. Change the directory to the installation root.

Author Comment: cd to what? need to check installation media

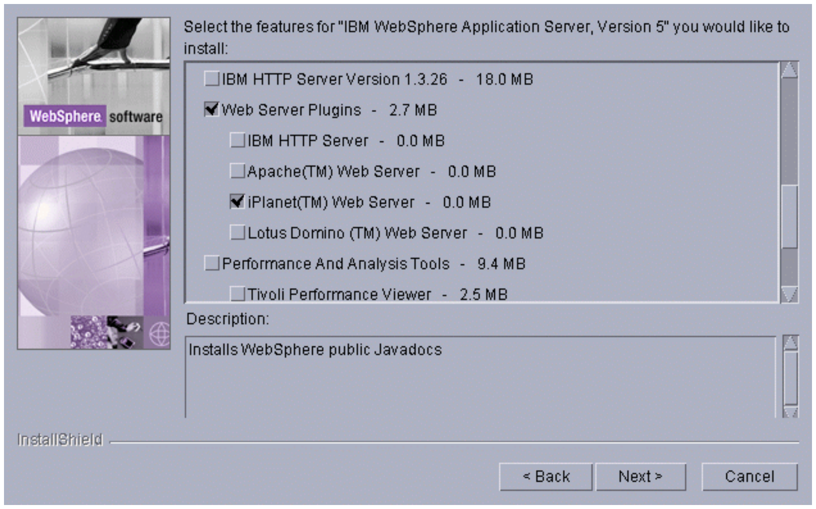
5. Ensure the DISPLAY and TERM environment variables are properly set.

Author Comment (Shafkat): please expand on this as you mentioned before. Apparently this is something that needs to be considered in a silent install.

6. Run the installation script:

```
# ./LaunchPad.sh
```

7. In the Select a Language window, select a language and click **OK**. In this example, `English` is selected.
8. In the LaunchPad main window, click **Install the product**.
9. In the Installation Wizard window, select a language and click **OK**. In this example, `English` is selected.
10. In the Welcome window, click **Next**.
11. In the Software License Agreement window, read the licence agreement, select **Accept**, and click **Next**.
12. In the Operating System Level Check window, Click **Next**.
13. In the Installation Options window, select **Custom** and click **Next**.
14. In the Select the features window, choose the iPlanet Web Server plug-in and click **Next**.


Select the features for "IBM WebSphere Application Server, Version 5" you would like to install:

- [ ] IBM HTTP Server Version 1.3.26 - 18.0 MB
- [✓] Web Server Plugins - 2.7 MB
  - [ ] IBM HTTP Server - 0.0 MB
  - [ ] Apache(TM) Web Server - 0.0 MB
  - [✓] iPlanet(TM) Web Server - 0.0 MB
  - [ ] Lotus Domino (TM) Web Server - 0.0 MB
- [ ] Performance And Analysis Tools - 9.4 MB
  - [ ] Tivoli Performance Viewer - 2.5 MB

Description:

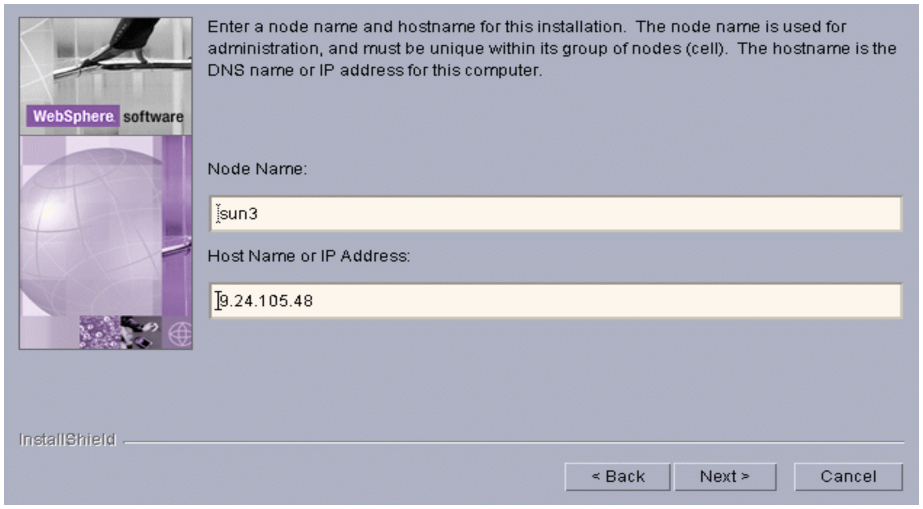
Installs WebSphere public Javadocs

Figure 11-17 Select the features window (plugin)

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.

15. In the Install directory window, the installation program provides the default location of WebSphere Application Server path `/opt/WebSphere/AppServer` automatically. Accept the default location and click **Next**:
16. In the next dialog you will be asked for the configuration file of the iPlanet Web server. Type the absolute path of the obj.conf file for your Web server instance. In our case the obj.conf file is located in:

    ```
    /opt/sunOne/server6/https-sun3.itso.ral.ibm.com/config directory
    ```
17. The next window asks for the node name and host name. Take the defaults and click **Next**.


Enter a node name and hostname for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer.

Node Name: sun3

Host Name or IP Address: 9.24.105.48

Figure 11-18 Node name and hostname window (plugin)

18. In the confirmation window, confirm the selected features will be installed, and click **Next**.
19. Proceed through the registration process or opt to register later.
20. In the Finish window, click **Finish** to begin the installation. When the installation is done, click **Exit** in the LaunchPad window.

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 1.3.1.

11.5.1 Verify the WebSphere plug-in installation

To check that required settings have been added to the Web server configuration follow the steps in "Check the Web server plug-in configuration changes" on page 368

11.5.2 Generating the 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. Regenerate the Web server plug-in settings.
2. Copy the plug-in settings to the Web server machine.
3. Restart the Web server.

Regenerate the Web server plug-in settings

To regenerate the Web server plug-in settings:

1. From a Web browser, access the administrative console using the following URL:

   ```
   http://<ND_websphere_hostname>:9090/admin
   ```
2. Select **Environment** in the left navigation screen.
3. Click on the **Update Web Server Virtual Host** to regenerate the plug-in file.
4. Click **OK.**

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. If you do not have a Network Deployment environment, copy the <WAS_BASE_HOME>/config/plugin-cfg.xml file from the WebSphere Application Server machine to the <WAS_BASE_HOME>/config directory on the remote Web server machine.

   If this is a Network Deployment environment, copy from the <WAS_ND_HOME>/config/plugin-cfg.xml file from the WebSphere Application Server Network Deployment to the Web server machine.

Important: WebSphere and the WebSphere HTTP plug-in can have different installation paths on the two servers. 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 Web server machine.
2. Start a terminal session.

3. Restart the Web server by issuing the following commands:

```
# cd <http_server_install_path>
# ./start
```

Note: In our example the http_server_install_path for the iPlanet netscape server is `/opt/sunOne/server6/https-sun3.itso.ral.ibm.com`.

11.5.3 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 log is specified by the `<Log>` element of the /opt/WebSphere/AppServer/config/plugin-cfg.xml configuration file. By default, this is set to the following:

   ```
   <Log LogLevel="Error" Name="<plugin_install_path>/logs/http_plugin.log"/>
   ```

   Using the defaults this would be:

   ```
   /opt/WebSphere/AppServer/logs/http_plugin.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:

Example 11-9 Excerpt of plug-in's log file

```
[Wed Sep 18 11:40:20 2002] 00000b7a 00000001 - PLUGIN: Plugins loaded.
[Wed Sep 18 11:40:20 2002] 00000b7a 00000001 - PLUGIN:
--------------------System Information-----------------------
[Wed Sep 18 11:40:20 2002] 00000b7a 00000001 - PLUGIN: Bld date: Sep  8 2002,
22:53:34
[Wed Sep 18 11:40:20 2002] 00000b7a 00000001 - PLUGIN: Webserver:
iPlanet-WebServer-Enterprise/6.0
[Wed Sep 18 11:40:20 2002] 00000b7a 00000001 - PLUGIN: Hostname = sun3
[Wed Sep 18 11:40:20 2002] 00000b7a 00000001 - PLUGIN: NOFILES = hard: 1024,
soft: 1024
[Wed Sep 18 11:40:20 2002] 00000b7a 00000001 - PLUGIN: MAX COREFILE SZ = hard:
INFINITE, soft: INFINITE
[Wed Sep 18 11:40:20 2002] 00000b7a 00000001 - PLUGIN: DATA = hard: INFINITE,
soft: INFINITE
```

```
[Wed Sep 18 11:40:20 2002] 00000b7a 00000001 - PLUGIN:
--------------------------------------------------------------#
```

Note: See Appendix C, “The plugin-cfg.xml file definitions” on page 1071 for more details.

Author Comment: The cross-reference to the plugin-cfg.xml should be updated later and pointed into the Web server configuration chapter.

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>/snoop
   ```

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 on page 340 will be obtained

11.6 Installing WebSphere in silent mode

This section describes how to install WebSphere Application Server using the non-interactive, or silent, mode. To complete a silent installation, you will create a customized response file from the default one, then execute the installation script, 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 do not have a previous version of WebSphere Application Server already installed on this machine. If you do have a previous version installed, do not follow these instructions.
- 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.

11.6.1 Default response file

A default response file, named responsefile.txt, is supplied with WebSphere Application Server. You can use this default response file as a template for creating a customized response file.

With default options, the following software and other resources are installed:

- IBM Java 2 Software Developer’s Kit (SDK) 1.3.1
- IBM HTTP Server 1.3.26
- IBM WebSphere Application Server
- Embedded Messaging
- Performance and Analysis Tools
- 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 when you use the default response file.

11.6.2 Customized response file

The default response file is used as a template for creating a customized response file. The default response file can be edited to select the components to install the install locations. Detailed comments within the default response file guide you through the installation and configuration options available for performing a silent installation.

The following lines are the example which lines should be updated at least for the basic installation with iPlanet Web server

```
-P wasBean.installLocation="/opt/WebSphere/AppServer"
-W nodeNameBean.nodeName="sun2"
-W nodeNameBean.hostName="sun2"
```

```
-W
defaultIHSConfigFileLocationBean.value="/opt/sunOne/server6/https-sun2.itso.ral
.ibm.com/config/obj.conf"
-P iplanet60PluginBean.active="true"
```

In this example, the following items are specified in the response file respectively.

- WebSphere installation directory
- node name
- host name or IP address
- configuration file of iPlanet Web server
- Enable the iPlanet plug-in installation

Note: If you decide to use IBM HTTP server, then you need to set the following options in the customized response file:

```
-P ihsFeatureBean.installLocation="/opt/IBMTTPServer"
-P ihsFeatureBean.active="true"
-P ihsPluginBean.active="true"
```

Important: All values should be enclosed in double quotes (““).

11.6.3 Perform 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.

   ```
   # <http_server_install_path/stop
   ```

   Or in our case:

   ```
   /opt/sunOne/server6/https-sun2.itso.ral.ibm.com/stop
   ```

   Note: If you plan to install IBM HTTP Server 1.3.26 as part of the WebSphere Application Server installation and you have a level of IBM HTTP Server prior to 1.3.26 on your system, you must uninstall it first.

3. Ensure that the DISPLAY and TERM environment variables set correctly.

Author Commen (Shafkat): please complete as you do the previous reference

4. Insert the WebSphere Application Server CD-ROM into the CD-ROM drive. The Solaris Volume Management daemon will automatically mount the CD under /cdrom/cdrom0.

Author Comment: check packaging and CDROM name

5. Navigate to the /cdrom/cdrom0/sun directory by entering the following command:

```
# cd /cdrom/cdrom0/sun
```

6. Create a copy of the default response file by using the **`cp`** command, as follows:

```
# cp responsefile.txt <new_responsefile.txt>
```

   In this command, `<new_responsefile.txt>` represents the full path name of the copy of the default response file you are creating (for example, `/tmp/my`_responsefile.txt).

7. To create a customized response file, perform the following steps:

   a. Use a text editor (vi) to open your copy of the default response file, <new_responsefile.txt>.

   b. Use the detailed comments throughout the file to help you select the appropriate options for your WebSphere Application Server installation (see 11.7.2, “Customized response file” on page 388).

   c. Save the changes that you have made to the customized response file.

8. 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_responsefile.txt> 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_responsefile.txt).

```
# ./install.sh -options <new_responsefile.txt>
```

9. After installation is completed, refer to the log file named log.txt located in the /tmp directory to determine if the silent installation was successful. A copy of this file also exists in the directory <WAS_ROOT>/logs.

10. Unmount the CD-ROM before removing it from the CD-ROM drive by using the **`umount`** command, as follows:

```
# umount /cdrom/cdrom0/sun
```

11. Ensure that the Web server is running or start it by entering the following commands:

```
# cd <http_server_install_path>/bin
# ./start
```

12. Start a Web browser and type the name of the host machine as the URL (http://<host_machine> where host_name is a hostname of WebSphere server machine). If you see the Welcome to the IBM HTTP Server Web page in Figure 11-4 on page 360, the server has been installed correctly.

11.7 Install WebSphere Application Server Network Deployment in silent mode

This section describes how to install WebSphere Application Server Network Deployment using the non-interactive, or silent, mode. To complete a silent installation, you will create a customized response file from the default one, then execute the installation script 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.

11.7.1 Default response file

A default response file, named responsefile.txt, is supplied with WebSphere Application Server Network Deployment. You can use this default response file as a template for creating a customized response file.

With default options, the following software and other resources are installed:

- IBM Java 2 Software Developer’s Kit (SDK) 1.3.1
- IBM WebSphere Application Server Network Deployment
- Embedded Messaging
- Web Services

11.7.2 Customized response file

The default response file is used as a template for creating a customized response file. The default response file can be edited to select the components to install and the installation location. Detailed comments within the default response file guide you through the installation and configuration options available for performing a silent installation.

The following lines are the example which lines should be updated at least for the basic installation.

```
-P wasBean.installLocation="/opt/WebSphere/DeploymentManager"
-W nodeNameBean.nodeName="sun2Manager"
-W nodeNameBean.cellName="sun2Network"
-W nodeNameBean.hostName="sun2"
```

In the example, the following items are specified in the response file respectively.

- WebSphere Network Deployment installation directory
- node name
- cell name
- host name or IP address

Important: All values should be enclosed in double quotes (““).

11.7.3 Perform a silent Installation

Perform the following steps to create a customized response file (if desired) and to install WebSphere Application Server Network Deployment. 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.

   ```
   # <http_server_install_path/stop
   ```

   In our case:

   ```
   /opt/sunOne/server6/https-sun2.itso.ral.ibm.com/stop
   ```

   Note: If you plan to install IBM HTTP Server 1.3.26 as part of the WebSphere Application Server installation and you have a level of IBM HTTP Server prior to 1.3.26 on your system, you must uninstall it first.

3. Ensure that the DISPLAY and TERM environment variables set correctly.

Author Comment (shafkat): here it is again

4. Insert the WebSphere Application Server Network Deployment CD-ROM into the CD-ROM drive. The Solaris Volume Management daemon will automatically mount the CD under /cdrom/cdrom0.

Author Comment: check packaging and CDROM names

5. Navigate to the /cdrom/cdrom0/sun directory by entering the following command:

   ```
   # cd /cdrom/cdrom0/sun
   ```

6. Navigate to the /mnt directory.
7. Ensure that you are in the install directory and create a copy of the default response file by using the **`cp`** command, as follows:

   ```
   # cp responsefile.txt <new_responsefile.txt>
   ```

   In this command, `<new_responsefile.txt>` represents the full path name of the copy of the default response file you are creating (for example, `/tmp/my`_responsefile.txt).
8. To create a customized response file, perform the following steps:
   a. Use a text editor to open your copy of the default response file, <new_responsefile.txt>.
   b. Use the detailed comments throughout the file to help you select the appropriate options for your installation (see 11.7.2, “Customized response file” on page 388).
   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_responsefile.txt> 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_responsefile.txt).

   ```
   # ./install.sh -options <new_responsefile.txt>
   ```

10. After installation is complete, refer to the log.txt log file located in the /tmp directory to determine if the silent installation was successful. A copy of this file also exists in the directory <WAS_ND_HOME>/logs.
11. Unmount the CD-ROM before removing it from the CD-ROM drive by using the **`umount`** command

11.8 Configure WebSphere HTTP transport for SSL

Author Comment: Move to security chapter when we get one

Each Web container transport 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 (Web container).
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. Exchange of the server and plug-in self-signed certificate as a trusted CA (certificate authority) between each other'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.

In this section we show the example of SSL configuration with client authentication disabled. To enable the client authentication see the note in “Configure WebSphere transport” on page 394.

11.8.1 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 running **`ikeyman.sh`** from the `<WAS_ROOT/bin` directory.
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: `WASWebContainer.jks`
   - Location: `<WAS_ROOT>/etc/`
5. In the Password Prompt window, enter the following, then click **OK** to continue
   - Password: password to protect keystore file contents
6. Close the keystore database file.

Create a new self-signed certificate for Web container

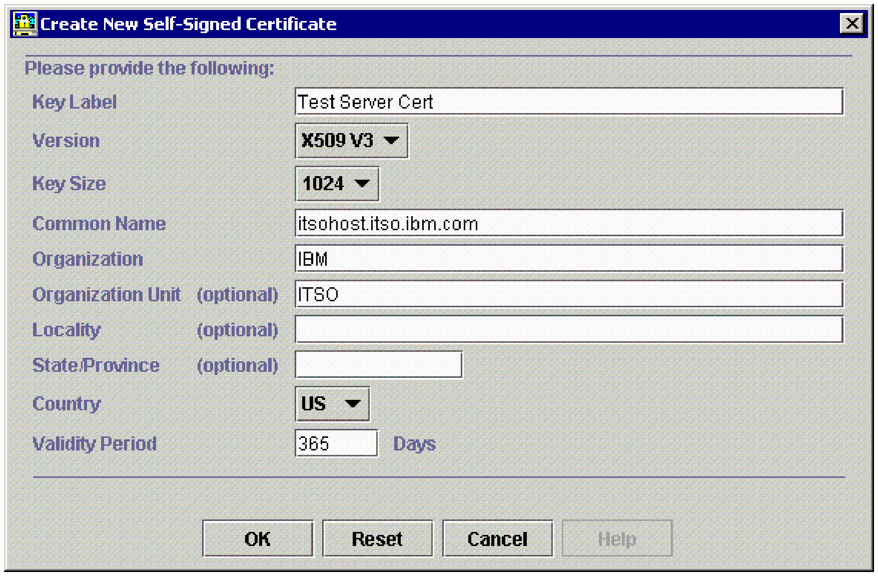
To create a new self-signed certificate, perform the following steps on the WebSphere server machine:

1. Log in as a local administrator.
2. Start the WebSphere IBM Key Management Utility from <WAS_ROOT>/bin directory by running **`ikeyman.sh`** script.
3. Select **Key Database File** from the menu bar, then select **Open**.
4. Specify the <WAS_ROOT/etc/WASWebContainer.jks file.
5. Provide the password.
6. 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.

7. Complete the required fields In the Create New Self-Signed Certificate window shown in Figure 11-19.

Author Comment (shafkat): I think the key label needs to be WASWebContainer in the following pic. Also need to change the common name to sun2.itso.ral.ibm.com. We had text and the graphic that didn't match. Just need to recapture the graphic.


Create New Self-Signed Certificate

Please provide the following:

Key Label	Test Server Cert
Version	X509 V3
Key Size	1024
Common Name	itsohost.itso.ibm.com
Organization	IBM
Organization Unit (optional)	ITSO
Locality (optional)	
State/Province (optional)	
Country	US
Validity Period	365 Days

Figure 11-19 Self-signed certificate settings

8. The certificate will be listed in the Personal Certificates pane of the utility.
9. Extract the public self-signed certificate key as it will be used later by the Web server plug-in peer to authenticate connections originating from the embedded HTTP Server in WebSphere.
10. Select **Personal Certificates** in the drop-down menu and select the WASWebContainer certificate that was just created.
11. Click the **Extract Certificate** button, ensuring that **WASWebContainer** remains selected. Extract the certificate to a file:

- Data Type: Base64-encoded ASCII data
- Certificate File Name: WASWebContainerPubCert.arm
- Location: <WAS_ROOT>/etc
  Click **OK** when you are finished

12. 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. Start the WebSphere administrative console
2. Select: **Security Center-> SSL Configuration Repertoires.**
3. Click on **New** to create a new entry in the repertoire. Provide the following values to fill out the form:
   - Alias: Web Container SSL
   - Key File Name: /opt/WebSphere/AppServer/etc/server.jks
   - Key File Password: password
   - Key File Format: JKS
   - Trust File Name: /opt/WebSphere/AppServer/etc/server.jks
   - Trust File Password: password
   - Trust File Format: JKS
   - Client Authentication: Unchecked

Note: If you want to use client authentication you have to check the Client Authentication box. The client authentication requires both client and server to exchange their public keys. In our example we have done it by default. If the client authentication is disabled you need the public key of the server to be stored on the client side (i.e. the plugin side keystore file)

   - Security Level: High
4. Save the configuration.
5. Select the **Servers->Application Servers**, then select the server you want to work with, in this case `Server1`.
6. Select the **Web Container** under the server.
7. Select **HTTP Transport** under the Web container.
8. Click **New** under the HTTP Transport and enter the appropriate values. In our example we use:
   - Host: *(asterisk)
   - Port: 9081

9. On the configuration panel, check-in the **Enable SSL** box.
10. Select the **Web Container SSL** entry from the repertoire in the SSL drop-down list.
11. Click **OK**, then save the configuration for WebSphere.

Create a self-signed certificate for the Web server plug-in

The Web server plug-in requires a keyring to store its own private and public keys and to store the public certificate from the Web container's key file.

To create a new plug-in keystore database file, complete the following steps on the **Web serve**r machine:

1. Log in as a local administrator.
2. Launch the IBM ikeyman tool that ships as part of GSKit and support the CMS key database format. This version of ikeyman tool comes with the IBM HTTP Server.

Important:

- The IBM HTTP Server provides an ikeyman utility that supports the KDB key file format
- WebSphere Application Server provides an ikeyman utility that support JKS key file format.
- Sun ONE Web Server does not provide an ikeyman utility.

In this example we are using the IBM HTTP Server ikeyman utility and have installed the IBM HTTP Server on both the WebSphere and Web server machines (even though we plan to use Sun ONE Web Server as our production Web server). This gives us the ikeyman utility on both machines and makes key file manipulation easier. There was no need to install the HTTP Server plug-in since we will not be using the IBM HTTP Server to access applications on WebSphere.

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: WASplugin.kdb
   – Location: <WAS_ROOT>/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:
<WAS_ROOT>/etc/client.sth
```

7. From the ikeyman menu select **Create-> New Self-Signed Certificate** to create a new self-signed certificate key pair. The following options then need to be specified: you may choose to complete all of the remaining fields for the sake of completeness:
   - Key Label: WASplugin
   - Version:X509 V3
   - Key Size:1024
   - Common Name:<Web_server_full_hostname>(sun1.itso.ral.ibm.com)
   - Organization: IBM
   - Country: US
   - Validity Period: 365

   Click **OK** when you are finished.
8. Extract the public self-signed certificate key, as this will be used later by the embedded HTTP Server peer to authenticate connections originating from the plug-in.
9. Select **Personal Certificates** in the drop-down menu and select the WASPlugin certificate that was just created.
10. Click the **Extract Certificate** button, ensuring that WASPlugin remains selected. Extract the certificate to a file :
    - Data Type: Base64-encoded ASCII data
    - Certificate File Name: WASpluginPubCert.arm
    - Location: <WAS_ROOT>/etc

Click **OK** when you are finished.

11. Close the keystore and exit the IBM Key Management Utility.

Exchanging public certificates

The following two sections will describe how to exchange certificates between the Web container keystore and the Web server plug-in keyfile. In order to import the certificates into the keystores as described in the next two sections, you will have to copy the two certificates (the extracted .arm files) to the proper directories for the Web server and for WebSphere.

- Copy the WASpluginPubCert.arm from the Web server machine to the WebSphere machine. The destination directory is: /opt/WebSphere /Appserver/etc.
- Copy the WASWebContainerPubCert.arm from the WebSphere machine to the Web server machine. The source directory in our case is /opt/WebSphere/AppServer/etc , while the destination is also : /opt/WebSphere/AppServer/etc on the Web server machine.

Importing the certificate into the Web server plug-in keyfile

On the Web server machine launch the ikeyman utility that supports the CMSkey database format.

```
cd /opt/IBMHttpServer/bin
ikeyman
```

- From the ikeyman menu select **Key Database File -> Open** and select the previously created key database file: **WASplugin.kdb.**
- At the password prompt window, enter the password then click **OK**.
- Select **Signer Certificates** from the drop-down list, then click the **Add** button.This will allow you to import the public certificate previously extracted from the embedded HTTP server/Web Container keystore.
  - Data type: Base64-encoded ASCII data
  - Certificate file name: WASWebContainerPubCert.arm
  - Location: /opt/WebSphere /Appserver /etc /

  Click **OK** when you are finished.

You will be prompted for a label name by which the trusted signer public certificate will be known. Enter a label for the certificate: **WASWebContaine**r. Close the key database and quit ikeyman when you are finished.

Importing the certificate into the Web container keystore

On the WebSphere machine, launch the IBM JKS capable ikeyman version that ships under the WebSphere bin directory.

```
cd /opt/WebSphere/AppServer/bin
./ikeyman.sh
```

- From the ikeyman menu, select: **Key Database File -> Open** and select the previously created **WASWebContainer.jks** file.
- At the password prompt, enter the password for the keyfile then click **OK**.
- Select **Signer Certificates** in the drop-down list and click on the **Add** button. This will allow you to import the public certificate previously extracted from the server plug-in keyfiles.
  - Data type: Base64-encoded ASCII data
  - Certificate file name: WASpluginPubCert.arm
  - Location: /opt/WebSphere /Appserver /etc /

  Click **OK** when you are finished.
- You will be prompted for a label name by which the trusted public certificate will be known. Enter a label for the certificate: **WASplugin**.

Modifying the Web server plug-in file

The plug-in configuration file must be modified to reference the plug-in keyring and the password stash file. This allows the transport protocol to be changed from HTTP to HTTPS, using the certificates stored in the keyring. A standard non-secure HTTP connection in the configuration looks like the following:

```
<Transport Hostname="wassrv01"Port="9081"Protocol="http"/>
```

The same entry, but secured, looks like the following:

```
<Transport Hostname="sun2"Port="9081"Protocol="https ">
<Property name="keyring"
value="c:\IBMHttpServer \conf \keys \WASplugin.kdb"/>
<Property name="stashfile"
value="c:\IBMHttpServer \conf \keys \WASplugin.sth"/>
</Transport>
```

It might be useful for production environment to replace the original plugin-key.kdb file with your own key file for the secure transport definition, port 9443.

Note: The Transport XML tag has a body and a closing tag. Make sure you remove the slash ‘/’ from the end of the opening tag

Testing the secure connection using embedded HTTP server

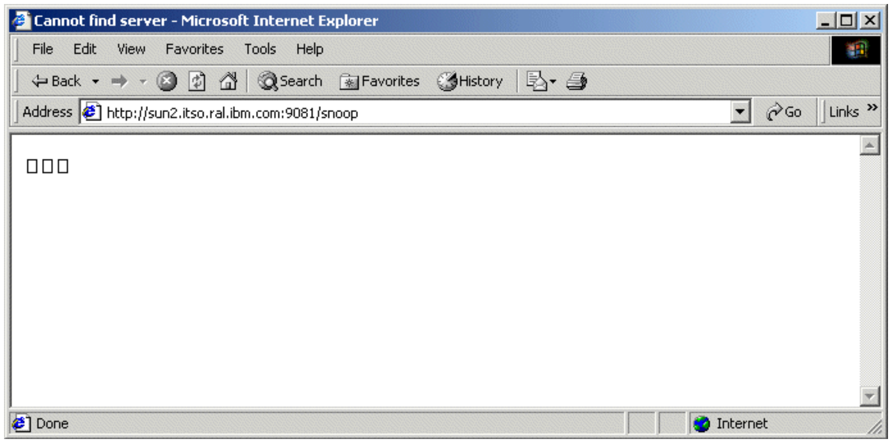
Prior to beginning the test the following tasks has to be completed:

- Configure port 9081 under default host so that https request can be made through this port
- Regenerate the plugin file. If the Web server is located on a separate physical machine, copy the plug-in file from the WebSphere machine to the Web server machine.
- Stop and start the application server and Web server to enable the configuration changes.

To test the secure connection use your favorite Web browser and access an application on WebSphere Application Server using the port 9080, for example:

```
https://wassrv01.itso.ibm.com:9080/itsobank .
```

Make sure you use the HTTPS protocol. If not you the page returned will look like Figure 11-20.


Cannot find server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://sun2.itso.ral.ibm.com:9081/snoop
Go Links
Done
Internet

□□□

Figure 11-20 Testing SSL configuration between plugin and Web Container

In order to test the secured connection when client side certification is required, the right certificate with public and private key has to be imported into the browser.

1. On the Web server machine, launch the ikeyman that can handle the CMS key database file.
2. Open the keyfile for the plugin. In our example this is /opt/WebSphere/AppServer/etc/WASPlugin.kdb. Provide the password when prompted.

3. Select the WASplugin certificate under the personal Certificates, then click on **Export**.
4. Save the certificate in PKCS12 format to a file, /opt/WebSphere/AppServer/etc/WASplugin.p12. Provide a password to secure the PKCS12 certificate file.
5. Select **Weak encryption (browser compatible)**.
6. Close the keyfile and quit ikeyman when you are done.
7. Copy the saved file to the client machine where you want to access the WebSphere server form.
8. Import the PKCS12 file into your browser.

   For example, in Microsoft Internet Explorer, select **Tools ->Internet options** from the menu. Switch to the Content tab and select **Certificates**. Import the WASplugin.p12 certificate by using the import button. Provide the password for the file where it is required. The new certificate should appear under the personal tab. Close the certificates and options dialog.
9. In the browser access the WebSphere application again, for example

   ```
   :https://sun2.itso.ral.ibm.com:9080/snoop
   ```

Note: Here we use the full host name of the WebSphere machine because the embedded server is located on the same machine.

The Web page should come up with the right content.

Testing the secure connection using the remote Web server

In this section we assume the Web server and WebSphere are installed on separate machines. We also assume that https communication is configured between the browser and the Sun ONE Web Server. The following tasks have to be completed for testing the secure connection:

- Make sure virtual host definition for default host has port 9081 defined
- Regenerate the plug-in file on the WebSphere Application Server machine
- Copy the plug-in file from the WebSphere machine to the Web server machine.
- Make sure the plug-in file is manually edited to add the key database and password stashed file information for port 9081 as shown in Example 11-10.

Example 11-10 Excerpt of plugin-cfg.xml file

```
<Transport Hostname="9.24.105.47" Port="9081" Protocol="https">
               <Property name="keyring"
value="/opt/WebSphere/AppServer/etc/WASPlugin.kdb"/>
```

```
                <Property name="stashfile"
value="/opt/WebSphere/AppServer/etc/WASPlugin.sth"/>
            </Transport>
```

- Manually edit the plugin-cfg.xml file to comment out the transport information related to port 9080, 9443, 9090 and 9043 as shown in Example 11-11. This is required to force the plug-in to use port 9081.

Example 11-11 Transport information for port 9080, 9443, 9090 and 9043

```
<Transport Hostname="9.24.105.47" Port="9080" Protocol="http"/>
            <Transport Hostname="9.24.105.47" Port="9443" Protocol="https">
                <Property name="keyring"
value="/opt/WebSphere/AppServer/etc/plugin-key.kdb"/>
                <Property name="stashfile"
value="/opt/WebSphere/AppServer/etc/plugin-key.sth"/>
            </Transport>
            <Transport Hostname="9.24.105.47" Port="9090" Protocol="http"/>
            <Transport Hostname="9.24.105.47" Port="9043" Protocol="https">
                <Property name="keyring"
value="/opt/WebSphere/AppServer/etc/plugin-key.kdb"/>
                <Property name="stashfile"
value="/opt/WebSphere/AppServer/etc/plugin-key.sth"/>
            </Transport>
```

- Stop the Sun ONE Web Server using the **`stop`** command from the <http_server_install_path> directory. In our example it is /opt/sunOne/server6/https-sun1-443.
- Start the Sun ONE Web Server using the **`start`** command from the <http_server_install_path> directory. In our example it is /opt/sunOne/server6/https-sun1-443.
- In the browser access the WebSphere application again, for example: https://sun2.itso.ral.ibm.com:9080/snoop
- The Web page should come up with the right content.

Part 3

Configuring WebSphere

Note to Author: Optionally, describe the book part here. If you are not using Part files, you need to restart the page numbering in the first chapter file of your book:

- In FrameMaker 5.5: **Open .book > select first chapter> File > Set Up File > Page Numbering: > Restart at 1 > Set**
  - Now delete the three Part files (p01, p02 and p03) from your book: **Open .book > select a file > File > Rearrange Files > Delete > Done**
- In FrameMaker 6.0: **Open .book > select first chapter> Format > Document > Numbering > Page “tab” > select First Page # radio button and set Page # to 1 > Set**
  - Now delete the three Part files (p01, p02 and p03) from your book: **Open .book > select a part file > Edit > Delete File from Book**

In this part we introduce/provide/describe/discuss...

12

System Management

This chapter describes in detail the System Management functionality of IBM WebSphere Application Server Network Deployment. We will cover:

- Key features of IBM WebSphere Application Server system management
- System management topology
- Distributed administration
- Configuration and application repositories
- Application management
- Java Management Extensions (JMX)
- System management tools
- Common system management tasks

For detailed information regarding the configuration of specific components administered using System Management, see Chapter 13, “WebSphere administration basics” on page 449.

12.1 Key features

The system management functionality of IBM WebSphere Application Server Network Deployment has a number of key features:

- Application servers have less reliance on a central repository or administration server for basic functions and bringup.
- No administrative repository database is required. Instead, the configuration is stored in XML format files.
- Administration client programs are used to modify configuration settings:
  - Web-based administration (Web administration console).
  - WebSphere scripting (wsadmin).
- Each managed process, node agent and deployment manager starts with its own configuration file.

Note: Compare this with WebSphere Application Server 4.0, where the single *admin.config* configuration file was used to start all managed processes.

- WebSphere managed processes are organized into groups. All managed processes configured and running on the same node must enroll with the local node agent. All node agent processes configured in the same cell must enroll with the deployment manager.
- The use of Java Management Extensions (JMX) provides a number of benefits:
  - Management of WebSphere over multiple protocols. For example:
    - SOAP, RMI/IIOP (current)
    - HTTP, JMS, SNMP (future)
  - Easy to add or extend administrative functions by adding custom management beans (MBeans).
    - MBeans are defined in an XML file.
    - Allows developers to use JMX to manage their own applications.
  - The management interface is open and easy to integrate with third party management tools.
- Resources within WebSphere managed processes are represented by individual management interfaces implemented as JMX MBeans. These managed resources are accessible from other processes through one of the WebSphere AdminService proxies.

Note: 3rd party and custom applications can also implement their management interfaces as JMX MBeans, which can be deployed and executed in the application server runtime.

12.2 Topology

The components of the IBM WebSphere Application Server Network Deployment System Management topology are summarized in Figure 12-1.

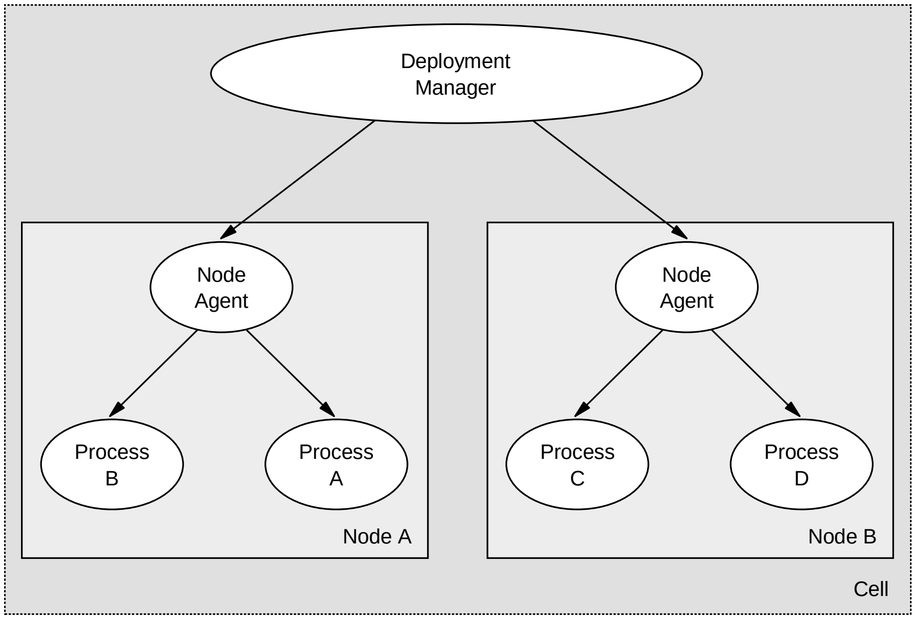


Figure 12-1 System Management topology

12.2.1 Definitions

The major new terms used include:

- Cell

  A cell is an aggregation of nodes. A deployment manager controls and communicates with all node agents that are part of the cell.

- Node

  A node is a set of managed servers on a physical machine in a topology composed of one or more machines. A node contains a IBM WebSphere Application Server installation and is managed by a node agent process on a single machine. A node cannot span machines, but a single machine can host multiple nodes.

- Deployment manager

  The process responsible for the management and control of the cell configuration and application data repository.

- Node Agent

  The process responsible for controlling the managed processes of a node.

- Managed process

  A managed process is a single IBM WebSphere Application Server process instance, running in its own JVM. All operating system processes that are components of the WebSphere product are called managed processes. This means that the processes all participate in the administrative domain. WebSphere JMX support is embedded in all managed processes and these processes are available to receive administration commands and to output administration information about the state of the managed resources within the processes.

12.2.2 Operation

IBM WebSphere Application Server Network Deployment can be installed on any machine in the network to create a deployment manager cell. It does not require the base IBM WebSphere Application Server to be installed on the same machine, although that is certainly possible.

The **`addNode`** command can be run (on the base node) to add a base instance (node) to the deployment manager cell. The deployment manager will create configuration files for each node that has been added to its cell and assumes responsibility for the configuration of all servers on the node.

The successful operation of a Network Deployment topology requires that each process run a specific set of system management services. In addition, some applications are used specifically to support the system management function, and therefore must be installed on specific processes.

12.3 Distributed administration

IBM WebSphere Application Server Network Deployment allows multiple IBM WebSphere Application Server nodes to be managed from a single central location. Distributed administration requires IBM WebSphere Application Server Network Deployment to be installed on a machine in the environment.

12.3.1 Administration Layers

The distributed administration of components is brought about by three tiers (layers) of administration services, as shown in Figure 12-2.

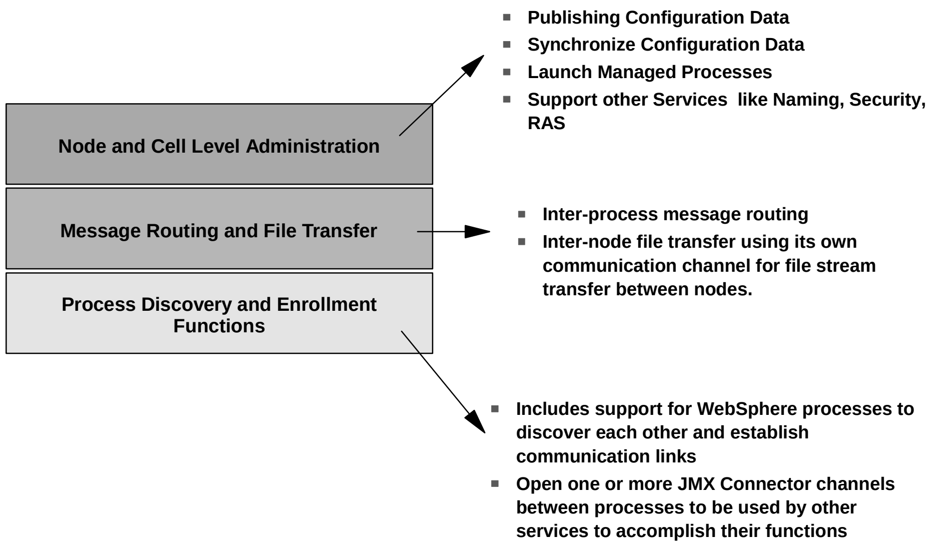

Node and Cell Level Administration
Message Routing and File Transfer
Process Discovery and Enrollment Functions
Publishing Configuration Data
Synchronize Configuration Data
Launch Managed Processes
Support other Services like Naming, Security, RAS
Inter-process message routing
Inter-node file transfer using its own communication channel for file stream transfer between nodes.
Includes support for WebSphere processes to discover each other and establish communication links
Open one or more JMX Connector channels between processes to be used by other services to accomplish their functions

Figure 12-2 Layers of distributed administration services

Between these tiers, communication is used to distribute configuration and application data updates from the deployment manager to the node agents, and in turn to the server instances.

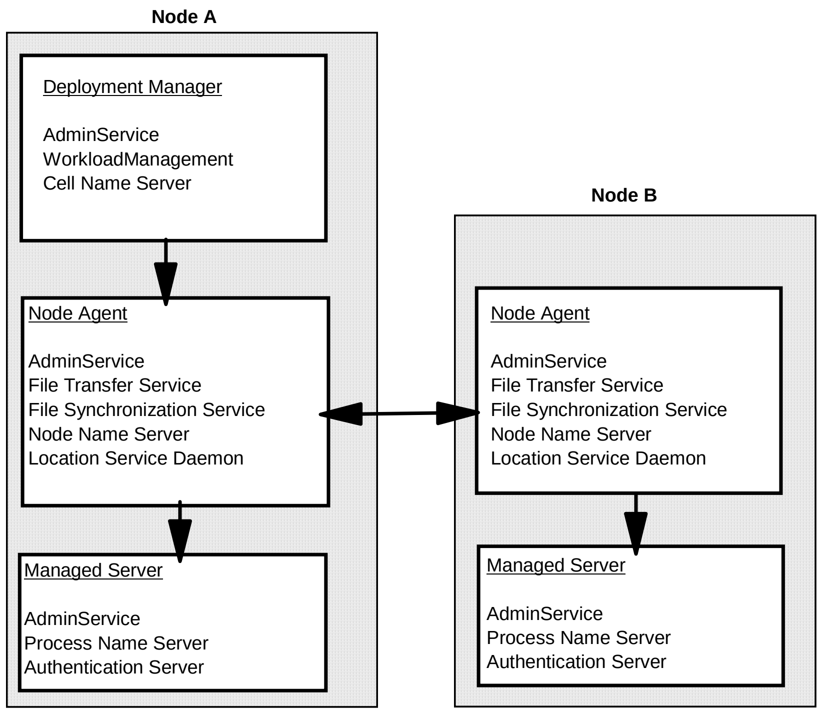


Figure 12-3 Distributed administration communication flow

The routing of administration messages between components makes use of the JMX ObjectName that identifies the target managed resource within the administrative cell. The ObjectName contains all of the information necessary to route a request targeted at the resource, to the appropriate node where the resource is executing.

An example is shown in Figure 12-4, where an operation on node Y invokes a management method on a management bean (MBean) located on another node (X). The message is forwarded from the AdminService on the local node (Y), to the AdminService on the node agent on the remote node (X). The node agent forwards the message to the application server’s AdminService, which invokes the MBean. For further details on management beans, see Section 12.6, “System management tools” on page 435.

.

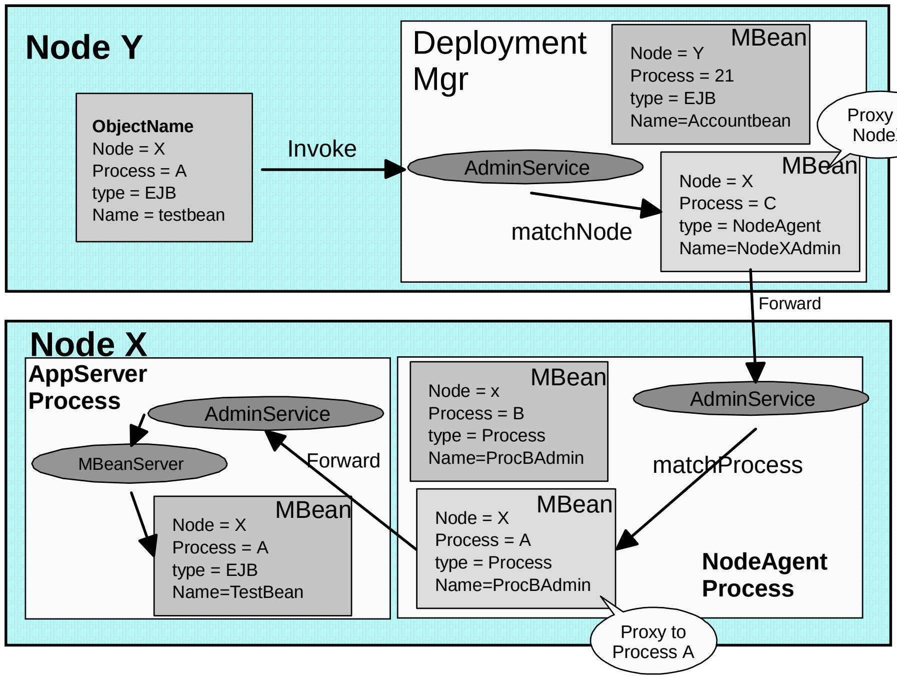


Figure 12-4 Distributed administration message routing

Author Comment: I think it would be worth walking thru this pic with numbered steps.

12.3.2 Administration points within a cell

Administration of configuration or application data can be performed at a number of points in the cell topology:

- Deployment manager

  Manage everything under the cell. *This is the recommended approach*.

- Node agent

  Manages everything under the specific node. Other cell configuration cannot be managed from this point.

- Managed processes

  Manages the server process configuration, but not the node or cell.

Note: The following issues should be borne in mind:

- The scripting administration client (**`wsadmin`**) can connect to the built-in AdminService of any deployment manager, node agent or managed process in a cell.
- The Web administration console can only connect to a server that has the adminconsole.ear application installed. In WAS-base, this is the default application server (server1). In WAS-ND, this is the deployment manager process. Node agents and managed processes in WAS-ND do not have the Web administration console client application installed by default.
- Any changes made at the node or server level are only local and therefore will not be reflected in the cell’s master repository. Local changes will be overridden when the next file synchronization (update) is performed for the node.
- Only changes made using the deployment manager (cell level) will be permanent for a node that is part of a cell.

12.3.3 Role based administration

WebSphere 4.0 did not provide any access control granularity for its administrative subsystems. There was only one role *Admin*, so that anyone with the user name and password could perform all administrative functions.

In contrast, IBM WebSphere Application Server provides finer granularity of access control, through the provision of four administrative roles:

- Monitor

  Can view the system state and configuration data, but cannot make any changes.
- Operator

  Has all the functions of *Monitor* as well as ability to make operational changes, e.g.. start / stop.
- Configurator

  Has all the functions of *Monitor* as well as ability to make configurational changes.
- Administrator

  Has all the functions of *Operator* and *Configurator*.

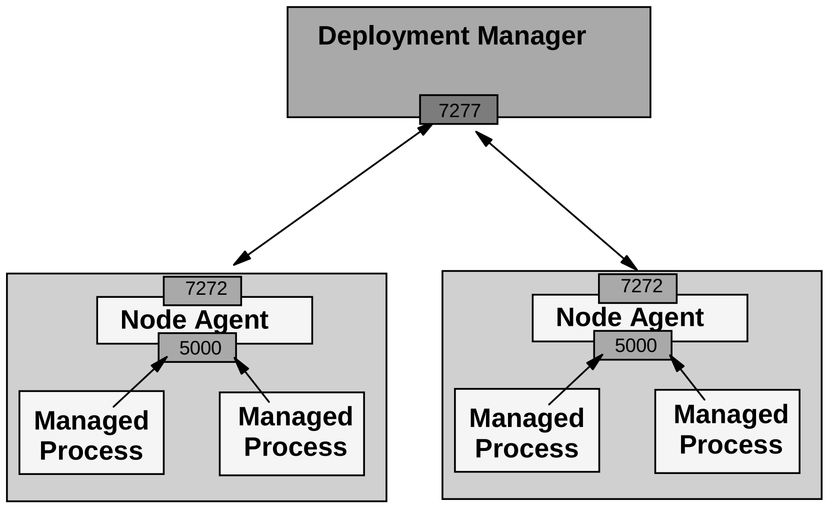
12.3.4 Common administration functions

The distributed administration of WAS-ND is built upon the following common functions:

- Distributed process discovery
- Centralized changes to configuration data
- File synchronization
- Configuration file support
- Launching processes

Distributed process discovery

When a managed process begins its startup, it sends a discovery request message that allows other processes to discover its existence and establish communication channels with the process. Figure 12-5 shows an overview of the discovery process.



```
serverindex.xml
serverType="Deployment_Manager"
    ... CELL_DISCOVERY_ADDRESS .. port:7277

serverType="Node_Agent"
    .... NODE_DISCOVERY_ADDRESS.. port 7272
    .... NODE_MULTICAST_DISCOVERY_ADDRESS .. port 5000
```

Figure 12-5 Distributed discovery process

The key features of process discovery in the distributed environment include:

- On startup, processes (deployment manager, node agent and managed processes) discover other running components and create communication channels between them.
- Each server has its own configuration and application data necessary for startup of the runtime and the installed applications.
- The order of process startup needs to adhere to the following rules:
  - A node agent can be running while the deployment manager is not, and vice versa.
  - The deployment manager can be running while a managed process is not, and vice versa.
  - Application servers can only be started if the local node agent is already started. The node agent contains the Location Service Daemon (LSD) in which each application registers itself on startup.

Discovery ports

Each node agent and deployment manager maintain status and configuration information by using discovery addresses.

- **Cell discovery address**

  Defines the TCP/IP port listened to by the deployment manager for discovery requests originating from node agents.

  The cell discovery address is defined under the DEPLOYMENT_MANAGER stanza in the *serverindex.xml* file of the deployment manager node:

  ```
  <WAS_ROOT>/config/cells/<cellname>/nodes/<depmgr_hostname>Manager/serverind
  ex.xml
  ```

  An example is shown in Example 12-1.

Example 12-1 CELL_DISCOVERY_ADDRESS definition

```
<serverEntries xmi:id="ServerEntry_1" serverDisplayName="dmgr"
serverName="dmgr" serverType="DEPLOYMENT_MANAGER">
...
<specialEndpoints xmi:id="NamedEndPoint_1"
endPointName="CELL_DISCOVERY_ADDRESS">
        <endPoint xmi:id="EndPoint_1" host="mka0kkcf" port="7277"/>
</specialEndpoints>
...
</serverEntries>
```

- **Node discovery address**

Defines the TCP/IP port listened to by a node agent for discovery requests originating from the cell deployment manager.

The node discovery address is defined under the NODE_AGENT stanza in the *serverindex.xml* file of the node:

```
<WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml
```

An example is shown in Example 12-2.

Example 12-2 NODE_DISCOVERY_ADDRESS definition

```
<serverEntries xmi:id="ServerEntry_2" serverDisplayName="Net1_JH"
serverName="Net1_JH" serverType="NODE_AGENT">
...
<specialEndpoints xmi:id="NamedEndPoint_1"
endPointName="NODE_DISCOVERY_ADDRESS">
        <endPoint xmi:id="EndPoint_1" host="mka0kkcf" port="7272"/>
</specialEndpoints>
...
</serverEntries>
```

- **Node multicast discovery address**

  Defines the multicast TCP/IP port listened to by a node agent for discovery requests originating from its managed processes (application servers and jmsserver).

  Multicast protocol is not supported for the deployment manager. The "node multicast port" is actually listened to by all of the managed processes on the node (not the node agent). That is why it is multicast - so one port can serve all processes on the host, rather than require yet another port to be used.

  A multicast address is used to prevent the usage of a large number of IP ports for managed process -> node agent discovery requests. Using multicast, a node agent can listen on a single IP port for any number of local servers.

  The node multicast discovery address is defined under the NODE_AGENT stanza in the *serverindex.xml* file of the node:

  ```
  <WAS_ROOT>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml
  ```

  An example is shown in Example 12-2.

Example 12-3 NODE_MULTICAST_DISCOVERY_ADDRESS definition

```
<serverEntries xmi:id="ServerEntry_2" serverDisplayName="Net1_JH"
serverName="Net1_JH" serverType="NODE_AGENT">
...
<specialEndpoints xmi:id="NamedEndPoint_1"
endPointName="NODE_MULTICAST_DISCOVERY_ADDRESS">
        <endPoint xmi:id="EndPoint_1" host="232.133.104.73" port="5000"/>
</specialEndpoints>
```

```
...
</serverEntries>
```

Important:

- The discovery service uses the *InetAddress.getLocalHost()* call to retrieve the IP address for the local machine's hostname. The network configuration of each machine must be configured so that *getLocalHost()* does not return the loopback address (127.0.0.1). It must return the actual (real) IP address of the correctly chosen NIC.
- A multicast address is a logical address, therefore it is not bound to an actual physical network interface, and will not be the same as the hostname (or IP address) of the host on which the node agent is executed.
- Multicast host addresses must be within a special range (224.0.0.0 to 239.255.255.255) defined by the IP standards and must never be a hostname value. The default for WebSphere 5.0 node agents is 232.133.104.73.

Examples

1. Situation: The node agent is not running and the deployment manager starts:
   a. The deployment manager tries to determine if the node agent is running. No luck.
   b. When the node agent is started, it contacts the deployment manager, creates a communication channel, and synchronizes data.
2. Situation: The node agent starts but no managed processes are started
   a. The node agent knows all about its managed processes and checks whether they are started. If so, it creates communication channels to these processes.
   b. When a managed process starts it checks whether the node agent is started and then creates a communication channel to it.

Centralized changes to configuration and application data

The deployment manager maintains a centralized repository of configuration data for the cell. Admin client programs (wsadmin and Web administration console) are used to modify configuration settings in this repository.

Modifications must be published (saved) to the repository before they are available for use by the IBM WebSphere Application Server system. This publish operation of the deployment manager results in changes being committed to the centralized repository and published out to the nodes of the cell. The files published to each of the nodes can include:

- All configuration documents of the cell.
- Only those configuration documents that apply to a particular node.
- Only those configuration documents that apply to a particular managed server on a particular node.

File synchronization

The configuration synchronization service is the administrative service responsible for keeping the configuration documents that are distributed across the WebSphere Application Server cell up-to-date. The service runs in the deployment manager and node agents, and ensures that the configuration changes made to the cell repository will be propagated out to the node repositories.

Configuration synchronization occurs in the following cases:

- The node agent starts.
- The node agent periodically requests any changes from the deployment manager.
- The end user forces synchronization.
- The node agent's *Auto Synchronization* flag is set to ON.

During the synchronization operation, a node agent checks with the deployment manager to see if any configuration documents that apply to the node have been updated. New or updated documents are copied to the node repository and deleted documents are then deleted from the node repository.

Synchronization scheduling

The occurrence of file synchronization is configured using an admin client. The available alternatives are:

- Automatic synchronization

  Synchronization can be made to operate automatically by configuring the *File Synchronization Service* of the node agent. The alternatives are:

  – Periodically. The time interval can be set via an admin client.

  – When the node agent starts and the deployment manager is already running.

  – When the deployment manager starts and the node agent(s) is already running.

> **Note:** By default, automatic synchronization is *enabled*, with an interval of 60 seconds.

- Explicit / forced synchronization

  Synchronization can be explicitly forced at anytime via use of an admin client (**`wsadmin`**, Web administration console, or **`syncNode`** command-line tool).

- Startup synchronization

  If startup Synchronization is enabled, by configuring the *File Synchronization Service* of the node agent, then an application server's configuration files are synchronized as part of the application server's startup process.

> **Tip:** In a production environment, the automatic synchronization interval should be increased so that the processing and network overhead is reduced.

Synchronization process

The configuration data is synchronized out from the deployment manager to each of the node agents (and in turn, managed processes) in the cell, as depicted in Figure 12-6.

1. Admin client programs are used to modify the configuration settings. Whether or not the master repository is updated depends upon at which point in the configuration hierarchy the change was made. See Section 12.4, "Configuration and application repositories" on page 422 for further details on the available administration points.
2. Individual nodes and servers synchronize the configuration data with the master repository.
3. Any changes made at the cell level are permanent.
4. Configuration changes made at the node agent or managed process level are temporary. At the next data update time, the cell's master configuration data is pushed down to the nodes (and therefore, managed processes).
5. The deployment manager checks-in/checks-out the configuration changes made to the master repository.

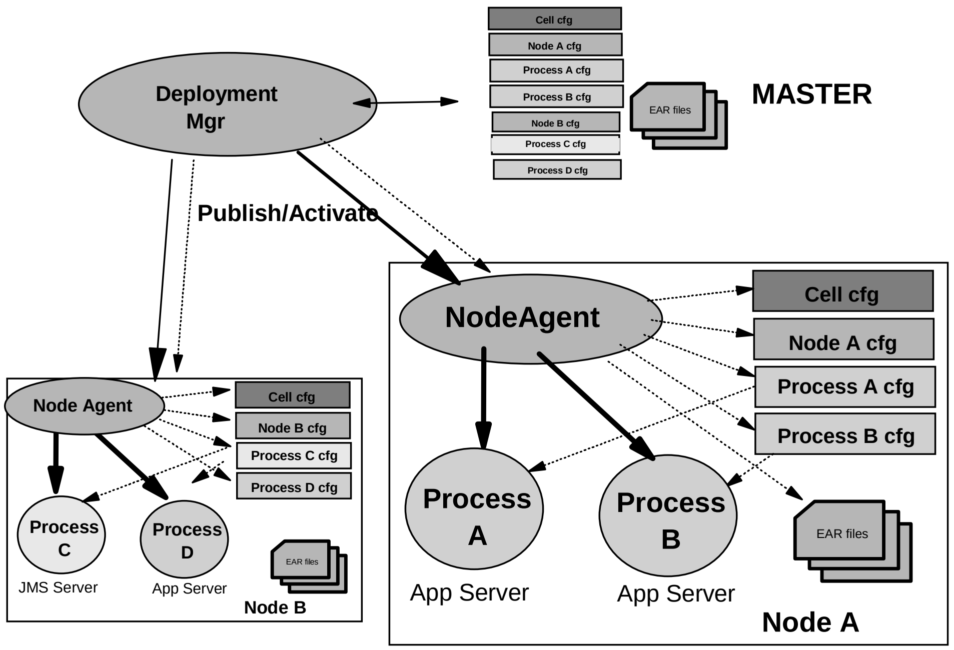


Figure 12-6 Configuration data synchronization

Each node will contain a separate instance (copy) of the repository. Each instance holds the documents pertaining to resources on that node, e.g. application server configuration documents for the application servers on that node.

Notes:

- The node with the deployment manager installed also maintains a centralized repository of configuration data for the cell.
- The cell configuration document (cell.xml) must be published to every node in the cell, and must be kept in synch.
- However, failures can occur in the system at times that prevent correct publishing of changes in the centralized repository out to the appropriate node(s). For example, the node agent may not be running, in which case it is unable to provide the *File Transfer Service* required to support the publish operation.

  Node agents use a *File Synchronization Servic*e that checks for differences between pairs of configuration documents (one in the node's local repository, the other in the cell's centralized repository) and provides a degree of automatic synchronization between the node and deployment manager based on a heuristic set of rules that decides which file contains the master configuration values.

Configuration file support

Each managed process has the data (configuration and application) necessary to start itself. The configuration data is as XML files and the application data is as EARs.

IBM WebSphere Application Server provides two administration clients that are to be used to edit / change the configuration of all components:

- Web administration console
- wsadmin scripting client

Configuration changes made using one of these client programs will be checked by built-in validator code. Invalid configurations produce a warning so corrective action can be taken. .

Tip: Although the XML configuration files could be edited by hand, this method is strongly discouraged. Manual changes do not undergo the validation process until the server runtime attempts to run using the new configuration. In this case, any config errors produce a warning message, and could result in the server runtime process failing to execute

For further information on these system management tools, see Section 12.6, "System management tools" on page 435.

Launching processes

- Managed processes are launched using the `startServer` command-line tool (see Section 12.6.2, “Command-line operational tools” on page 437).
- The node agent or deployment manager can start a managed process.

> **Tip:** Each managed process has a configuration setting which determines whether it should be started automatically when the node agent is started.

12.3.5 Required administration services

The distributed administration provides support for the services hosted by managed processes. JMX Management Bean (MBean) wrappers are used to expose the processes management services. The services hosted by each of the WAS-ND components is detailed in Table 12-1.

Table 12-1 Distributed administration services Vs. WebSphere process

Process	Services
Deployment manager	AdminService WorkloadManagement Cell Name Server
Node agent	AdminService File Transfer Service File Synchronization Service Node Name Server Location Service Daemon
Application Server	AdminService Process Name Server Authentication Server
JMS Server	AdminService Authentication Server

All processes host an AdminService (JMX agent) to support distributed administration. The JMX agent exposes the managed resources within the process for access and control from outside the process.

Also notice that all processes (except the JMS Server) host a name server, used for the local JNDI namespace.

Author Comment: Is the comment about the name server relevant here?

12.3.6 Required administrative applications

The operation of a WAS-ND environment requires that a number of applications be installed on the deployment manager in order to provide support for distributed administration.

Web administration console (adminconsole.ear)

The Web administration console is provided as a standard J2EE 1.3 compliant Enterprise Application Archive (EAR).

Note: By default, the node agent process is not configured to run a Web container. As such, the adminconsole application cannot be installed on any node agent process.

File Transfer Service (filetransfer.ear)

The deployment manager’s File Transfer Service is invoked by node agents to transfer new and updated files from the cell configuration repository to a node’s local repository. It is provided as a standard J2EE 1.3 compliant Enterprise Application Archive (EAR).

12.4 Configuration and application repositories

Author Comment: we need to firm up the terms used for repository. So far I have seen the following repository types: master, configuration, configuration data, application data, cell, node. Is each of these really a repository? Or a piece of a repository?

The configuration repository is a collection of documents containing everything WebSphere needs to know about the configuration. Most configuration files are in XML format. There are multiple repositories in a WAS-ND environment.

- The master configuration repository is held on the node containing the deployment manager. It contains all master copies of the configuration files for all nodes and servers in the cell.
- Each node and server also has its own node repository with local configuration files. Changes made to a local node or server file are temporary if the node belongs to a cell. While in effect, local changes override cell configuration. Changes at the cell level to node and server files are

permanent. Synchronization occurs at designated intervals or events, such as when the server starts.

Important: The cell repository is considered the master repository. Configuration changes made to node repositories are not propagated up to the cell.

12.4.1 Configuration data repository

The configuration files are arranged in a set of cascading directories under <WAS_ROOT>/config.

The cell repository hierarchy consists of four tiers of directories, with each directory containing several documents relating to different parts of the system:

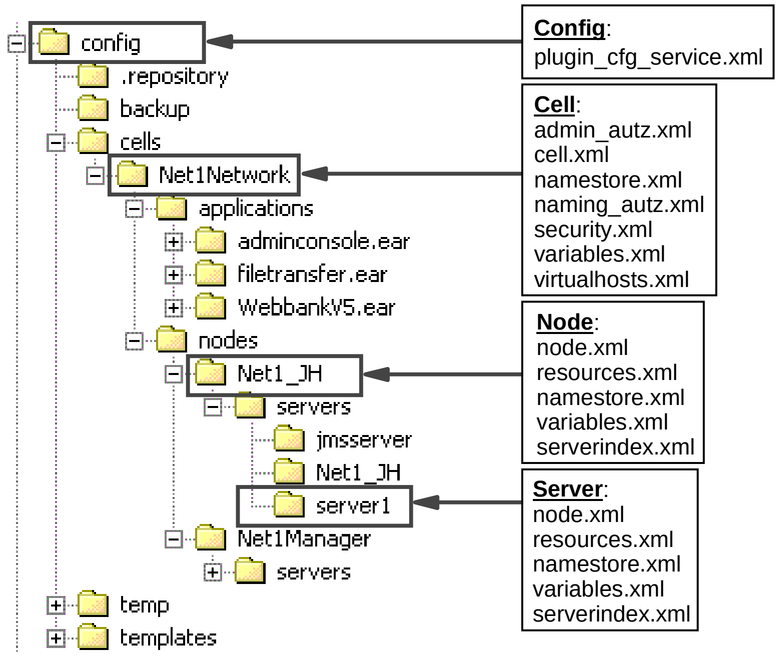


Figure 12-7 Cell configuration repository directories

Author Comment: Check this structure closer to GA . The pic doesn't show the cluster directory .. don't know if that is necessary but consider it..

As shown in Figure 12-7, the <WAS_ROOT>/config directory is the root of the configuration repository, containing the following directory structure:

- cells/<depmgr nodename>Network/

  The root level of configuration for the cell. Note that in this example, the deployment manager node name is Net1, so the cell directory is called Net1Network. There are two subdirectories under the cell directory:

  – An *applications* directory for application deployment data. The applications directory contains one subdirectory for every application that has been deployed within the cell. The directory name in this case must match the deployed application name.

  – A *nodes* directory (see below).

  The cell directory also contains a number of cell-level configuration settings files.

  > **Note:** The name of the cell in a IBM WebSphere Application Server Network Deployment configuration is *<depmgr hostname>Network*.

- cells/<depmgr nodename>Network/clusters/

  Contains one directory for each of the clusters managed as part of this cell. Each cluster directory contains a single file, *cluster.xml*, that defines the application servers (of one or more nodes) that are members of the cluster.

- cells/<depmgr nodename>Network/nodes/

  Contains one directory for each of the nodes managed as part of this cell.

- cells/<depmgr nodename>Network/nodes/<nodename>/

  Contains a *servers* directory (see below) as well as node-level configuration settings.

- cells/<depmgr nodename>Network/nodes/<nodename>/servers/

  Contains one directory for each of the servers managed as part of this node.

  > **Note:** In a Network Deployment configuration, in addition to the application server directories, there will exist the following server directories on each node:
  >
  > - jmsserver
  >
  >   Contains the definition and settings for the JMS Server process.
  >
  > - <nodename>
  >
  >   Contains the definition and settings for the node agent process.

- cells/<depmgr nodename>Network/nodes/<nodename>/servers/<servername>/

  Contains server-level configuration settings.

Note: The same data repository structure is used for the Base and Network Deployment topologies. The only difference is the cell name.

Variable-scoped documents

Identically named documents that exist at differing levels of the configuration hierarchy are called “variable scoped” documents.

There are two uses for variable scoped documents:

- Configuration data contained in a document at one level is logically combined with data from documents at other levels of the configuration hierarchy. In the case of conflicting definitions, the “most specific” value takes precedence. For example:

  If an identical entry exists in the files at the cell and node level (like a variable defined in both the cell and node’s *variables.xml*), the entry at the node level takes precedence.

- Documents representing data that is not merged but is rather scoped to a specific level of the topology. For example:

  The *namestore.xml* document at the cell level contains the cell persistent portion of the namespace, while the *namestore.xml* at the node level contains the node persistent root of the namespace.

Top level configuration files

The top level (config) directory typically contains the following documents.

Table 12-2 Top level configuration files

File	Purpose	Variable scope (merged / not merged)
Plugin-cfg.xml	Web server plug-in configuration settings.	not merged
Plugin-cfg-service.xmi	Configuration of automatic plug-in regeneration settings.	not merged

Cell level configuration files

Under individual cell directories, there is a *cell.xml* file that contains configuration data specific to that cell. There are also several cell level documents that contain configuration data for all managed processes running on that node. The cell level directory typically contains the following documents.

Table 12-3 Cell level configuration files

File	Purpose	Variable scope (merged / not merged)
admin-autz.xml	Administrator access authorization settings	not merged
cell.xml	Configuration data specific to the cell	not merged
filter.policy	Java 2 security policy file - permissions defined here will be filtered out in the node (app.policy) and application (was.policy) level files.	not merged
integral-jms-authorizations.xml	Manage access authorizations to resources owned by the WebSphere Application Server Integral JMS Provider	not merged
multibroker.xml	Data replication service settings	not merged
namestore.xml	Cell persistent portion of the namespace	not merged
naming-autz.xml	Naming authorization settings	not merged
pmirm.xml	PMI request metrics	not merged
resources.xml	Resources available for entire cell, e.g.. JMS provider resources, JMS connection factories, JMS Destinations, JDBC providers, etc.	merged
security.xml	Security configuration for the cell	not merged
variables.xml	Variable substitution values to use for processes running on the cell	merged
virtualhosts.xml	Virtualhosts used by all servers in the cell	not merged

Node level configuration files

Under each individual node directory, there is a *node.xml* file that contains configuration data specific to that node. There are also several node level documents that contain configuration data for all managed processes running on that node. The node level directory typically contains the following documents.

Table 12-4 Node level configuration files

File	Purpose	Variable scope (merged / not merged)
app.policy	Java 2 security file - access policies for all applications on this node.	not merged
library.policy	Java 2 security file - access policies for all libraries used by this node	not merged
namestore.xml	Node persistent portion of the namespace	not merged
node.xml	Configuration data specific to the node	not merged
resources.xml	Resources available for entire node only	merged
serverindex.xml	Lists the servers associated with the node, the applications deployed to each server and the IP port numbers used by each server's services.	not merged
spi.policy	Java 2 security file - access policies for all service provider interface (SPI) classes on this node	not merged
variables.xml	Variable substitution values to use for processes running on the node	merged

Server level configuration files

Under each individual server directory, there is a *server.xml* file that contains configuration data specific to that server. There are also several server level documents that contain configuration data for the services hosted by the server. The server level directory typically contains the following documents.

Table 12-5 Server level configuration files

File	Purpose	Variable scope (merged / not merged)
namestore-cell.xml	Cell persistent portion of the namespace	not merged
namestore-node.xml	Node persistent portion of the namespace	not merged
resources.xml	Resources available on the server	merged
server.xml	Configuration data specific to the server	not merged

File	Purpose	Variable scope (merged / not merged)
variables.xml	Variable substitution values to use for processes running on the server	merged

12.4.2 Application data repository - Base

The configuration repository is also used to store the application data (binaries and deployment descriptors). This allows modified deployment descriptors to be kept at the configuration level, and allows System Management to make application updates more automatic.

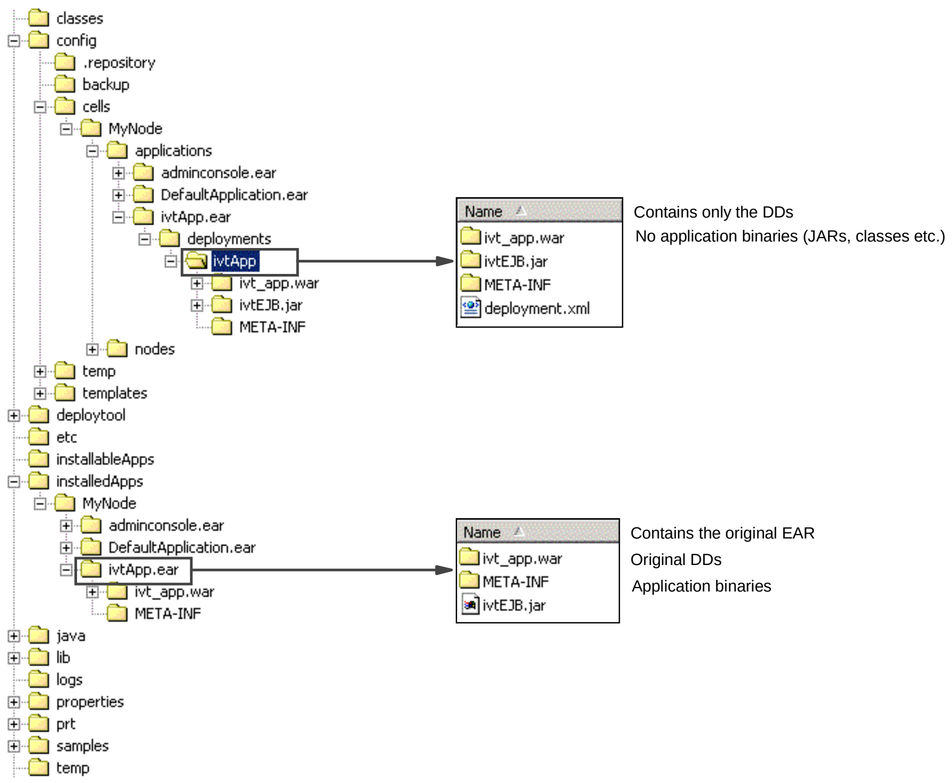


Figure 12-8 Application data repository for Base configuration

Key features

The application data repository of the base environment has a number of key features as shown in Figure 12-8:

- The *config/cells/<cellname>/applications* directory contains a subdirectory for each application deployed in the cell. The names of the directories match the names of the deployed applications.

  Note: The name of the deployed application does not have to match the name of the original EAR used to install it. Any name can be chosen when deploying a new application, as long as the name is unique across all applications in the cell.

- Each *config/cells/<cellname>/applications/<appname>.ear* directory contains:
  - A copy of the original EAR, called *<appname>.ear*. This EAR does not contain any of the bindings specified during installation of the application.
  - A *deployments* directory, which contains a single *<appname>* directory used to contain the deployed application configuration.
- Each *config/cells/<cellname>/applications/<appname>.ear/deployments/<appname>* directory‘:
  - Contains a *deployment.xml* file which contains configuration data for the application deployment.
  - Contains a META-INF directory containing the J2EE application deployment descriptor file (*application.xml*)as well as the IBM bindings and extensions files.

    Note: The deployment descriptors contains the bindings chosen during application installation.

  - Contains subdirectories for all application modules (WARs and EJB JARs), along with each module’s deployment descriptors.
  - Does not contain application binaries (JARs, classes, JSPs).
- The *installedApps/<cellname>* directory contains a subdirectory for each application deployed to the local node. This is the *local application binaries directory*.

Note:

1. The name of the directory reflects the name under which the application is installed, not the name of the original EAR. For example, if an application is called *myapp*, then the *installedApps/<cellname>* directory will contain a *myapp.ear* subdirectory.
2. The *installedApps/<cellname>* directory only contains the binaries for applications that have modules deployed to application servers on the node. The master copy of each application's binaries is stored as an EAR file in the *config/cells/<cellname>/applications/<appname>.ear* directory of the cell master configuration repository.

- Each *installedApps/<cellname>/<appname>.ear* directory contains:
  - The contents of the original EAR used to install the application.
  - The deployment descriptors from the original EAR. These descriptors do not contain any of the bindings specified during application deployment.
  - All application binaries (JARs, classes, JSPs).

12.4.3 Application data repository - Network Deployment

The application data repository used by a Network Deployment environment contains a few minor differences compared to the Base environment.

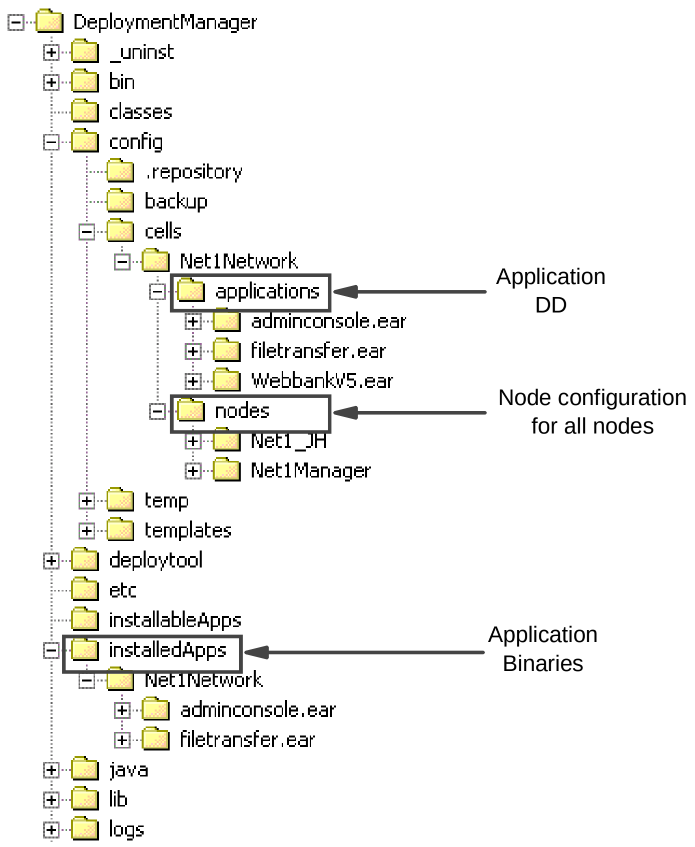


Figure 12-9 Application data repository for Network Deployment configuration

The differences as shown in Figure 12-9 are:

- The configuration repository structure *config/cells/...* on the deployment manager machine contains the configuration and application binaries for all nodes in the cell.
- The configuration repository structure config/cells/... on a node machine contains the directory structure for all nodes, but only the configuration data and application binaries for this node.

12.5 Application management

The system management of IBM WebSphere Application Server encompasses functionality relating to application management. The areas of application management addressed include:

- Application installation
- Application distribution
- Operational control
- Application upgrade and reinstall
- Application tools

12.5.1 Application installation

IBM WebSphere Application Server provides a common installation architecture across all editions, as compared to WebSphere 4.x where different tools and techniques had to be used depending upon the edition.

The steps performed for application installation involve:

- Validation of the application archive.
- Collection of configuration information from the end-user, either interactively (via **`wsadmin`** or the Web administration console) or non-interactively (via a pre-prepared script).
- Registration of the application with the cell.
- Distribution of configuration changes and application binaries to each of the nodes controlling application servers chosen to host the application’s modules.

For detailed information on application installation, see Chapter 20, “Deploying an application” on page 835.

Author Comment: Update the reference above with a specific heading when Isabelle sends the chapter.

12.5.2 Application distribution

IBM WebSphere Application Server provides automatic distribution of applications which involves the transfer of application configuration and binaries to servers that the application runs on in a multi-node environment.

This is a much improved feature in comparison to WebSphere 4.0 where the administrator was expected to copy and expand the application binaries on all the nodes that the application is installed, before the application could be run.

Application management performs distribution in the following cases:

- Application installation
- Application reinstall or upgrade
- Addition of a new member server in a server cluster

  When an application server is added as a member to a server cluster then the modules installed on other members are also installed on the new member.
- Rebinding of a module to a server or a server cluster

  After an application is installed in the WebSphere cell, its individual modules can be rebound to another stand-alone server or server cluster. When a module is bound to a server and if the node that the server runs on does not already have the application binaries then application transfer logic is executed.

12.5.3 Application operational control

Applications and modules are represented using JMX Management Beans (MBeans) in the system management architecture. For an introduction to JMX, see Section 12.6, "System management tools" on page 435.

Author Comment: Rework this list and references when the other chapters are done.

The following operations are possible on applications and modules:

- Start / stop of an application.
  - Using the Web administration console (see 13.7.3, "Enterprise applications" on page 485)
  - Using wsadmin (see need reference here)
- Viewing and editing of an application's configuration (see need reference here to specific sections in admin config and app deployment).
- Monitoring application status (see Chapter 21, "Monitoring and tuning your runtime environment" on page 839).

12.5.4 Application upgrade and reinstall

The current application upgrade functionality provided by IBM WebSphere Application Server requires that an application be reinstalled. There is no introspection on the changes (or otherwise) of individual components.

The current upgrade functionality has does not provide for continuous application availability. The upgrade process performs the following tasks which cannot provide continuous availability of an application to end-users.

1. Stops the application if it is running.
2. Uninstalls the current application.
3. Installs the new version of the application.

Note: A future version of IBM WebSphere Application Server may provide built-in support for application versioning, where multiple versions of an application can co-exist.

12.5.5 Application tools

IBM WebSphere Application Server provides tools for the deployment and management of applications.

Author Comment: Need to figure out where the admin console and wsadmin tools are first mentioned and decide where to introduce them. I think this section comes a little late.

Application assembly tools

IBM WebSphere Application Server provides a version of the Application Assembly Tool (AAT) that provides:

- Support for J2EE 1.3.
- Usability enhancements.

For detailed information on assembling and packaging applications using this tool, see Chapter 19, “Packaging an application” on page 691.

Application management tools

The common IBM WebSphere Application Server system management tools are used for application management:

- Web administration console
- wsadmin command-line tool

For information on the system management tools, see Section 12.6, “System management tools” on page 435.

12.6 System management tools

The IBM WebSphere Application Server administration tools are used for system management for the entire distributed topology:

- Web administration console
- Command-line operational tools
- WebSphere scripting: IBM WebSphere Application Server includes a new scripting tool called *wsadmin*. This will be covered in detail in Chapter 19, “Scripting” on page 339.
- Java APIs: The Java-based JMX APIs can be accessed directly by custom Java applications. See Section 12.6, “System management tools” on page 435 for further details on the JMX Java APIs.

Although any of these interfaces can be used to configure system management functions, use of the Web administration console is preferred because it validates any changes made to the configuration.

12.6.1 Web administration console

The Web administration console runs as a J2EE 1.3 web application within a managed server process on a node. The location and configuration of the console differs depending upon whether the Base or Network Deployment environment is in use:

Base

In the IBM WebSphere Application Server configuration, the Web administration console is installed and accessed on the default application server (server1).

Note: New application servers created using the admin tools do not have the Web administration console installed. However, the Web administration console Enterprise Application Archive (EAR) can be installed manually into each application server if so desired.

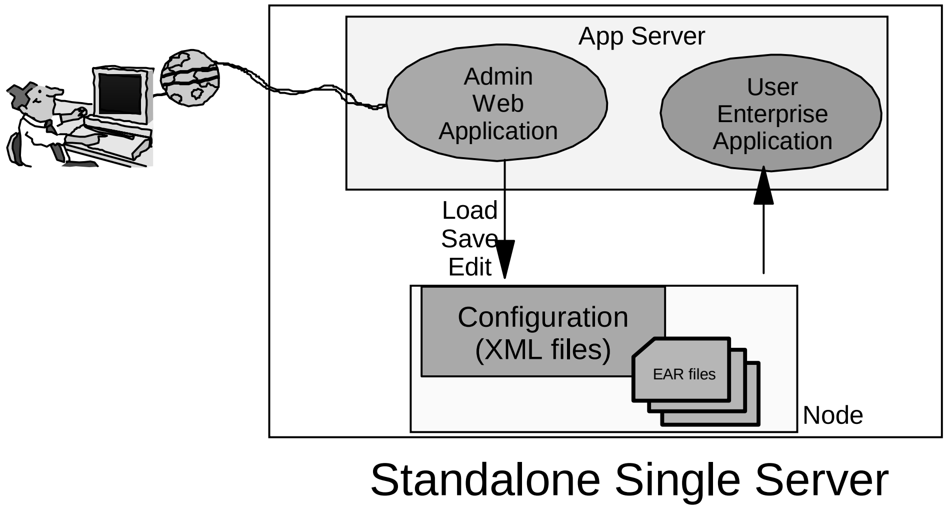


Figure 12-10 Standalone server Web administration console

Network Deployment

In the IBM WebSphere Application Server Network Deployment configuration, the Web administration console is hosted on the deployment manager process. The act of adding a node to a cell results in the Web administration console being removed from any application server. However, the Web administration console can be installed manually into one or more Application Servers, for example, in a very large cell configuration where some local web -based “tweaking” of application server settings is desired.

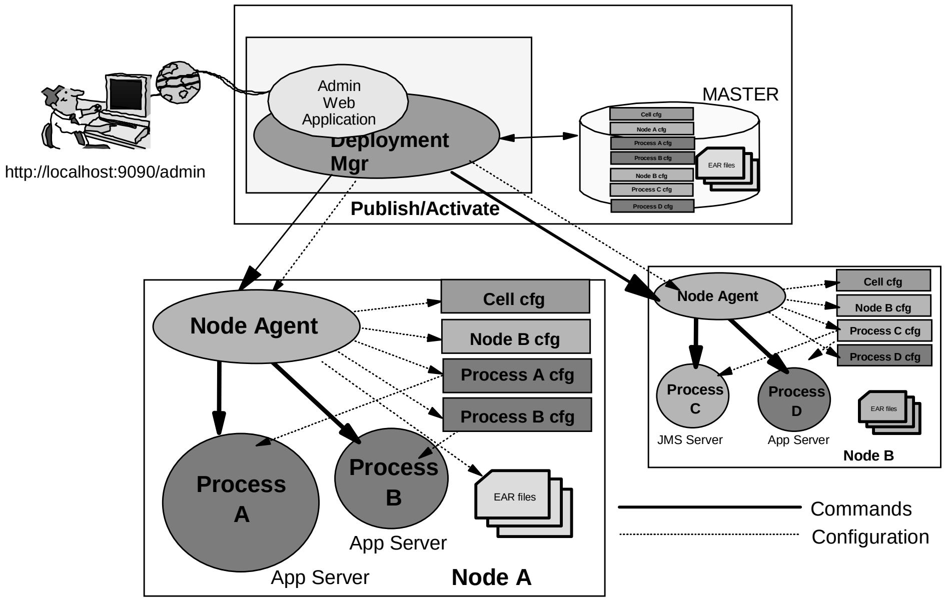


Figure 12-11 Deployment manager Web administration console

For detailed information regarding the features and use of the Web administration console, see Chapter 13, “WebSphere administration basics” on page 449.

12.6.2 Command-line operational tools

IBM WebSphere Application Server provides command-line tools for operational administration of the runtime environment. Table 12-6 shows a summary of the commands available and a pointer to the syntax.

Table 12-6 Commands

Command	**Syntax reference**
startManager	Section A.1, “startManager” on page 1036
stopManager	Section A.2, “stopManager” on page 1037
startNode	Section A.3, “startNode” on page 1039
stopNode	Section A.4, “stopNode” on page 1040
addNode	Section A.5, “addNode” on page 1041
removeNode	Section A.6, “removeNode” on page 1044

Command	**Syntax reference**
syncNode	Section A.7, “syncNode” on page 1046
cleanupNode	Section A.8, “cleanupNode” on page 1048
startServer	Section A.9, “startServer” on page 1049
stopServer	Section A.10, “stopServer” on page 1050
serverStatus	Section A.11, “serverStatus” on page 1052
createmq	Section A.12, “createmq” on page 1053
deletemq	Section A.13, “deletemq” on page 1055
backupConfig	Section A.14, “backupConfig” on page 1056
restoreConfig	Section A.15, “restoreConfig” on page 1057
instance	Section A.16, “instance” on page 1059

12.7 Common system management tasks

This section summarizes some of the most common system management tasks:

- Adding a node to a cell.
- Removing a node from a cell.
- Force synchronization of a node’s configuration.
- Cleaning up a node.
- Backing up a node configuration.
- Restoring a node configuration.
- Backing up the cell configuration.
- Restoring the cell configuration.
- Creating multiple instances (nodes) on a single machine.
- Starting a Network Deployment environment.
- Stopping a Network Deployment environment.
- Enabling process restart on failure.
- Installing an application.
- Updating an application.

12.7.1 Add node to a cell

Use the **`addNode`** command to add a standalone node to an existing cell. The principles of this operation are shown in Figure 12-12.

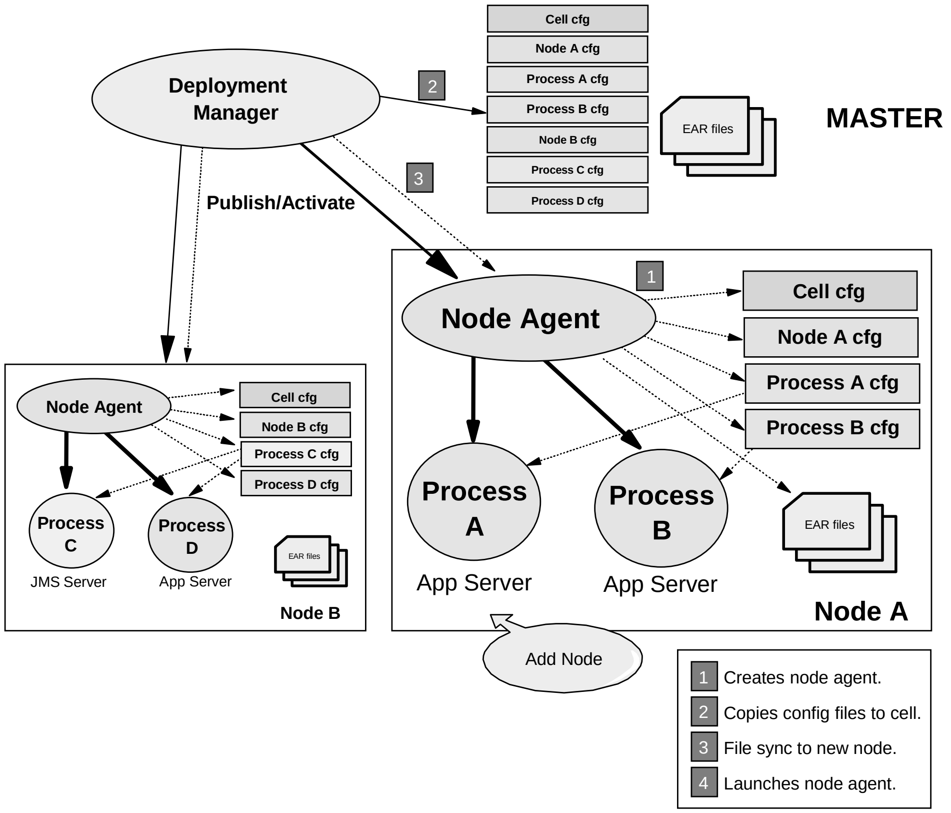


Figure 12-12 Add node to cell

12.7.2 Remove node from a cell

Use the **`removeNode`** command to detach a node from a cell and return it to a standalone configuration. See Section 12.6.2, “Command-line operational tools” on page 437 for detailed information on the **`removeNode`** command.

12.7.3 Force synchronization of node configuration

An Administrator may be required to explicitly force a resynchronization of a node’s configuration, and thereby update its local configuration repository to reflect the current contents of the cell’s master repository. For example, if the node agent was not running when changes were made to the cell repository.

In such instances, the **`syncNode`** command is used to force the resynchronization of a node with its cell. See Section 12.6.2, “Command-line operational tools” on page 437 for detailed information on the **`syncNode`** command.

12.7.4 Clean up a node

The IBM WebSphere Application Server software does not provide a command specifically for cleaning up an installation, e.g.. removing log files. As such, an Administrator must use a manual process.

Suggested directories that can be cleaned up include:

1. On deployment manager machine:
   - <WAS_ROOT>/wstemp

     Contains the temporary working areas for unsaved (uncommitted) workspace changes for each administrator.
   - <WAS_ROOT>/logs/dmgr/

     Contains the logs files generated by the deployment manager process.
   - <WAS_ROOT>/logs/ffdc/

     Contains logs generated by the First Failure Data Collection (FFDC) subsystem.
2. On each node machine:
   - <WAS_ROOT>/logs/<server name>/

     Contains the logs files generated by this server process.
   - <WAS_ROOT>/logs/nodeagent/

     Contains the logs files generated by the node agent process.
   - <WAS_ROOT>/logs/ffdc/

     Contains logs generated by the First Failure Data Collection (FFDC) subsystem.

Important: Stop all managed servers on a machine before deleting any logs on that machine.

12.7.5 Backing up a node configuration

Use the **`backupConfig`** command to backup the configuration of a node. The command backs up an entire copy of the node’s config directory to a ZIP archive.

See Section 12.6.2, “Command-line operational tools” on page 437 for detailed information on the **`backupConfig`** command.

12.7.6 Restoring a node configuration

Use the **`restoreConfig`** command to restore the configuration of a node using an archive previously generated using **`backupConfig`**. The command restores the entire contents under the node’s *config* directory.

See Section 12.6.2, “Command-line operational tools” on page 437 for detailed information on the **`restoreConfig`** command.

12.7.7 Backing up the cell configuration

The master cell configuration repository can be backed up by running the **`backupConfig`** command on the machine hosting the deployment manager. The command backs up an entire copy of the node’s *config* directory to a ZIP archive.

See Section 12.6.2, “Command-line operational tools” on page 437 for detailed information on the **`backupConfig`** command.

12.7.8 Restoring the cell configuration

The master cell configuration repository can be restored by running **`restoreConfig`** on the machine hosting the deployment manager. This requires a backup archive previously generated by **`backupConfig`** on the same machine.

The command restores the entire contents under the deployment manager’s *config* directory.

See Section 12.6.2, “Command-line operational tools” on page 437 for detailed information on the **`restoreConfig`** command.

12.7.9 Creating multiple instances (nodes) on a single machine

IBM WebSphere Application Server provides the **`instance`** command as a means by which multiple instances (nodes) can be run on a single machine while using a single IBM WebSphere Application Server installation.

Each instance uses a separate configuration but uses a single installation of the WebSphere binaries. Each instance is distinguished by its base path, its own directory structure and its own *setupCmdLine* script to configure its command-line environment.

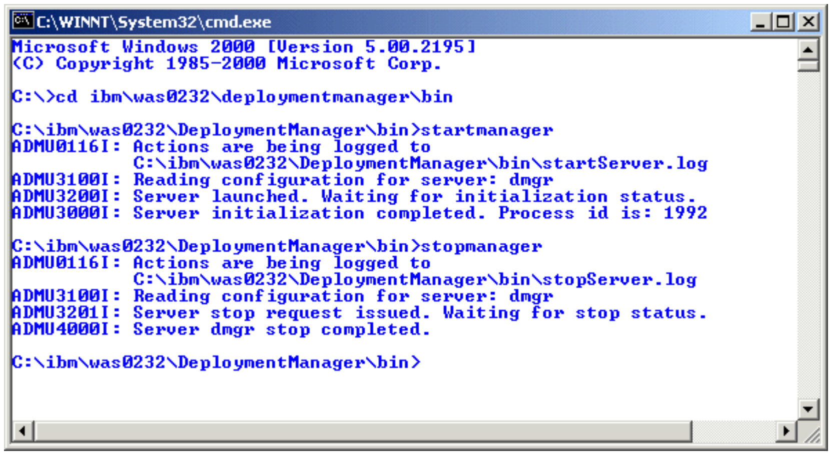
See Section 12.6.2, “Command-line operational tools” on page 437 for detailed information on the **`instance`** command.

12.7.10 Start the Network Deployment environment

The sequence of steps to follow to correctly start a Network Deployment environment involves the following:

1. On the deployment manager machine:
   a. Change directory to the bin directory of the Network Deployment installation.
   b. Use **`startmanager`** to start the deployment manager (dmgr) process. See A.1, “startManager” on page 1036, for command line options.

   If successful you will see the process ID for the network deployment manager process displayed on the screen, see Figure 12-13.



```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>cd ibm\was0232\deploymentmanager\bin

C:\ibm\was0232\DeploymentManager\bin>startmanager
ADMU0116I: Actions are being logged to
           C:\ibm\was0232\DeploymentManager\bin\startServer.log
ADMU3100I: Reading configuration for server: dmgr
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server initialization completed. Process id is: 1992

C:\ibm\was0232\DeploymentManager\bin>stopmanager
ADMU0116I: Actions are being logged to
           C:\ibm\was0232\DeploymentManager\bin\stopServer.log
ADMU3100I: Reading configuration for server: dmgr
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server dmgr stop completed.

C:\ibm\was0232\DeploymentManager\bin>
```

Figure 12-13 Starting and stopping the deployment manager from the command line

   If there are any errors, check the log file for the dmgr process:

   ```
   <WAS_ROOT>/logs/dmgr/SystemOut.log
   ```

2. On each node agent machine:
   a. Change directory to the bin directory of the base application server installation for that node.

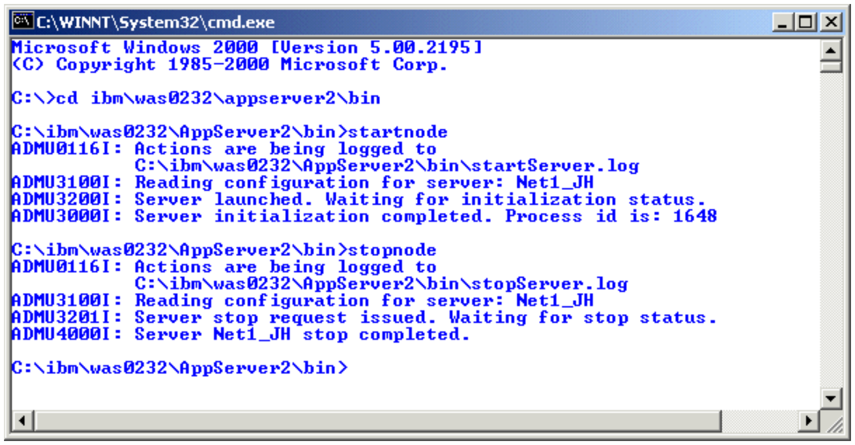
> **Note:** Installation of multiple nodes in a single physical machine requires the use of the **`instance`** script. See Chapter 12, “System Management” on page 405 for details. need more detailed reference here ..

b. Run **`startNode`** from the command line.

If successful, the node agent server process ID will be displayed on the screen, as shown in Figure 12-14.

If there are any errors check the log file for the node agent process:

```
<WAS_ROOT>/logs/nodeagent/SystemOut.log
```



```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>cd ibm\was0232\appserver2\bin

C:\ibm\was0232\AppServer2\bin>startnode
ADMU0116I: Actions are being logged to
           C:\ibm\was0232\AppServer2\bin\startServer.log
ADMU3100I: Reading configuration for server: Net1_JH
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server initialization completed. Process id is: 1648

C:\ibm\was0232\AppServer2\bin>stopnode
ADMU0116I: Actions are being logged to
           C:\ibm\was0232\AppServer2\bin\stopServer.log
ADMU3100I: Reading configuration for server: Net1_JH
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server Net1_JH stop completed.

C:\ibm\was0232\AppServer2\bin>
```

Figure 12-14 Starting and stopping the node agent from the command line

.

Note: The node agent and the deployment manager processes are independent and therefore can be started in any order. (this seems to contradict the logic here ..)

c. Use **`startserver`** to start each of the application server processes on the node.

d. Use **`startserver`** to start the *jmsserver* process on the node.

e. Check the node status by running the **`serverStatus -all`** command.

Note: Managed servers can be configured to be automatically started on the start of the node agent. If already running, each application server must be restarted anyway as each much register with the Location Service Daemon (LSD) of the node agent.

3. Repeat (2) for each and every node machine associated with this deployment manager.

12.7.11 Stop the Network Deployment environment

The sequence of steps to follow to correctly stop a Network Deployment environment involves the following steps. .

1. On each node agent machine:
   a. Use **`stopserver`** to stop each of the application server processes on the node.
   b. Use **`stopserver`** to stop the *jmsserver* process on the node.
   c. Use **`stopnode`** to stop the node agent process.
      i. Change directory to the *bin* directory of the base application server installation for that node.
      ii. Run **`stopNode`** from the command line. .

         You will see the message “`Server <node_agent> stop completed`” if the server stopped successfully, Figure 12-14.

         If there are any errors check the log file for the node agent process:

         <WAS_ROOT>/logs/dmgr/SystemOut.log

Important: Stopping the node agent doesn’t stop the application servers managed by that node. However, because the node agent runs the Location Service Daemon, it is recommended that you restart the application servers after restarting the node agent.

Author Comment: That needs to be confirmed - the implications of stopping and restarting a node agent without restarting the app servers

   d. Check the node status by running the **`serverStatus -all`** command.
2. Repeat (2) for each and every node machine associated with this deployment manager.
3. On the deployment manager machine:
   a. Change directory to the bin directory of the Network Deployment installation.
   b. Use **`stopmanager`** to stop the deployment manager (dmgr) process.

   The message “`Server dmgr stop completed`” will be seen if the server stopped successfully, as shown in Figure 12-13.

   If there are any errors check the log file for the dmgr process:

```
<WAS_ROOT>/logs/dmgr/SystemOut.log
```

.

Note: Stopping the deployment manager does not stop any node agents.

12.7.12 Enabling process restart on failure

Unlike IBM WebSphere Application Server 4.0, the 5.0 software does not have either:

- A nanny process to monitor whether the AdminServer process is running, and restart it if it has failed.
- An AdminServer process to monitor whether each application server process is running, and restart it if it has failed.

Instead, IBM WebSphere Application Server 5.0 uses the native operating system functionality to restart a failed process:

Windows 2000

The administrator can choose to register one (or more) of the IBM WebSphere Application Server 5.0 processes on a machine as Windows Services. Windows will then automatically attempt to restart the process if it has failed.

Syntax

```
WASService.exe
> >|| -add <service name>
> >   -serverName <Server>
> >   [-cellName <Cell>]
> >   [-nodeName <Node>]
> >   [-wasHome <Websphere Install Directory>]
> >   [-startArgs <additional launcher arguments>]
> >   [-userid <execution id> -password <password>]
> >   [-logFile <service log file>]
> >   [-restart <true | false>]
> >|| -remove <service name>
> >|| -start <service name>
> >|| -stop <service name>
> >|| -status <service name>
```

Notes

1. When adding a new service, only the *serverName* argument is mandatory.

Recommendation: Always supply the *cellName*, *nodeName* and *wasHome* arguments when adding a new service. Doing so will ensure that the process in the correct cell will be started and that the correct installation is used (for cases where multiple WAS 5.0 installations exist on the one machine, e.g.. a machine that hosts both the deployment manager and a node.

2. Use unique names for the WebSphere server processes registered as Windows services. The services will be listed in the Windows *Services* control panel as:

   IBM WebSphere Application Server V5 - <service name>

3. To configure the node agent as a Windows service, use a *serverName* that is the same as the *nodeName*. That is the convention for node agent server process names.
4. To configure the deployment manager as a Windows service, use the deployment manager specific *nodeName*.

Example

Example 12-4 Registering WebSphere processes as Windows Services

```
On Deployment Manager machine:
$cd c:\ibm\was\DeploymentManager\bin
$ WASService -add “Deployment Mgr” dmgr -cellname NetworkDeploymentCell
-nodeName NetmgrNode_JH -wasHome “c:\ibm\was\DeploymentManager” -restart

On node machine:
$cd c:\ibm\was\AppServer\bin
$ WASService -add “Node agent” Net1_JH -cellname NetworkDeploymentCell
-nodeName Net1_JH -wasHome “c:\ibm\was\AppServer” -restart
$ WASService -add “Server1” Net1_JH_server1 -cellname NetworkDeploymentCell
-nodeName Net1_JH -wasHome “c:\ibm\was\AppServer” -restart
$ WASService -add “Server2” Net1_JH_server2 -cellname NetworkDeploymentCell
-nodeName Net1_JH -wasHome “c:\ibm\was\AppServer” -restart
$ WASService -add “Server3” Net1_JH_server3 -cellname NetworkDeploymentCell
-nodeName Net1_JH -wasHome “c:\ibm\was\AppServer” -restart
```

Unix / Linux

The administrator can choose to include entries in inittab for one (or more) of the IBM WebSphere Application Server 5.0 processes on a machine, as shown in Example 12-5. Each such process will then be automatically restarted if it has failed.

Example 12-5 Inittab contents for process restart

```
On Deployment Manager machine:
   ws1:23:respawn:/usr/WebSphere/DeploymentManager/bin/startManager

On node machine:
   ws1:23:respawn:/usr/WebSphere/AppServer/bin/startNode
   ws2:23:respawn:/usr/WebSphere/AppServer/bin/startServer nodename_server1
   ws3:23:respawn:/usr/WebSphere/AppServer/bin/startServer nodename_server2
   ws4:23:respawn:/usr/WebSphere/AppServer/bin/startServer nodename_server2
```

13

WebSphere administration basics

In this chapter we introduce the WebShere administrative console and describe some of the basic tasks that are commonly performed by WebSphere administrators. The tasks we look at include:

- Working with cells, 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 WebShere administrative console

The WebShere administrative console is the graphical, Web-based admin tool that you use to configure and manage an entire WebSphere cell. It supports the full range of product administrative activities, such as creating and managing resources, applications, viewing product messages, etc. The administrative console is a standard J2EE 1.3 Web application running under the Deployment Manager server, *dmgr* and is installed by default when the Network Deployment Manager is installed.

Note: The administrative console application also gets installed when you install a Base instance of the IBM WebSphere Application Server on a node. However as the node is added to a network deployment cell, the administrative console application is removed from the node.

The administrative console provides centralized administration of multiple nodes, and allows nodes on multiple machines to be administered. The configuration data for an IBM WebSphere Application Server Network Deployment cell, is a set of XML documents arranged in a set of cascading directories under <WAS_ROOT>/config directory. With the administrative console we load and make changes to the master repository XML configuration files. It is the Deployment Manager server's responsibility to push those changes to the local XML repositories on the nodes.

Note: In the Network Deployment environment, it is possible to install the administrative console an application server on any of the nodes of the cell and attach the administrative console to the server. This allows for local administration of the server. However any changes made to the server configuration will be **temporary.** At the next scheduled data update time (file synchronization time), the network deployment manager pushes the master configuration data to the nodes and any changes made at the server level are lost. For changes to be **permanent**, they must be performed at the Deployment Manager level.

Author Comment: Picture? Admin Console in a Multi-node Topology. ArchitectAdmin.pdf pg 18

In order for the administrative console to run, the *dmgr* server must be running in the node where Network Deployment (and therefore the administrative console) is installed.

In order for the changes to the master repository to be pushed to the nodes, the node agents must also be running in the nodes where the WebSphere 5.0 instances are installed.

Note: WebSphere scripting can also be used to configure and modify configuration settings. See Chapter 19, “Scripting” on page 339

In WebSphere 5.0, the administrative console groups administrative tasks by the type of component, as follows:

Author Comment: Verify this at GM time

- Security Center
- Environment
- Servers
- Applications
- Resources
- Cluster
- Problem Determination

Note: For users new to J2EE, there have been major changes to the WebSphere administrative console between WebSphere V3.5 and WebSphere V5.0. The changes reflect the structure of J2EE applications.Familiarity with the concepts underlying a J2EE runtime environment is required in order to effectively manage a WebSphere V5.0 environment.

For the latest support information, check the prerequisites Web page:

`http://www-3.ibm.com/software/webservers/appserv/doc/latest/prereq.html`

for supported product levels.

The following browsers are supported:

- Microsoft Internet Explorer 5.0, 5.5, 6.0 (or later).
- Netscape Navigator 4.7.x (varies by platform).

13.2 Starting the administrative console

In IBM WebSphere Application Server Network Deployment, the administrative console is deployed as a J2EE application:

- Application binaries

  ```
  <WAS_ROOT>/config/apps/adminconsole.ear
  ```

- Application configuration:

  ```
  <WAS_ROOT>/config/cells/NetworkDeploymentCell/applications/adminconsole
  ```

The application is managed by the Network Deployment Manager process, dmgr.

Therefore, to start the WebSphere administrative console:

1. Make sure that the application server for the administrative console, dmgr, is running.

   ```
   <WAS_ROOT>\bin\startManager.bat(sh) [options]
   ```

Author Comment: Mention the windows first steps window .. plus add boxes for other platforms. This step says to make sure dmgr is running .. how?

   See A.1, “startManager” on page 1036, for information on the **`startManager`** command.

   Note: In this section we assume that connection is made to the administrative console installed in the Network Deployment Manager node, which is recommended. If you have installed, and are connecting to one of the nodes via the administrative console , then make sure that the application server on which the administrative console application is installed, is started.

2. Start the Web Server, if accessing the administrative console through a Web Server.
3. Point the Web browser at the URL for the administrative console:. The default is port is 9090 for HTTP and 9043 for HTTPS.
   – http://your.server.name:9090/admin
   – https://your.server.name:9043/admin

   where *your.server.name* is the hostname for the machine running the Network Deployment Manager process, dmgr.

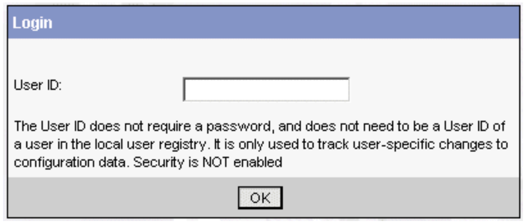
Author Comment: For base installations, I think the admin console port is 9080. Not sure about https.

Note: The URL above assumes you are accessing the administrative console through the embedded transport for the dmgr server. To access the administrative console through a Web server, see Chapter 16, “Configuring the Web server interface” on page 549.

If you need two concurrent sessions on the same client machine, access the administrative console from two different browser types, whether or not you use the same user ID. This will allow for two different HTTP session objects.

Author Comment: The above needs to be covered, acces to the Admin Console through a Web Server and the reference should be made more specific.

4. The administrative console will load into the browser and you will be asked to login.



The User ID does not require a password, and does not need to be a User ID of a user in the local user registry. It is only used to track user-specific changes to configuration data. Security is NOT enabled

Figure 13-1 Administrative console login

13.2.1 Logging into the console

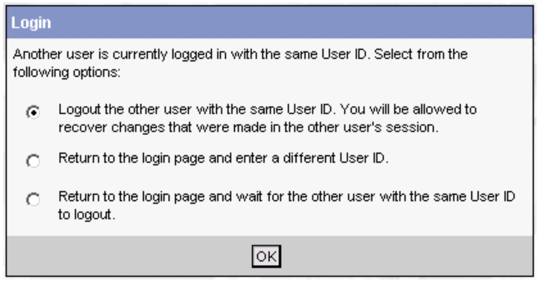
The user ID specified during login is used to track configuration changes made by the user. This allows you to recover from unsaved session changes made under the same user ID, for example when a session times out or the user closes the Web browser without saving.

The user ID used for login depends on whether WebSphere global security is enabled.

- **No security:** If there global security is not enabled, you can enter any user ID, valid or not to login to the administrative console. The user ID is used to track changes to the configuration but is not authenticated.
- **WebSphere global security is enabled**: If global security is enabled, you must enter a valid user ID and password.

A user ID must be unique to the Network Deployment Manager. If you enter an ID that is already in use (and in session), you will receive the message *Another user is currently logged with the same User ID* and you will be prompted to do one of the following, as shown in Figure 13-2:

- Force the existing user ID out of session. You will be allowed to recover changes that were made in the other user's session.
- Wait for the existing user ID to log out or time out of the session.
- Specify a different user ID.
-



Login

Another user is currently logged in with the same User ID. Select from the following options:

[✓ Logout the other user with the same User ID. You will be allowed to recover changes that were made in the other user's session.]

Return to the login page and enter a different User ID.

Return to the login page and wait for the other user with the same User ID to logout.

Figure 13-2 Another user is currently logged on with the same user name

Note: This message will appear if the user closed a Web browser after browsing the administrative console and did not first log out.

Recovering from an interrupted session

Until you save the configuration changes you make during a session, the changes do not become effective. If you close a session without saving the configuration changes you made, or if the session times out, these changes are remembered and you are given the chance to pick up where you left off.

When unsaved changes for the user ID exist during login, you will be prompted to do one of the following as shown in Figure 13-3:

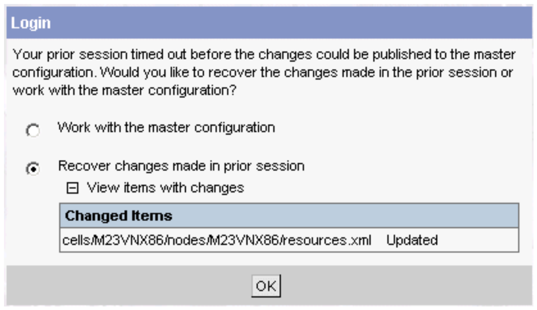
- Work with the master configuration

When enabled, specifies that you want to use the last saved administrative configuration. Changes made to the user's session since the last saving of the administrative configuration will be lost.

- Recover changes made in prior session

  When enabled, specifies that you want to use the same administrative configuration last used for the user's session. Recovers all changes made by the user since the last saving of the administrative configuration for the user's session.

Note: These fields appears only if the user changed the administrative configuration and then logged out without saving the changes. or if the session times out??



Login

Your prior session timed out before the changes could be published to the master configuration. Would you like to recover the changes made in the prior session or work with the master configuration?

( ) Work with the master configuration

[✓ Recover changes made in prior session]

⊟ View items with changes

Changed Items	
cells/M23VNX86/nodes/M23VNX86/resources.xml	Updated

Figure 13-3 Recover changes from previous unsaved sessions

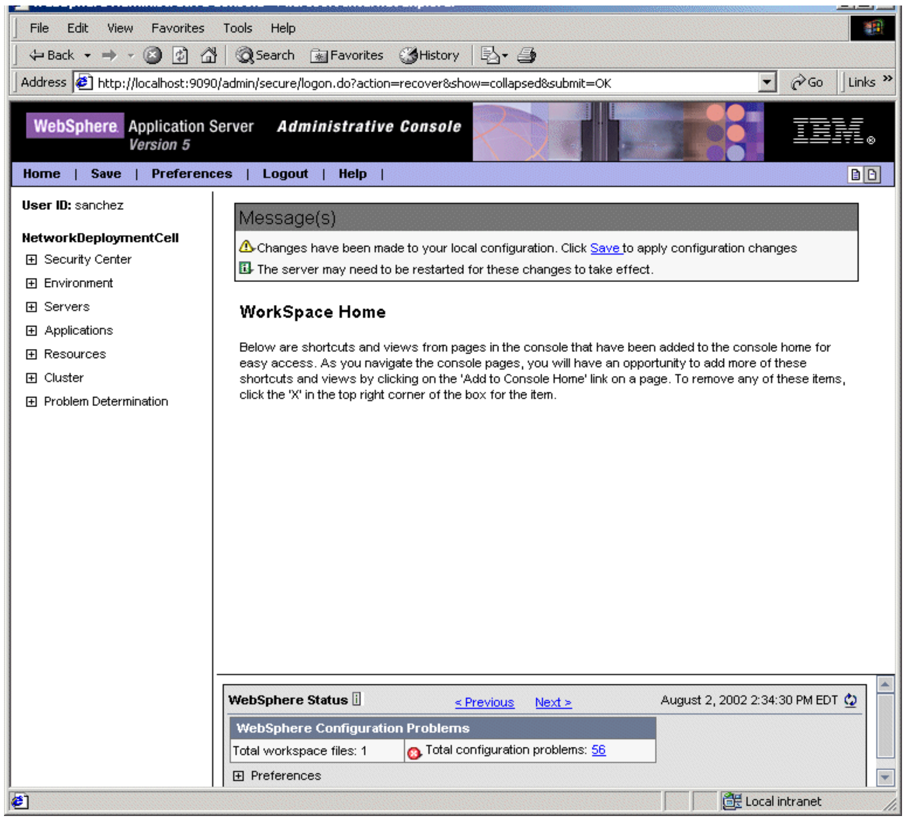
Author Comment: need to recapture the fig above in an ND environment and make sure it matches the text (the text below says deployment.xml was changed but not saved.

Tip: You may want to change the session timeout for the administrative console application. This is the time for the session to time out when the console is not used. The default is 30 minutes. how ??

For example, Figure 13-3 shows that in this configuration and under the specified user ID, the deployment.xml file for the administrative console was changed but not saved during the last session. The changes were stored in the user's workspace under:

`%user.install.root%/wstemp/userID/workspace/cells/NetworkDeploymentCell/applications/adminconsole/deployment.xml`.

If we choose to recover the changes and **Save** the master repository gets updated with the new deployment.xml file and the file gets removed from the user's workspace.


File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do?action=recover&show=collapsed&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: sanchez
NetworkDeploymentCell
Security Center
Environment
Servers
Applications
Resources
Cluster
Problem Determination
Local intranet

Message(s)

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

The server may need to be restarted for these changes to take effect.

WorkSpace Home

Below are shortcuts and views from pages in the console that have been added to the console home for easy access. As you navigate the console pages, you will have an opportunity to add more of these shortcuts and views by clicking on the 'Add to Console Home' link on a page. To remove any of these items, click the 'X' in the top right corner of the box for the item.

WebSphere Status < Previous Next > August 2, 2002 2:34:30 PM EDT

WebSphere Configuration Problems	
Total workspace files: 1	Total configuration problems: 56

Preferences

Figure 13-4 Saving changes made in prior session

All changes made to the administrative configuration during a normal session will be saved under the user id temporary workspace.

The default location for temporary workspace files and for a particular user ID is calculated based on the user installation root:

`%user.install.root%/wstemp/userID/workspace`

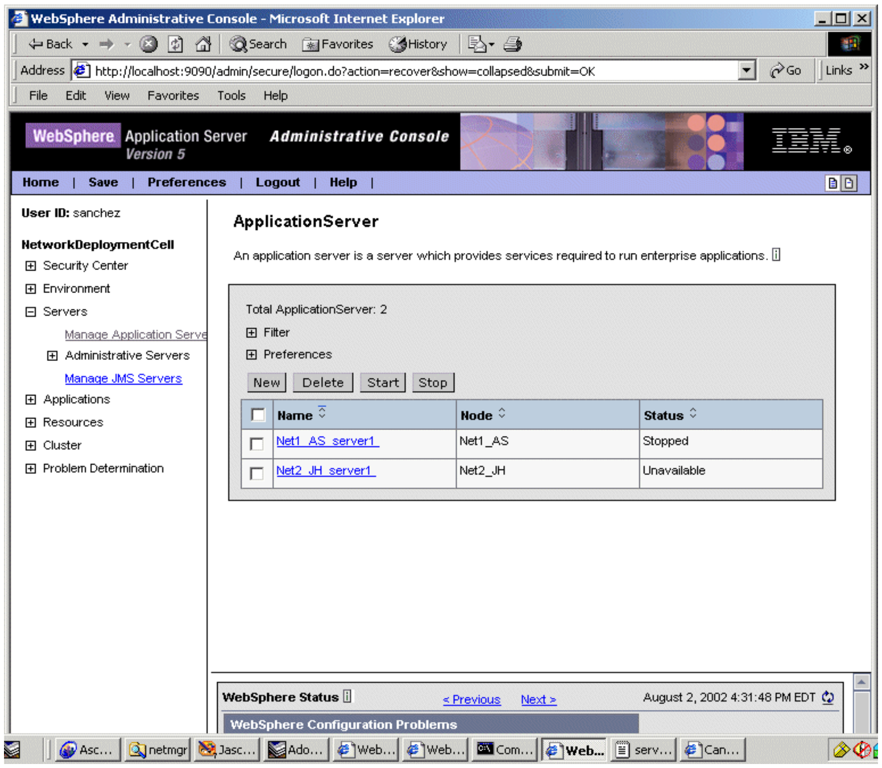
For information on how to change the default location refer to the documentation in the InfoCenter.

13.3 The graphical interface

The WebShere administrative console has the following main areas:

- Taskbar
- Navigation tree
- Workspace
- Status message area

Each area can be resized as desired.



http://localhost:9090/admin/secure/logon.do?action=recover&show=collapsed&submit=OK

User ID: sanchez

NetworkDeploymentCell
- ⊞ Security Center
- ⊞ Environment
- ⊟ Servers
  - Manage Application Serve
  - ⊞ Administrative Servers
  - Manage JMS Servers
- ⊞ Applications
- ⊞ Resources
- ⊞ Cluster
- ⊞ Problem Determination

ApplicationServer

An application server is a server which provides services required to run enterprise applications.

Total ApplicationServer: 2

⊞ Filter

⊞ Preferences

	Name	Node	Status
☐	Net1_AS_server1	Net1_AS	Stopped
☐	Net2_JH_server1	Net2_JH	Unavailable

WebSphere Status August 2, 2002 4:31:48 PM EDT

WebSphere Configuration Problems

Figure 13-5 The web administration console graphical Interface

13.3.1 Taskbar

The taskbar, shown in Figure 13-5, offers options for you to:

- Return to the console Home page.
- Save changes that you have made to administrative configurations.
- Specify console-wide preferences.

- Log out of the console.
- Access product information.

To view the taskbar, go to the WebSphere Application Server administrative console and look at the horizontal bar near the top of the console. The taskbar provides the following actions.

- **Home**

  Displays the administrative console Home page, called the WorkSpace Home. The WorkSpace Home is the original page displayed when you access the console.

  It contains shortcuts and views from pages in the console that have been added to the console home for easy access.
- **Save**

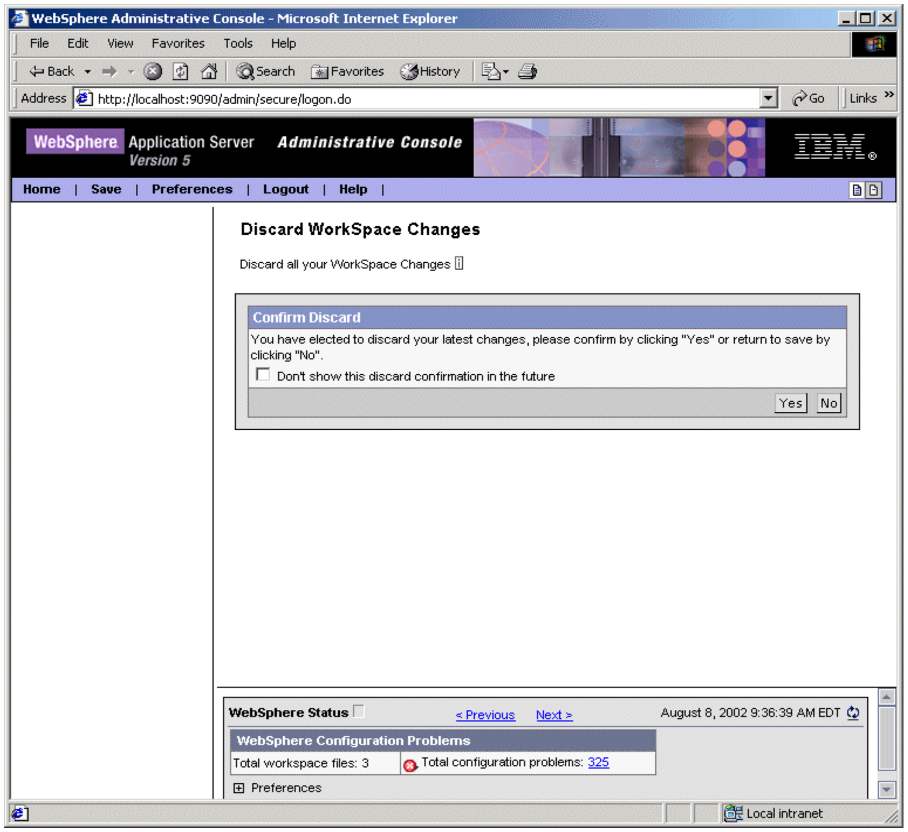
  Displays the Save page, which is used to save changes made to the administrative configuration to the master repository(.xml files).

  To see what changes will be saved, expand View items with changes in the Save to Master Repository window.

Important: The Save page presents you with three options: **`Save`**, **`Discard`** and **`Cancel.`** Discard reverses any changes made during the working session and reverts to the master configuration. Cancel however doesn't reverse changes made during the working session. It just cancels the action of Saving to the master repository for now. So if you don't want the changes made during your working session to be saved, choose Discard.

- **Preferences**

  Displays the Preferences page, in which you can specify whether you want the administrative console WorkSpace page to refresh automatically after administrative configuration changes, whether a confirmation dialog will display after you choose to discard all your Workspace changes,see Figure 13-6, and whether the default scope will be the administrative console node.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Yes No
Local intranet

Discard WorkSpace Changes

Discard all your WorkSpace Changes

Confirm Discard

You have elected to discard your latest changes, please confirm by clicking "Yes" or return to save by clicking "No".

Don't show this discard confirmation in the future

WebSphere Status < Previous Next > August 8, 2002 9:36:39 AM EDT

WebSphere Configuration Problems

Total workspace files: 3	Total configuration problems: 325

Preferences

Figure 13-6 Confirm discard

- **Logout**

  Logs you out of the administrative console session and displays the Login page. If you have changed the administrative configuration since last saving the configuration to the master repository, the Save page will display before returning to the Login page. Click **`Save`** to save the changes, **`Discard`** to return to the administrative console, or **`Logout`** to exit the session without saving changes.

- **Help**

  Opens a new Web browser with online help for the WebSphere Application Server product.

- **Hide Field and Page Descriptions toggle**

Enables you to select whether information on console pages and fields within the console pages is shown. Two icons on the right-hand side of the taskbar provide the toggle.

When enabled, click on the “i” icon beside a field or page description to view more information.

13.3.2 Navigation tree

The navigation tree, on the left side of the console, Figure 13-5, offers links for you to survey, select and manage components in the WebSphere administrative cell.

Clicking on a "+" beside a tree folder or item expands the tree for the folder or item. Clicking on a "-" collapses the tree for the folder or item. Double-clicking on an item in toggles its state between expanded and collapsed.

The content displayed on the right side of the console, the *Workspace*, depends on the folder or item selected in the tree view.

The following folders are provided for selection:

- **Security Center**

  Accesses the Security Center, which is used to secure applications and servers and to manage users of the administrative console.
- **Environment**

  Enables configuration of hosts, Web servers, variables and other components.
- **Servers**

  Enables configuration of administrative servers and other types of servers such as JMS servers.
- **Applications**

  Enables installation of applications onto servers and manage the installed applications.
- **Resources**

  Enables configuration of resources and to view information on resources existing in the administrative cell.
- **Clusters**

  Enables management and viewing of cluster topology.
- **Problem Determination**

  Enables you to check for and track configuration errors and problems.

13.3.3 WorkSpace

The workspace, on the right side of the console in Figure 13-5, allows you to work with your administrative configuration after selecting an item from the console navigation tree.

When you click a folder in the tree view, the workspace lists information on instances of that folder type. For example, selecting **`Servers`** -> **`Manage Application Servers`**, shows all the application servers configured in this cell. Selecting an item, an application server in this example, will bring the **`Properties`** page for that item. The Properties page can then be used to view and edit property values.

13.3.4 Status Message area

The messages area, shown in Figure 13-5, displays along the bottom of the console and remains visible as you navigate from the WebSphere Home page to other pages. The area displays two frames:

1. **WebSphere Configuration Problems** and
2. **WebSphere Runtime Messages**.

Click **`Previous`** or **`Next`** to toggle between the frames. Click on the number to view details.

The interval between automatic refreshes can be adjusted in the **`Preferences`** settings. In addition, the information displayed can be refreshed at any time by clicking the icon in the upper-right of the area.

For information on how to filter messages and select logging levels, refer to Chapter 22, “Troubleshooting” on page 903.

13.4 Using the administrative console

The following sections describe how to use the graphical Web-based administrative console tool to manage the WebSphere Application Server cell.

13.4.1 Finding an item

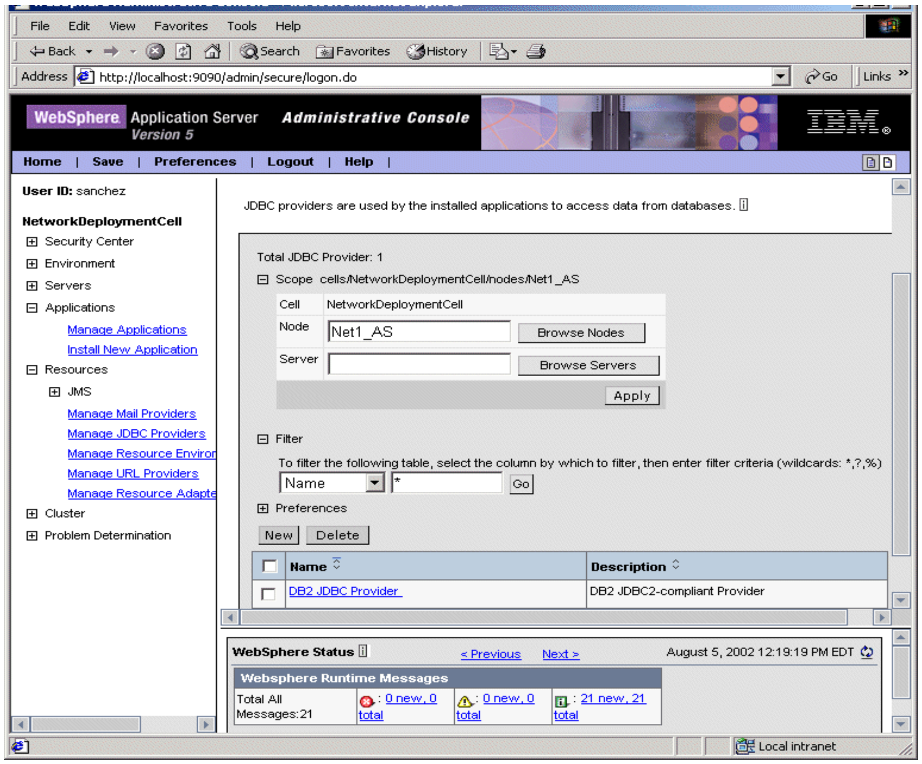
To locate and display items within a cell:

1. Select the associated task from the the Navigation tree. For example to locate an Application Server, select `Servers` -> `Manage Application Servers`.
2. Select the scope to a particular cell, node or server

3. Specify Preference settings to specify how you would like information to be displayed on the administrative console page

Select task

The Navigation tree on the left side of the console offers links to console pages that you use to create and manage components in a WebSphere administrative cell. The main actions available are listed in Section 13.3.2, “Navigation tree” on page 460. To use the console navigation tree, expand a tree item and select an action from the expanded item to display a page on which you can complete the needed work. For example, to create a JDBC provider you would expand `Resources` and then select the `Manage JDBC Providers` action.


File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |

User ID: sanchez

NetworkDeploymentCell
- ⊞ Security Center
- ⊞ Environment
- ⊞ Servers
- ⊟ Applications
  - Manage Applications
  - Install New Application
- ⊟ Resources
  - ⊞ JMS
  - Manage Mail Providers
  - Manage JDBC Providers
  - Manage Resource Enviror
  - Manage URL Providers
  - Manage Resource Adapte
- ⊞ Cluster
- ⊞ Problem Determination

JDBC providers are used by the installed applications to access data from databases.

Total JDBC Provider: 1

⊟ Scope cells/NetworkDeploymentCell/nodes/Net1_AS

Cell	NetworkDeploymentCell	
Node	Net1_AS	Browse Nodes
Server		Browse Servers
		Apply

⊟ Filter

To filter the following table, select the column by which to filter, then enter filter criteria (wildcards: *,?,%)

Name * Go

⊞ Preferences

New Delete

	Name	Description
☐	DB2 JDBC Provider	DB2 JDBC2-compliant Provider

WebSphere Status < Previous Next > August 5, 2002 12:19:19 PM EDT

Websphere Runtime Messages

Total All Messages: 21	: 0 new, 0 total	: 0 new, 0 total	: 21 new, 21 total

Figure 13-7 Working with the administrative console

Select a scope

After selecting an action, such as Manage JDBC Providers as shown in Figure 13-7, use the `Scope` settings to limit the scope of the information displayed in the administrative console to a particular cell, node, or server.

Click **Browse** next to a field to select the scope and click **Apply**.

Note: You have to click **Apply** for the collection table on the console page to refresh.

If you were to create a new object by clicking **New**, for example a new JDBC Provider, the Scope relates to its visibility from the application server's perspective. If you configure an object at the *Application server* scope, then only that application server can use that object. If you configure it at the *Node* scope, then all application servers running on that node can access it. Also if you configure an object at the *Cell* scope, then all managed servers in the cell may access that object.

The *Scope* setting is available for the *Resources*, *WebSphere Variables*, *Name Space Bindings* and *Shared Libraries* console pages.

Set preferences for viewing the console page

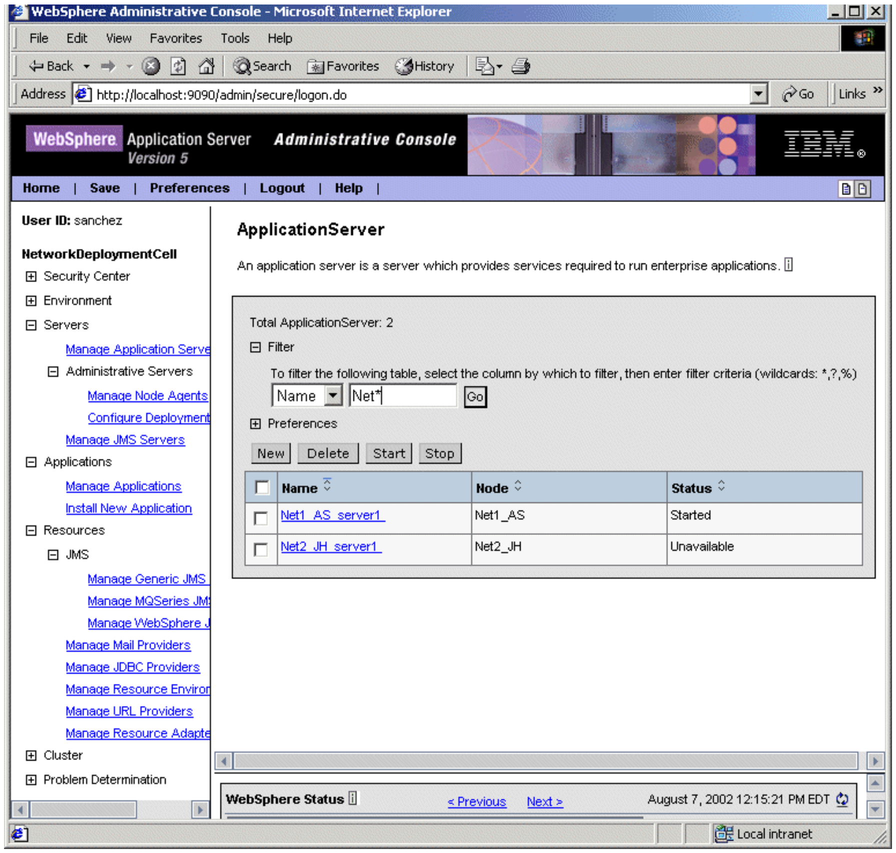
After selecting a task and a scope, the administrative console page shows a collection table with all the objects created at that particular scope. For example Figure 13-7 shows that there is only one JDBC Provider, `DB2 JDBC Provider`, created at the Node `Net1_AS` scope. All application servers running on node `Net1_AS` can access the `DB2 JDBC Provider`.

You can filter the contents of the administrative console collection table by using the **Filter** and **Preference** settings. For example, on Figure 13-8 the following task was selected: **Servers** -> **Manage Application Servers.** Then we used the Filter settings to display only those application servers whose name starts with Net.

Entries in the collection table can be filtered by:

- Name
- Server
- Status

The **Preferences** settings allows you to specify the maximun number of rows to display per page and whether to remember search criteria.



User ID: sanchez

ApplicationServer

An application server is a server which provides services required to run enterprise applications.

Total ApplicationServer: 2

To filter the following table, select the column by which to filter, then enter filter criteria (wildcards: *,?,%)

Name Net*

	Name	Node	Status
☐	Net1_AS_server1	Net1_AS	Started
☐	Net2_JH_server1	Net2_JH	Unavailable

August 7, 2002 12:15:21 PM EDT

Figure 13-8 Settings that affect how information is displayed on the admin console

13.4.2 Updating existing items

To edit the properties of an existing item, complete these tasks:

1. Find the item as described in Section 13.4.1, “Finding an item” on page 461.
2. Select it from the collection table by clicking on it.
3. In the **`Configuration`** page either specify new properties or edit the properties already configured for that item.

   The configurable properties will depend on the type of item selected.

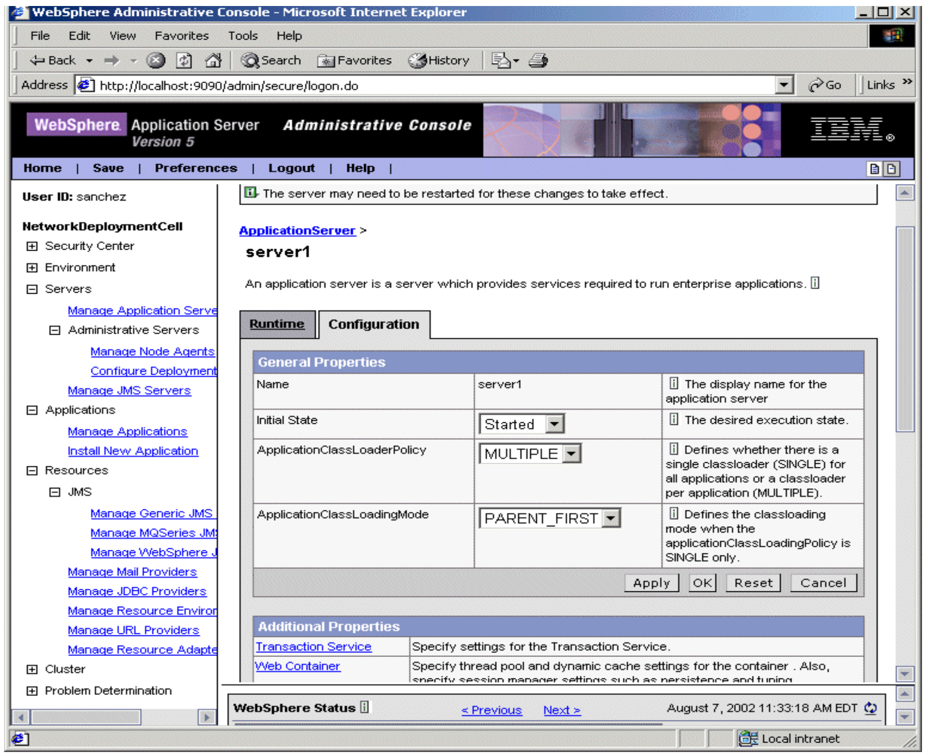
   For example, if we click on the Application Server `Net1_AS_server1` on node `Net1_AS` as shown in Figure 13-8, this will bring the console page for `Net1_AS_server1`, Figure 13-9. For this type of item the **`General Properties`** panel allows you to specify:

– Desired initial execution state (either started or stopped).

– Number of classloaders (one for all applications or one per each enterprise application running under the server).

– Classloading mode when a single classloader is specified, see Figure 13-9.

The **Additional Properties** panel allows configuration of additionbal properties relating to the object being edited. For example, you can configure properties for the Web Container, EJB Container, Dynamic Cache, etc, of an Application Server. For a full description of properties, see Chapter 15, “Server Configuration” on page 525.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Apply OK Reset Cancel
WebSphere Status
< Previous Next >
Local intranet

http://localhost:9090/admin/secure/logon.do

User ID: sanchez

NetworkDeploymentCell
- Security Center
- Environment
- Servers
  - Manage Application Serve
  - Administrative Servers
    - Manage Node Agents
    - Configure Deployment
  - Manage JMS Servers
- Applications
  - Manage Applications
  - Install New Application
- Resources
  - JMS
    - Manage Generic JMS
    - Manage MQSeries JM
    - Manage WebSphere J
  - Manage Mail Providers
  - Manage JDBC Providers
  - Manage Resource Enviror
  - Manage URL Providers
  - Manage Resource Adapte
- Cluster
- Problem Determination

The server may need to be restarted for these changes to take effect.

ApplicationServer >

server1

An application server is a server which provides services required to run enterprise applications.

Runtime | Configuration

General Properties

Name	server1	The display name for the application server
Initial State	Started	The desired execution state.
ApplicationClassLoaderPolicy	MULTIPLE	Defines whether there is a single classloader (SINGLE) for all applications or a classloader per application (MULTIPLE).
ApplicationClassLoadingMode	PARENT_FIRST	Defines the classloading mode when the applicationClassLoadingPolicy is SINGLE only.

Additional Properties

Transaction Service	Specify settings for the Transaction Service.
Web Container	Specify thread pool and dynamic cache settings for the container . Also, specify session manager settings such as persistence and tuning

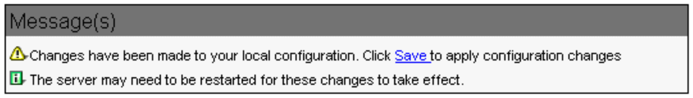
August 7, 2002 11:33:18 AM EDT

Figure 13-9 Configuration panel

4. Save changes after working on a page:

   a. Click **OK** to save your changes under your userid temporary workspace and exit the page, or **Apply** if you want to save the changes without exiting.

   b. Click **Save** when the message in Figure 13-10 appears at the top of your workspace.


Message(s)

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

The server may need to be restarted for these changes to take effect.

Figure 13-10 Save changes to the master repository

c. Click **Save** again

A new message box will be displayed asking you whether you want to:

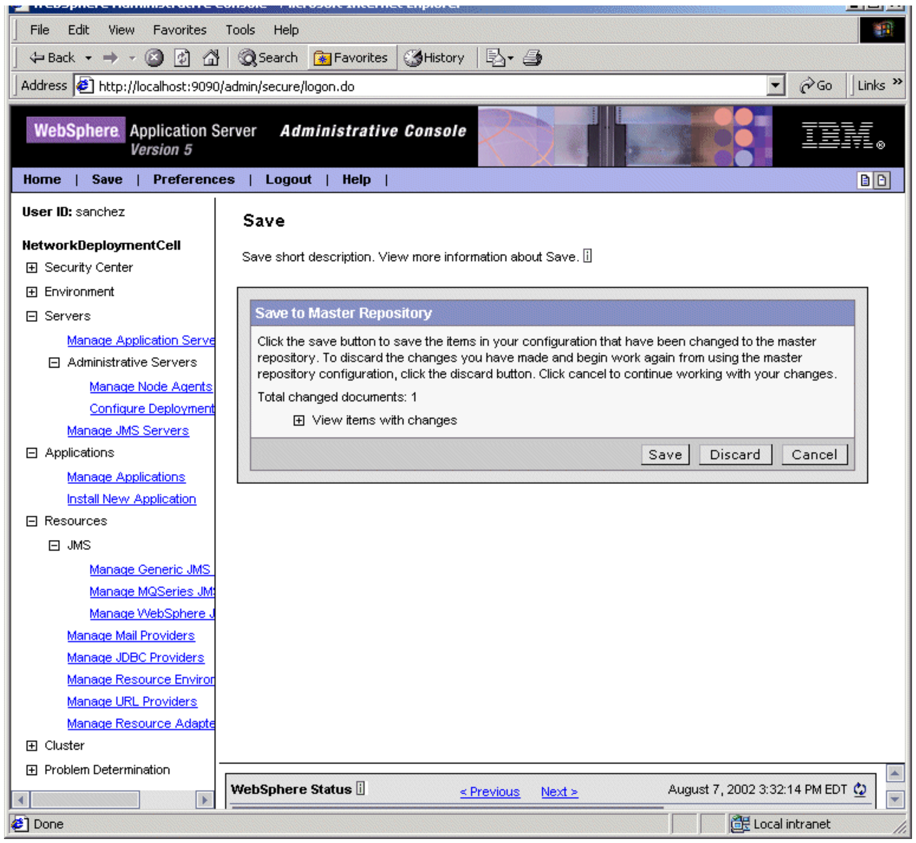
i. **Save** changes to the master repository,

ii. **Discard** the changes and start work using the master repository configuration, or

iii. **Cancel** to continue working with your changes.

Expand the View items in Figure 13-11, to see what items have been changed.

d. Click **Save**.



Save

Save short description. View more information about Save.

Save to Master Repository

Click the save button to save the items in your configuration that have been changed to the master repository. To discard the changes you have made and begin work again from using the master repository configuration, click the discard button. Click cancel to continue working with your changes.

Total changed documents: 1

View items with changes

Save Discard Cancel

Figure 13-11 Saving changes to the master repository

13.4.3 Adding new items

To create new instances of most item types, complete these tasks:

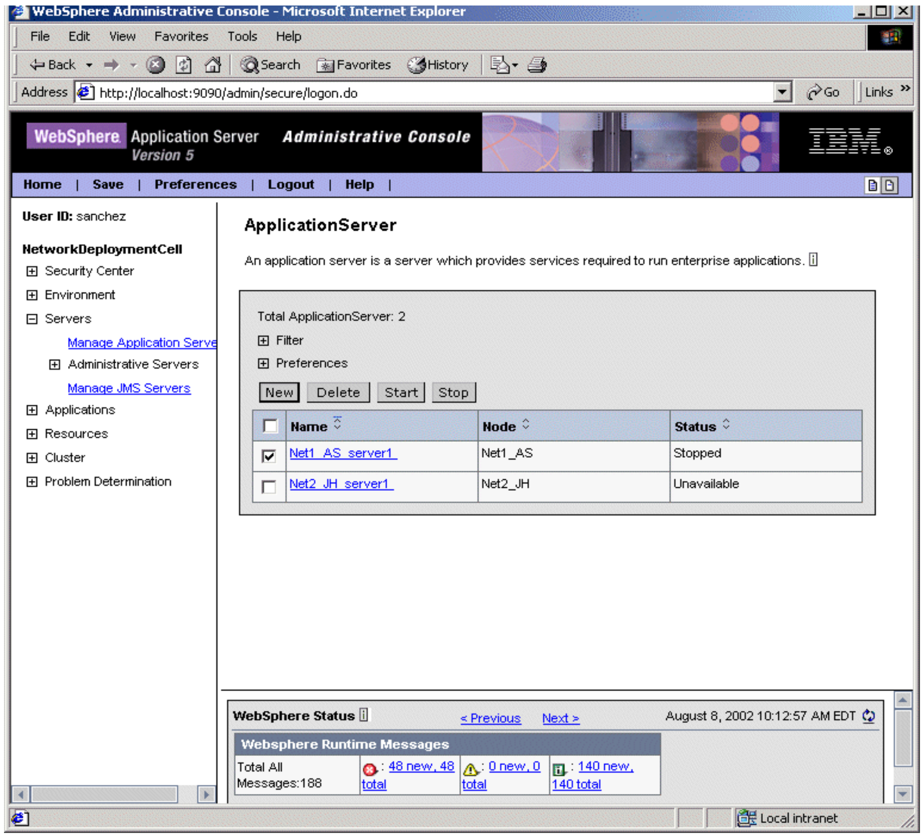
1. Find the item type as described in Section 13.4.1, “Finding an item” on page 461. Make sure you select the correct scope before creating it.

   As mentioned in Section 13.4.1, “Finding an item” on page 461, when creating an item, the Scope relates to its visibility from the application server’s perspective. If you configure an object at the *Application server* scope, then only that application server can use that object. If you configure it at the *Node scope*, then all application servers running on that node can access it. And if you configure an object at the *Cell* scope, then all application servers in the cell may access that object.

The Scope settings are available for the *Resources*, *WebSphere Variables*, *Name Space Bindings* and *Shared Libraries* console pages.

2. Select **New**.

For example, to create a new application server, select **Servers** -> **Manage Application Servers** -> **New**, as shown in Figure 13-12.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: sanchez
NetworkDeploymentCell
Security Center
Environment
Servers
Manage Application Serve
Administrative Servers
Manage JMS Servers
Applications
Resources
Cluster
Problem Determination
Filter
Preferences
New
Delete
Start
Stop
WebSphere Status
< Previous
Next >
Local intranet

ApplicationServer

An application server is a server which provides services required to run enterprise applications.

Total ApplicationServer: 2

	Name	Node	Status
☑	Net1_AS_server1	Net1_AS	Stopped
☐	Net2_JH_server1	Net2_JH	Unavailable

August 8, 2002 10:12:57 AM EDT

Websphere Runtime Messages

Total All Messages:188	: 48 new, 48 total	: 0 new, 0 total	: 140 new, 140 total

Figure 13-12 Creating, deleting, starting and stopping items

In general you will be presented with one or more **Configuration** pages in which you have to specify the item properties.

3. Click **Save** when finished, as shown in Figure 13-10.

13.4.4 Removing items

To remove an item, complete these tasks:

1. Find the item type as described in Section 13.4.1, “Finding an item” on page 461.
2. Select the item by selecting the checkbox next to it.
3. Click **`Delete.`**
4. If asked whether you want to delete it click **`OK`**.
5. Click **`Save`** to save the changes to the master repository, as shown in Figure 13-10.

For example, to delete an existing application server, select **`Servers`** -> **`Manage Application Servers`**. Place a check mark in the check box beside the application server you want to remove and click **`Delete`**, as shown in Figure 13-12.

13.4.5 Starting and stopping items

To start or stop an item, complete these tasks:

1. Find the item type as described in Section 13.4.1, “Finding an item” on page 461.
2. Select the item by selecting the checkbox next to it.
3. Click **`Start`** or **`Stop.`**

The collection table in the admin console will show the **`Status`** of the server as **`Started`** or **`Stopped`** depending on whether you selected to start or stop the server.

Note: The Status of the server can also show as Unavailable, as shown in Figure 13-12. This will happen when the node agent on the node in which the application server is installed is not active. In this case, the server cannot be started or stopped.

For example, to start an existing application server, select **`Servers`** -> **`Manage Application Servers.`** Place a check mark in the check box beside the application server you want started and click **`Start`**, as shown in Figure 13-12.

The following items can be started and/or stopped:

- **`Applications`**

  They can be started and stopped from the administrative console. Select **`Applications`** -> **`Manage Applications`**.

- **JMS Servers**

They can be both started and stopped from the administrative console. Select **`Servers`** -> **`Manage JMS Servers`**.

- **`Application Servers`**

  As for JMS Servers, application servers can be both started and stopped from the administrative console. Select **`Servers`** -> **`Manage Application Servers`**.

- **`Node Agents`**. They cannot be started nor stopped from the administrative console. To start and stop node agents, refer to Chapter 12, "System Management" on page 405.

- **Deployment manager Server (dmgr)**

  The deployment manager can be stopped from the administrative console as follows: select **`Servers`** -> **`Administrative Servers`** -> **`Configure Deployment Manager`** -> **`Configuration`** -> **`Stop`**.

  To restart it, you need to do it from the command line, Chapter 12, "System Management" on page 405.

  Since the deployment manager server, *dmgr*, is running the Administrative Console application, stopping the deployment manager from the administrative console will log you out of the current session. Login under the same user ID will allow you to save any changes made that were not published to the master configuration repository in the previous session.

  Note: Stopping the deployment manager server, dmgr, doesn't stop any of the node agents or the application servers running under those node agents.

13.4.6 Saving work

We have seen how to save changes to the master repository in Section 13.4.2, "Updating existing items" on page 464 .Consider the following points:

1. If you work on a page, and click **`Apply`** or **`OK`** , the changes will be saved under your userid temporary workspace. This will allow you to recover changes under the same user id if you exit the session without saving.
2. You need to click **`Save`** to save changes to the master repository. This can be done from the Taskbar, see Section 13.3.1, "Taskbar" on page 457, or when you log in, if you logged out without saving the changes, see Section 13.2.1, "Logging into the console" on page 453.
3. If you don't save changes to the master repository, the changes won't be pushed to your node's configuration repository. Effectively the new settings are lost. They are just available as configuration settings in your temporary workspace.

4. The Save screen presents you with three options:
   a. **Save**
   b. **Discard**

      Discard reverses any changes made during the working session and reverts to the master configuration.
   c. **Cancel**

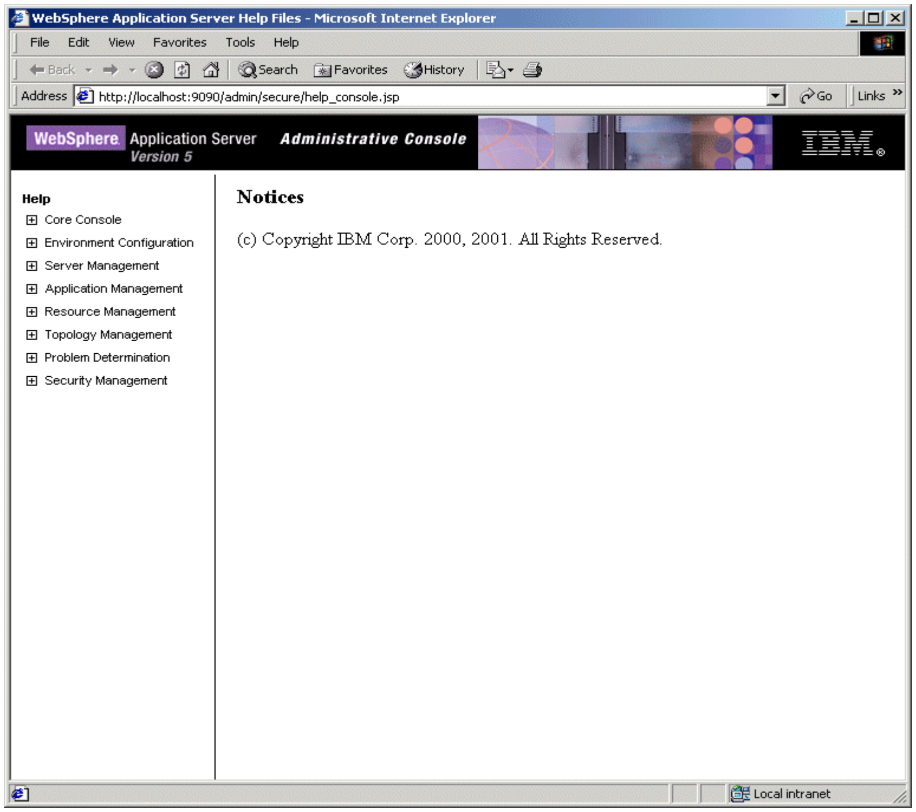
      Cancel doesn’t reverse changes made during the working session. It just cancels the action of saving to the master repository for now.
5. Before deciding whether you want to Save or Discard changes, you can see what changes will be saved by expanding **`View items with changes`** in the Save screen.

Important: All the changes made during a session are cumulative. Therefore when you decide to save changes to the master repository, either at log on or after clicking Save on the Taskbar, all changes will be committed. There is no way of being selective about what changes will get saved to the master repository.

13.4.7 Getting help

As described in the Section 13.3.1, “Taskbar” on page 457, help is accessible through:

1. The **Help** menu in the Taskbar. This Opens a new Web browser with online help for the WebSphere Application Server product, see Figure 13-13.


WebSphere Application Server Help Files - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/help_console.jsp
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Local intranet

Help
- Core Console
- Environment Configuration
- Server Management
- Application Management
- Resource Management
- Topology Management
- Problem Determination
- Security Management

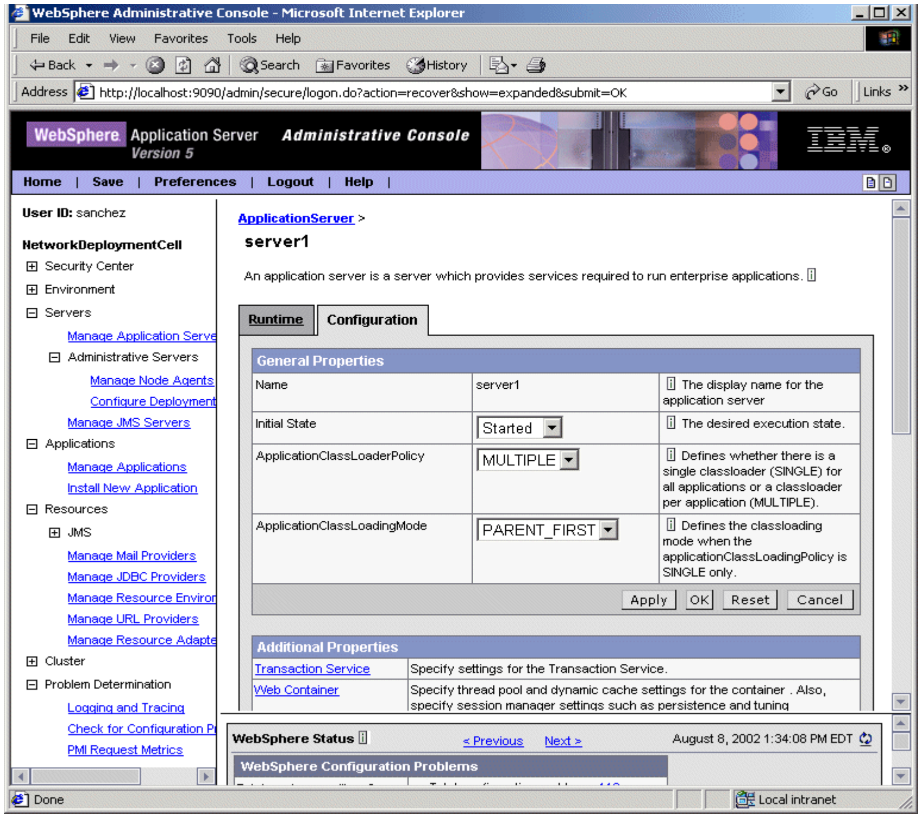
Notices

(c) Copyright IBM Corp. 2000, 2001. All Rights Reserved.

Figure 13-13 Online help

2. The **Hide Field and Page Descriptions toggle**. When disabled, see 13.3.1, “Taskbar” on page 457, console pages will show an “**i**” icon at the top of the workspace for page descriptions, and beside a field to see information just about that particular item. Click on it to access description information.

   For example, Figure 13-14, shows that there is description information available at the page level, `Application server settings`, and at the field level. This will just be a subset of the information contained at the page level.


WebSphere Administrative Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost:9090/admin/secure/logon.do?action=recover&show=expanded&submit=OK

WebSphere Application Server Version 5 Administrative Console

Home | Save | Preferences | Logout | Help |

User ID: sanchez

NetworkDeploymentCell
- Security Center
- Environment
- Servers
  - Manage Application Servers
  - Administrative Servers
    - Manage Node Agents
    - Configure Deployment
  - Manage JMS Servers
- Applications
  - Manage Applications
  - Install New Application
- Resources
  - JMS
  - Manage Mail Providers
  - Manage JDBC Providers
  - Manage Resource Environment
  - Manage URL Providers
  - Manage Resource Adapters
- Cluster
- Problem Determination
  - Logging and Tracing
  - Check for Configuration Problems
  - PMI Request Metrics

ApplicationServer >
server1

An application server is a server which provides services required to run enterprise applications.

Runtime | Configuration

General Properties

Name	server1	The display name for the application server
Initial State	Started	The desired execution state.
ApplicationClassLoaderPolicy	MULTIPLE	Defines whether there is a single classloader (SINGLE) for all applications or a classloader per application (MULTIPLE).
ApplicationClassLoadingMode	PARENT_FIRST	Defines the classloading mode when the applicationClassLoadingPolicy is SINGLE only.

Apply OK Reset Cancel

Additional Properties

Transaction Service	Specify settings for the Transaction Service.
Web Container	Specify thread pool and dynamic cache settings for the container . Also, specify session manager settings such as persistence and tuning

WebSphere Status < Previous Next > August 8, 2002 1:34:08 PM EDT

WebSphere Configuration Problems

Done Local intranet

Figure 13-14 Description information

3. The InfoCenter can be viewed online or downloaded from:

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

13.5 Stopping the administrative console

To stop the console, click **`Save`** on the console taskbar to save work that you have done and then click **`Logout`** to exit the console.

If you close the browser before saving your work, next time you log in under the same user ID, you can recover any unsaved changes, see “Logging into the console” on page 453 for more information.

13.6 Securing the administrative console

If you need to secure the administrative console, so only authenticated users can use it, complete the following steps:

Note: By doing this you are configuring authentication for all administrative services, that is the WebSphere administrative console and the administrative scripting interface, **`wsadmin`**.

1. Configure the User Registry that is to be used for user authentication:
   a. Expand **`Security`** -> **`User Registries`** from the admin console tree view.
   b. Select the type of user registry to be used for authentication:
      i. Local OS User Registry
      ii. LDAP User Registry
      iii. Custom User Registry

      Note: The local operating system user registry is intended for simple, non-distributed, single application server type runtime environments, ie. standalone setups. For Network Deployment environments (multiple nodes and multiple servers), users need to be authenticated against an LTPA based LDAP registry or custom registry.

   c. Configure the User Registry from the User Registry configuration page.
   d. Click **`OK`**.
   e. **`Save`** changes.

Author Comment: Refer to the Security chapter for information on how to configure the User Registry

2. Enable Global Security:

Author Comment: Refer to the Security chapter for what are the implications of doing this

a. Expand **Security Center** -> **Global Security**.

b. Select **Global Security**.

   From the Global Security console page ensure that the following properties are configured:

   i. Select the Security **Enabled** checkbox.

   ii. Specify the **Active Authentication Mechanism.** Possible values are SWAM (Simple WebSphere Authentication Mechanism) and LTPA (Lightweight Third Party Authentication).

   Note: Use LTPA for an LDAP or Custom User Registry. If you use SWAM, you can use any of type of registry (LocalOS, LDAP, or Custom), but you are limited to a standalone setup.

   iii. Specify the **Active User Registry**. Possible values are Local OS, LDAP and Custom.

   Note: The Active User Registry selected needs to match one of the existing user registries.

c. Click **OK**.

d. **Save** changes.

Author Comment: Refer to the Security chapter for more information on configuring Global Security

3. Configure console users and or groups to give users authority to administer WebSphere using the administrative console (and wsadmin):

   a. Expand **Security Center** -> **Manage Console Users** or **Security Center** -> **Manage Console Groups**.

   The admin roles available are:

   i. **Monitor**. Allows a user to view the WebSphere configuration and current state.

   ii. **Configurator**. Monitor privilege plus the ability to change the WebSphere configuration.

   iii. **Operator**. Monitor privilege plus the ability to change the runtime state, such as starting and stopping services.

iv. **Administrator**. Operator plus configurator properties.

Note: The user/group must be in the active user registry

b. Click `OK`.

c. `Save` changes.

4. Restart the Application Server (WebSphere Application Server) or the Network Deployment Manager (Network Deployment environment).

Note: In WebSphere NetWork Deployment, all the application servers will also need to be restarted.

5. Logback in on console. You should be now prompted for an user ID and Password for authentication.

Author Comment: In m0232.11 build this is not working properly

13.7 Common administrative tasks

Common administrative tasks performed using the WebShere administrative console include:

- Node
  - Add a node.
  - Remove a node.
  - Start a node agent.
  - Stop a node agent.
  - Adjust node agent synchronization.
- Application server
  - Create an application server.
  - Start an application server.
  - Stop an application server.
- Enterprise application
  - Install an enterprise application.
  - Uninstall an enterprise application.
  - Start an enterprise application.
  - Stop an enterprise application.
  - Export an enterprise application.

  - View installed applications.
  - View EJB modules.
  - View web modules.
  - Determine Servlet or JSP URL.
- Miscellaneous
  - Regenerate plug-in configuration file.
  - Check product versions.

This chapter will look at these tasks from the WebSphere administrative console viewpoint. For information about using the wsadmin tool to do the same tasks, see Chapter 23, “Command line administration and scripting” on page 971.

13.7.1 Nodes

Author Comment: Need intro text here ..

Adding a node

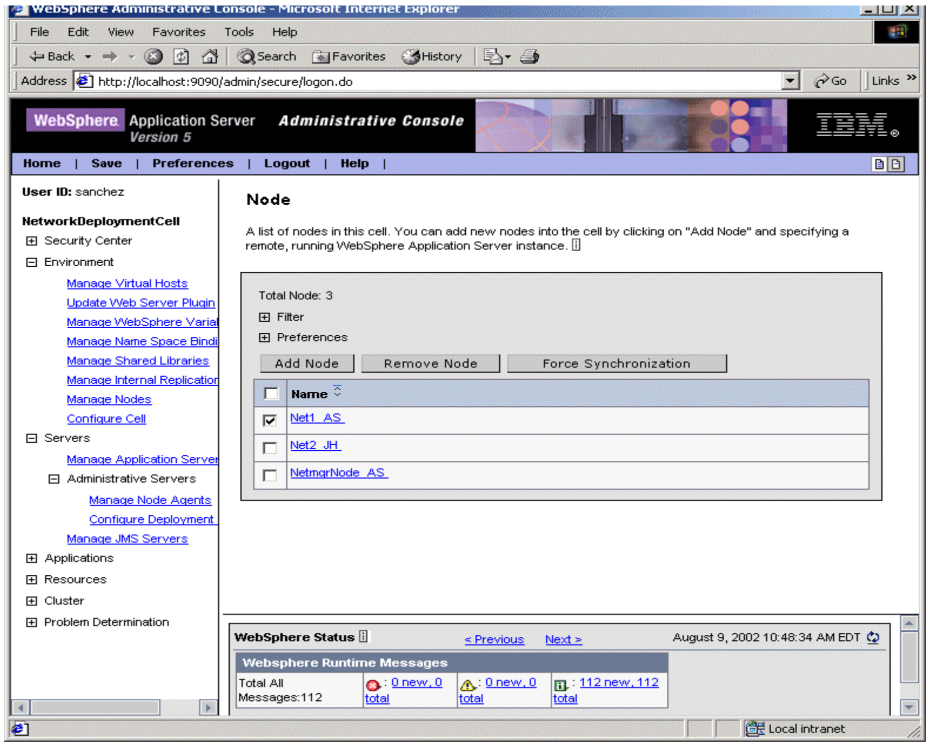
Author Comment: This is a dup of 12.8.1

There are two ways of incorporating a standalone application server installation (node) into an existing administration cell:

1. From the ND administrative console, as shown in Figure 13-15.

   Select **`System Administration`**-> **`Nodes`** -> **`Add Node`**

   Specify the hostname of the node to be added to the cell.



A list of nodes in this cell. You can add new nodes into the cell by clicking on "Add Node" and specifying a remote, running WebSphere Application Server instance.

	Name
✓	Net1_AS
	Net2_JH
	NetmgrNode_AS

Figure 13-15 Adding and removing a node from a cell using the administrative console

2. From the command line:

 a. Change directory to the *bin* directory of the standalone application server installation.

 b. Run **`addnode <dmgr host> <dmgr JMX port>`**. See “addNode” on page 1041 for information on this command.

 Example:

```
addnode deploymgr 8879
addnode deploymgr 9809 -conntype rmi
```

Note: By default **`addNode`** does not carry over applications from the standalone servers to the cell. So after adding a node you get a node agent,a JMSServer, any application servers and their configurations, but not the applications.

If you want the applications to be carried over from the servers on the node into the cell, use the addnode program with the “**`-includeapps`**” option.

Removing a node

There are two ways of removing a node from a network distributed administration cell:

1. From the Administrative Console, as shown in Figure 13-15.
   a. Select **System Administration** -> **Nodes**
   b. Place a check mark in the check box beside the node you want to remove and click **Remove Node.**
2. From the command line:
   a. Change directory to the *bin* directory of the base application server installation for that node.
   b. Run **removeNode**. See "removeNode" on page 1044 for information on this command.

Note: To avoid loss of configuration information when a node is detached from a cell, before removing the node modify the Application Servers as follows: Click **Servers** -> **Manage Application Servers** in the console navigation tree. Click an application server, and then click **Administrative Services -> Standalone**. This option is also available at the Node Agent level.

Adjusting node agent synchronization

Configuration synchronization between the node and the deployment manager is enabled by default. During a synchronization operation a node agent checks with the deployment manager to see if any configuration documents that apply to the node have been updated. New or updated documents are copied to the node repository, and deleted documents are removed from the node repository. The interval between synchronizations is configurable via the administrative console:

1. Expand **Servers** -> **Administrative Servers** in the administrative console.
2. Click **Manage Node Agents**.
3. In the Work Area pane, click on the Node.
4. Under **Additional Properties** click on **File Synchronization Service**.
5. Configure the **Synchronization Interval**. By default the synchronization interval is set to one minute.

Explicit synchronization can be forced by selecting **Environment** -> **Manage Nodes** -> **Select Node** -> **Force Synchronization** from the administrative console.

Tip: It is recommended to increase the synchronization interval in a production environment to reduce the overhead.

13.7.2 Application servers

This section describes how to perform common administration tasks on application servers using the WebShere administrative console.

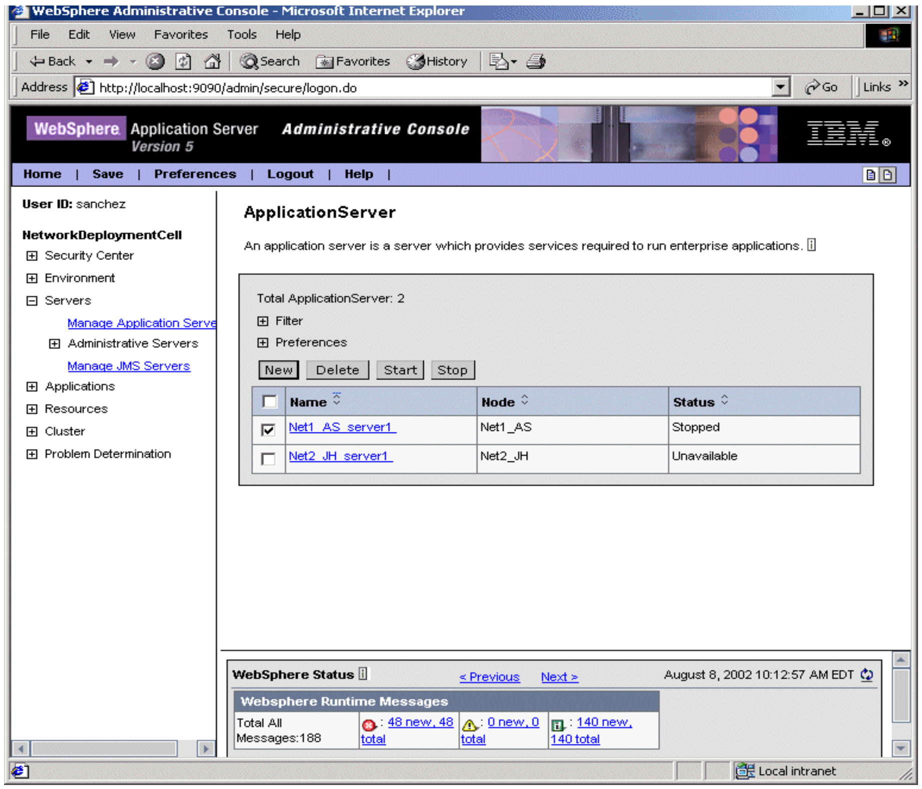
Author Comment: Any advice on when you need a new application server or when you should run multiple apps on one?

Creating an application server

In this section we provide an example of creating a new application server. The application server properties will be discussed in Chapter 15, “Server Configuration” on page 525.

You can create a new application server with the administrative console using the following steps:

1. From the console navigation tree select **`Servers`** -> **`Manage Application Servers`**.
2. In the admin console page select **`New`**, as shown in Figure 13-16.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: sanchez
NetworkDeploymentCell
Security Center
Environment
Servers
Manage Application Serve
Administrative Servers
Manage JMS Servers
Applications
Resources
Cluster
Problem Determination
New Delete Start Stop
Filter
Preferences
WebSphere Status
< Previous Next >
Websphere Runtime Messages
Local intranet

ApplicationServer

An application server is a server which provides services required to run enterprise applications.

Total ApplicationServer: 2

	Name	Node	Status
✓	Net1_AS_server1	Net1_AS	Stopped
	Net2_JH_server1	Net2_JH	Unavailable

August 8, 2002 10:12:57 AM EDT

Total All Messages:188 | : 48 new, 48 total | : 0 new, 0 total | : 140 new, 140 total

Figure 13-16 Creating a new application server

3. In the Create Application Server console page, Figure 13-17, you must specify the following properties:

   a. The node in which the application server is created.

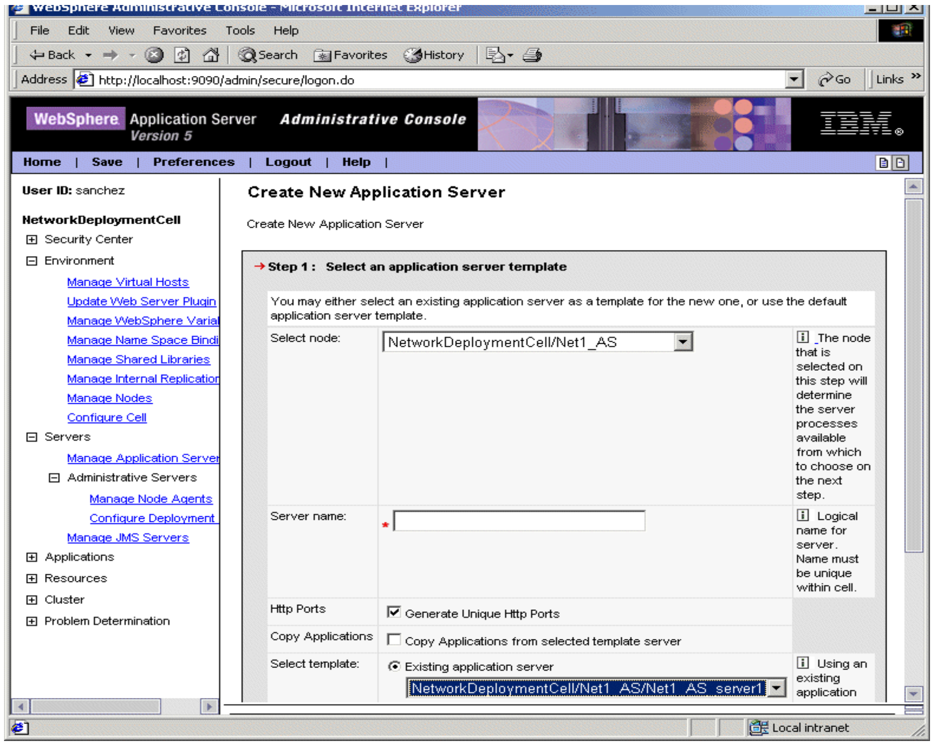
   b. A name for the new application server. This must be unique within the node.

   c. A template to be used for the new application server. You may select an existing application server, or use the default application server.

   d. Leave the Generate Unique Http Ports option selected.

      WebSphere maintains a counter for HTTP transport ports. The counter starts at port 9080 for HTTP and 9443 HTTPS. When this option is selected, and a new application server is created, WebSphere assigns the next port number available to the Web container HTTP transports. For example 9081 for HTTP and 9444 for HTTPS.

> **Note:** Port numbers must be unique for each application server instance on a given machine. Refer to Appendix A, “IP Ports used by WebSphere” on page 17, for more information on ports used by WebSphere.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Local intranet

User ID: sanchez

NetworkDeploymentCell
- Security Center
- Environment
  - Manage Virtual Hosts
  - Update Web Server Plugin
  - Manage WebSphere Variał
  - Manage Name Space Bindi
  - Manage Shared Libraries
  - Manage Internal Replicatior
  - Manage Nodes
  - Configure Cell
- Servers
  - Manage Application Server
  - Administrative Servers
    - Manage Node Agents
    - Configure Deployment
  - Manage JMS Servers
- Applications
- Resources
- Cluster
- Problem Determination

Create New Application Server

Create New Application Server

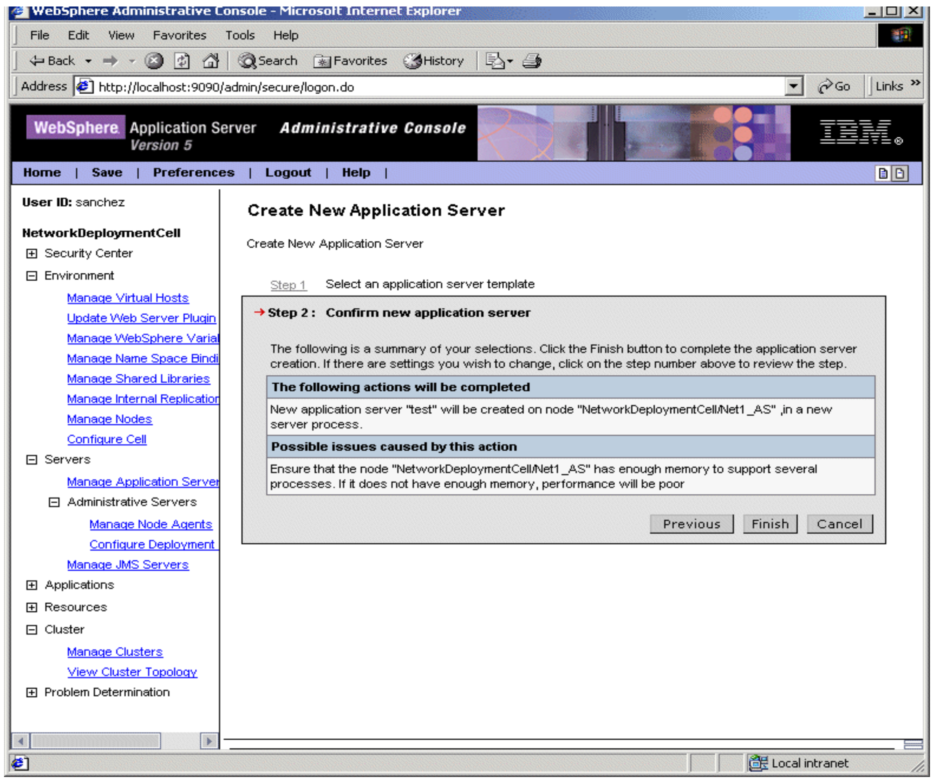
Step 1 : Select an application server template

You may either select an existing application server as a template for the new one, or use the default application server template.

Select node:	NetworkDeploymentCell/Net1_AS	The node that is selected on this step will determine the server processes available from which to choose on the next step.
Server name:	*	Logical name for server. Name must be unique within cell.
Http Ports	[✓] Generate Unique Http Ports	
Copy Applications	[ ] Copy Applications from selected template server	
Select template:	(●) Existing application server NetworkDeploymentCell/Net1_AS/Net1_AS_server1	Using an existing application

Figure 13-17 Application Server properties

4. Confirm creation of the new application server by clicking **`Finish`** in the next step, as shown in Figure 13-18.
5. Click **`Save`**, to save the changes.



Create New Application Server

Create New Application Server

Step 1 Select an application server template

→ Step 2 : Confirm new application server

The following is a summary of your selections. Click the Finish button to complete the application server creation. If there are settings you wish to change, click on the step number above to review the step.

The following actions will be completed

New application server "test" will be created on node "NetworkDeploymentCell/Net1_AS" ,in a new server process.

Possible issues caused by this action

Ensure that the node "NetworkDeploymentCell/Net1_AS" has enough memory to support several processes. If it does not have enough memory, performance will be poor

Figure 13-18 Application server creation

Note: Whether you used the default or an existing application server as a template, the application server just created has many default values specified for it.

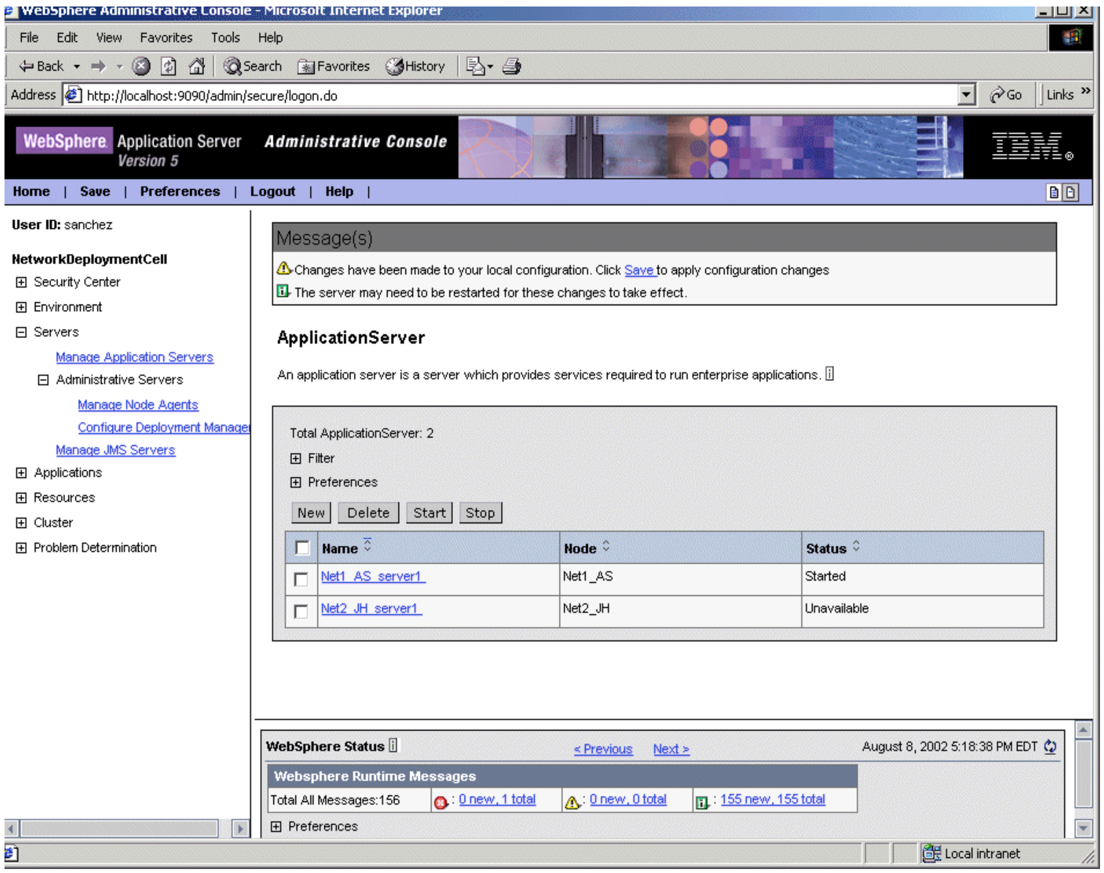
To view all the properties of your application server, select **`Servers`** -> **`Manage Application Servers`**. Place a check mark in the check box beside the application server you want to edit and click **`Properties`**. Change the settings for your application server as needed.

Application server log files are kept under the local node configuration. The location is configurable via the **`View logs and trace`** property -> **`JVM logs`**. By default this is the logs/<appserver> subdirectory of the base application server installation for that node.

Starting an application server

An application server can be started using one of these options:

1. From the administrative console:
   a. Select **`Servers`** -> **`Manage Application Servers`**.
   b. Place a check mark in the check box beside the application server you want started and click **`Start`**, as shown in Figure 13-19.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: sanchez
NetworkDeploymentCell
Security Center
Environment
Servers
Manage Application Servers
Administrative Servers
Manage Node Agents
Configure Deployment Manager
Manage JMS Servers
Applications
Resources
Cluster
Problem Determination

Message(s)

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

The server may need to be restarted for these changes to take effect.

ApplicationServer

An application server is a server which provides services required to run enterprise applications.

Total ApplicationServer: 2

Filter

Preferences

New Delete Start Stop

	Name	Node	Status
☐	Net1_AS_server1	Net1_AS	Started
☐	Net2_JH_server1	Net2_JH	Unavailable

WebSphere Status < Previous Next > August 8, 2002 5:18:38 PM EDT

Websphere Runtime Messages

Total All Messages:156	: 0 new, 1 total	: 0 new, 0 total	: 155 new, 155 total

Preferences

Figure 13-19 Starting and stopping the application server from the administrative console

2. From the command line:
   a. Change directory to the *bin* directory of the base application server installation for that node.
   b. Run **`startServer`** <appserver name>. See Chapter 12, “System Management” on page 405 for information on this command.

If there are any errors check the log file for the application server process:

```
<WAS_ROOT>/logs/app_server_namer/SystemOut.log
```

Note: In order to start an application server, the node agent on the node must be active. This is because the node agent hosts the Location Service Daemon that is needed by the application server.

You can configure the desired state of the application server when the process that manages it, the Node Agent, starts. To specify the desired state, select **`Servers`** -> **`Manage Application Server.`** Place a check mark in the check box beside the application server you want to configure and click **`Initial State`**. The options are *Started* and *Stopped*.

Note: When the application server starts all the applications running under that application server will be started.

Stopping an application server

An application server can be stopped using one of these options:

1. From the Administrative Console:
   a. Select **`Servers`** -> **`Manage Application Servers`**
   b. Place a check mark in the check box beside the application server you want to stop and click **`Stop`**, as shown in Figure 13-19.
2. From the command line:
   a. Change directory to the *bin* directory of the base application server installation for that node.
   b. Run **`stopServer`** <appserver name>. See Chapter 12, “System Management” on page 405 for information on this command.

   If there are any errors check the log file for the application server process:

```
<WAS_ROOT>/logs/app_server_name/SystemOut.log
```

Note: If you stop the application server all applications served by that application server will become unavailable.

13.7.3 Enterprise applications

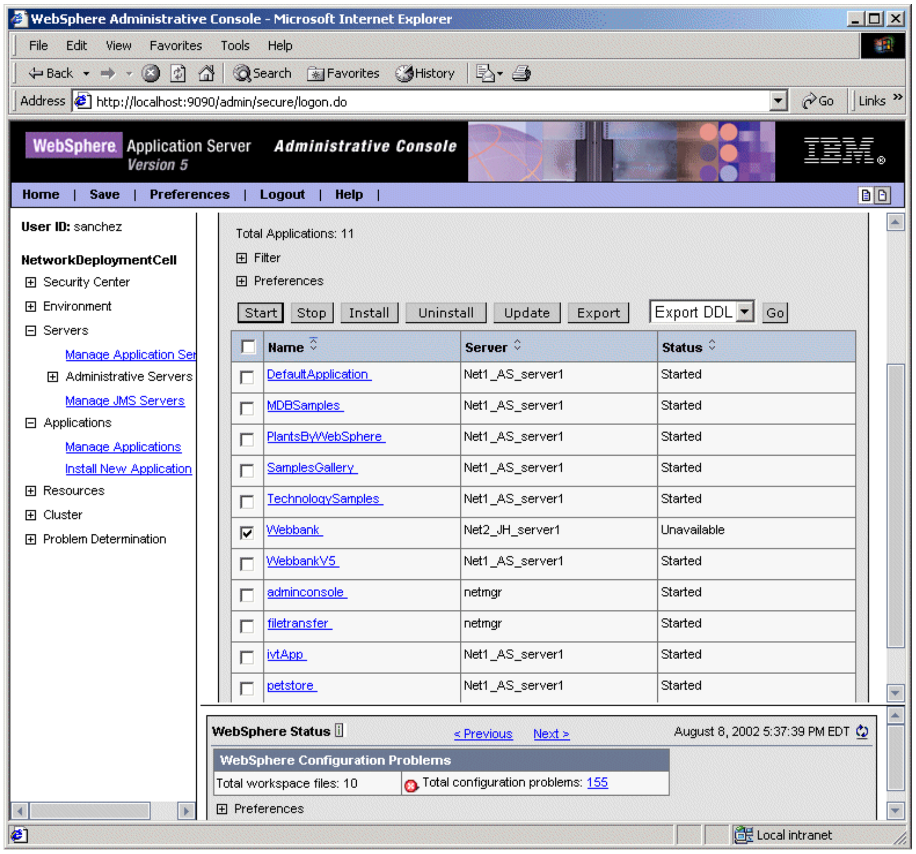
This section describes how to perform common administration tasks on enterprise applications using the WebSphere web admin console. To do the same using the scripting interface, **`wsadmin`**, refer to Chapter 19, “Scripting” on page 339.

Installing an enterprise application

To install an enterprise application into a WebSphere configuration you must install its modules onto one or more application servers. The steps for this task are as follows:

1. Select **`Applications`** -> **`Manage Applications`** -> **`Install`**, from the console navigation tree, see Figure 13-20.

   Or select **`Applications`** -> **`Install New Application`**



WebSphere Administrative Console - Microsoft Internet Explorer

Address http://localhost:9090/admin/secure/logon.do

Home | Save | Preferences | Logout | Help |

User ID: sanchez

NetworkDeploymentCell
- Security Center
- Environment
- Servers
  - Manage Application Ser
  - Administrative Servers
  - Manage JMS Servers
- Applications
  - Manage Applications
  - Install New Application
- Resources
- Cluster
- Problem Determination

Total Applications: 11

Filter

Preferences

Start | Stop | Install | Uninstall | Update | Export | Export DDL | Go

	Name	Server	Status
☐	DefaultApplication	Net1_AS_server1	Started
☐	MDBSamples	Net1_AS_server1	Started
☐	PlantsByWebSphere	Net1_AS_server1	Started
☐	SamplesGallery	Net1_AS_server1	Started
☐	TechnologySamples	Net1_AS_server1	Started
✓	Webbank	Net2_JH_server1	Unavailable
☐	WebbankV5	Net1_AS_server1	Started
☐	adminconsole	netmgr	Started
☐	filetransfer	netmgr	Started
☐	ivtApp	Net1_AS_server1	Started
☐	petstore	Net1_AS_server1	Started

WebSphere Status < Previous Next > August 8, 2002 5:37:39 PM EDT

WebSphere Configuration Problems

Total workspace files: 10	Total configuration problems: 155

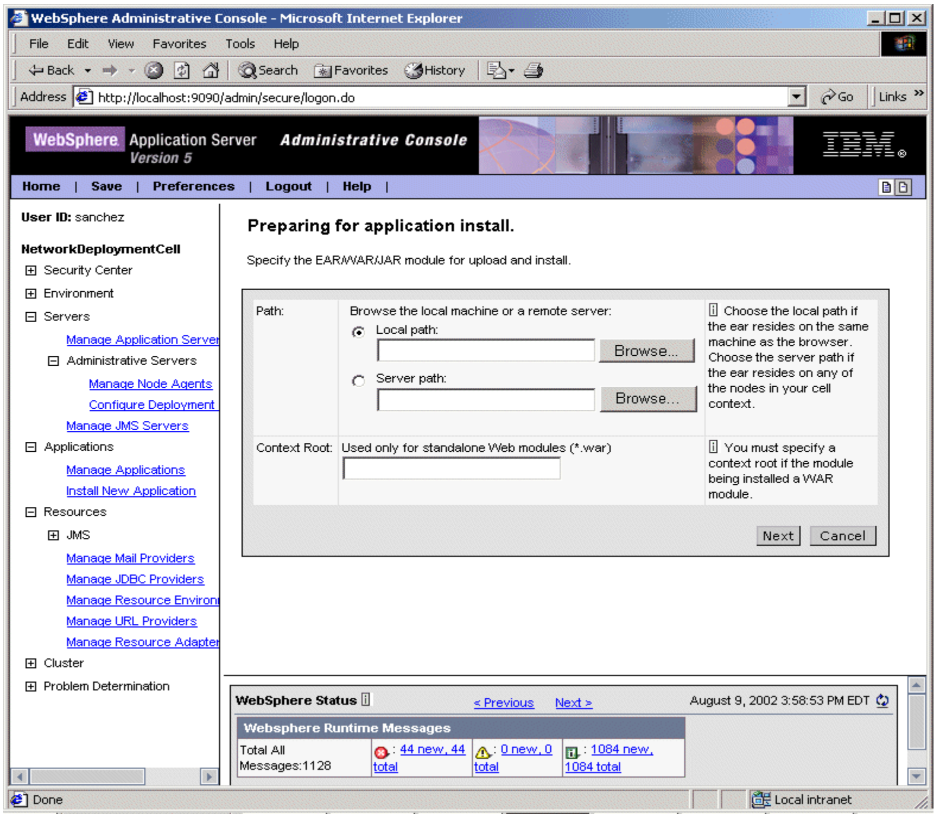
Preferences

Local intranet

Figure 13-20 Installing, uninstalling, updating and exporting an enterprise application

2. Specify the location of the EAR file to install, as shown in Figure 13-21.

   The EAR file that you are installing can be either on the client machine (the machine that runs the Web browser) or on any of the nodes in the cell.



Preparing for application install.

Specify the EAR/WAR/JAR module for upload and install.

Path:	Browse the local machine or a remote server: (●) Local path: [ ] ( ) Server path: [ ]	Choose the local path if the ear resides on the same machine as the browser. Choose the server path if the ear resides on any of the nodes in your cell context.
Context Root:	Used only for standalone Web modules (*.war) [ ]	You must specify a context root if the module being installed a WAR module.

WebSphere Status < Previous Next > August 9, 2002 3:58:53 PM EDT

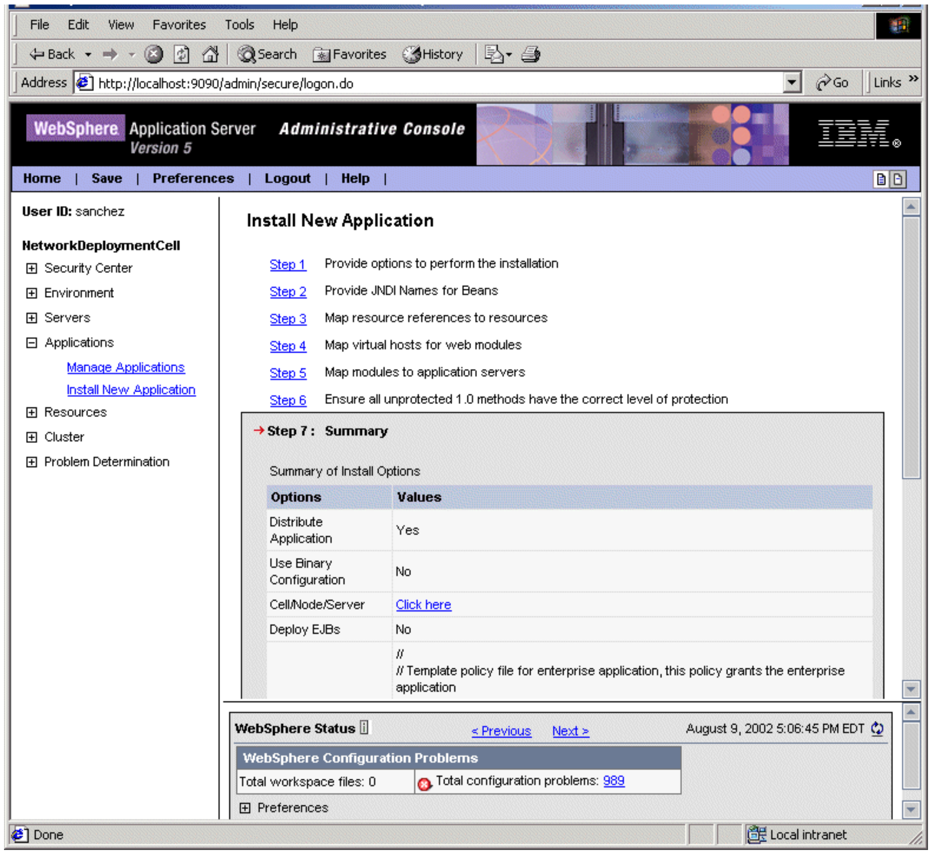
Websphere Runtime Messages

Total All Messages:1128	: 44 new, 44 total	: 0 new, 0 total	: 1084 new, 1084 total

Figure 13-21 Installing an enterprise application

3. Follow the steps in the Install New Application wizard to finish the install. The installation steps are, Figure 13-22:
   a. Provide options to perform the installation.
   b. Provide JNDI names for beans.
   c. Map resource references to resources.
   d. Map virtual hosts for web modules.
   e. Map modules to application servers.
   f. Ensure all unprotected 1.0 methods have the correct level of protection.
   g. Summary.

Author Comment: There can be different pages and number of pages displayed, depending upon the features indicated in the application’s deployment descriptors.


File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: sanchez
NetworkDeploymentCell
Security Center
Environment
Servers
Applications
Manage Applications
Install New Application
Resources
Cluster
Problem Determination
Done
Local intranet

Install New Application

Step 1	Provide options to perform the installation
Step 2	Provide JNDI Names for Beans
Step 3	Map resource references to resources
Step 4	Map virtual hosts for web modules
Step 5	Map modules to application servers
Step 6	Ensure all unprotected 1.0 methods have the correct level of protection

→ Step 7 : Summary

Summary of Install Options

Options	Values
Distribute Application	Yes
Use Binary Configuration	No
Cell/Node/Server	Click here
Deploy EJBs	No
	// // Template policy file for enterprise application, this policy grants the enterprise application

WebSphere Status < Previous Next > August 9, 2002 5:06:45 PM EDT

WebSphere Configuration Problems	
Total workspace files: 0	Total configuration problems: 989

Preferences

Figure 13-22 Steps for installing an application

Note: When you install an application via the administrative console, the files comprising the application, binaries and configuration files, both get placed on the network deployment node and on the node in which the application server is running. The default location is:

```
binaries: <WAS_ROOT>/config/app
configuration: <WAS_ROOT>/config/cells/<cellname>/applications/<appname>.ear
```

Save the configuration before attempting to start the application.

For detailed information on installation applications, see Chapter 20, “Deploying an application” on page 835.

Uninstalling an enterprise application

To uninstall an enterprise application that is no longer needed do the following:

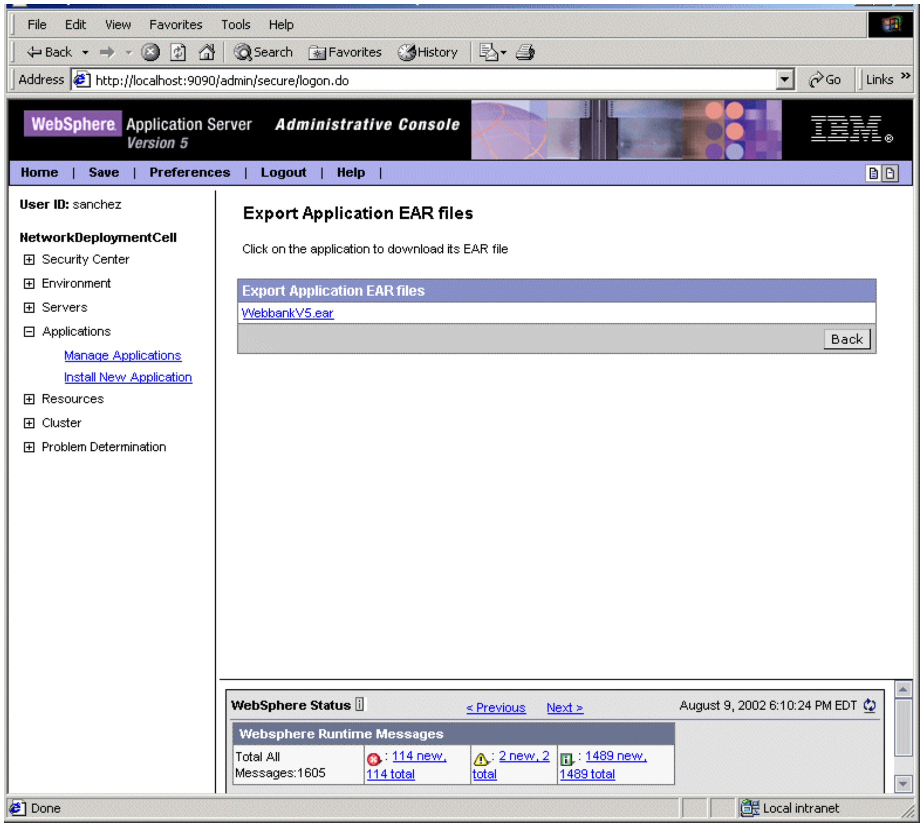
1. From the console navigation tree, see Figure 13-20, select **`Applications`** -> **`Manage Applications`**.
2. Place a check mark in the check box beside the appplication you want uninstall and click **`Uninstall`**.

Note: Uninstalling an application deletes the application binaries and configuration files from the network deployment node and from the node in which the application was deployed.

Exporting an enterprise 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:

1. From the console navigation tree, see Figure 13-20, select **`Applications`** -> **`Manage Applications`**.
2. Place a check mark in the check box beside the application you want to export and click **`Export`**.
3. On the Export Application EAR Files page, Figure 13-23, click on the link for the file you want to export.


File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: sanchez
NetworkDeploymentCell
Security Center
Environment
Servers
Applications
Manage Applications
Install New Application
Resources
Cluster
Problem Determination
Back
WebSphere Status
< Previous Next >
August 9, 2002 6:10:24 PM EDT
Done
Local intranet

Export Application EAR files

Click on the application to download its EAR file

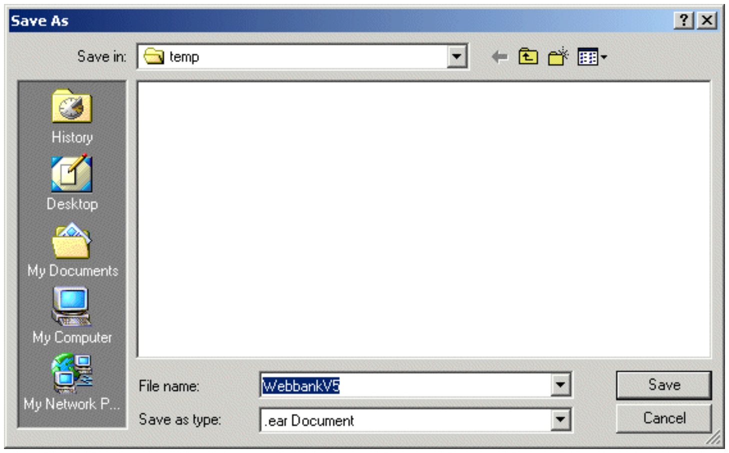
Export Application EAR files
WebbankV5.ear

Websphere Runtime Messages

Total All Messages:1605	: 114 new, 114 total	: 2 new, 2 total	: 1489 new, 1489 total

Figure 13-23 Exporting an enterprise application to an EAR file

4. Specify the directory on the local machine where the export to be saved and click **OK**, as shown in Figure 13-24.



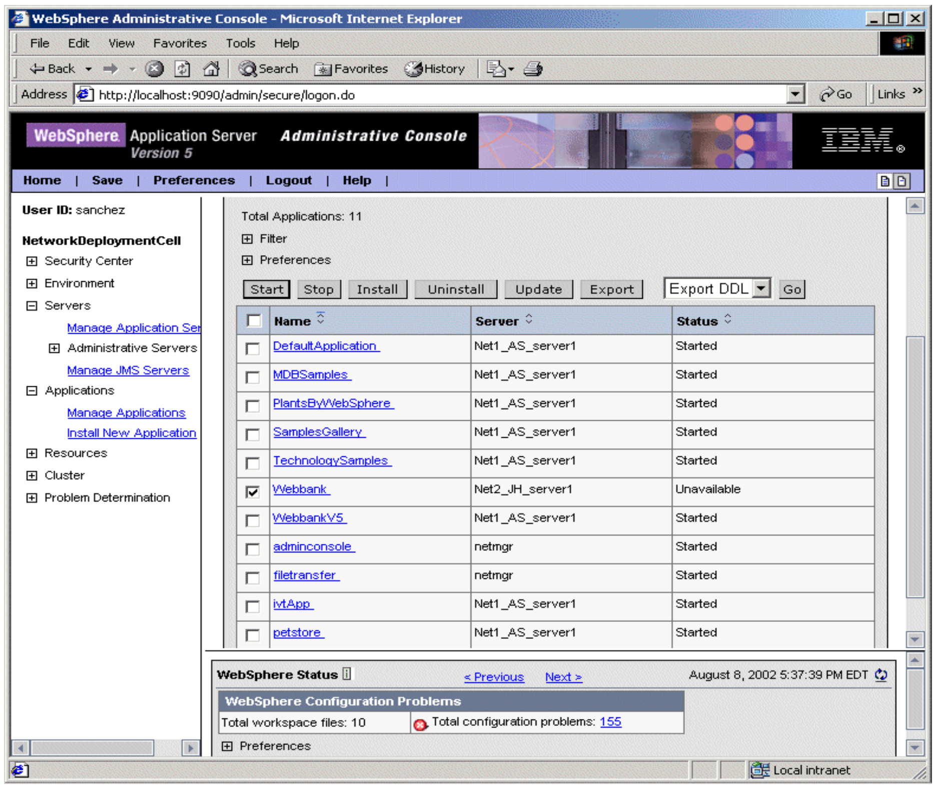
File name: WebbankV5
Save as type: .ear Document

Figure 13-24 Saving the ear file

Starting an enterprise application

An application can be started using one of these options:

1. From the administrative console.
   a. Select **`Applications`** -> **`Manage Applications`**.
   b. Place a check mark in the check box beside the application you want started and click **`Start`**, as shown in Figure 13-25.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: sanchez
NetworkDeploymentCell
Security Center
Environment
Servers
Manage Application Ser
Administrative Servers
Manage JMS Servers
Applications
Manage Applications
Install New Application
Resources
Cluster
Problem Determination
Total Applications: 11
Filter
Preferences
Start Stop Install Uninstall Update Export Export DDL Go
WebSphere Status
< Previous Next >
August 8, 2002 5:37:39 PM EDT
WebSphere Configuration Problems
Total workspace files: 10
Total configuration problems: 155
Preferences
Local intranet

	Name	Server	Status
☐	DefaultApplication	Net1_AS_server1	Started
☐	MDBSamples	Net1_AS_server1	Started
☐	PlantsByWebSphere	Net1_AS_server1	Started
☐	SamplesGallery	Net1_AS_server1	Started
☐	TechnologySamples	Net1_AS_server1	Started
☑	Webbank	Net2_JH_server1	Unavailable
☐	WebbankV5	Net1_AS_server1	Started
☐	adminconsole	netmgr	Started
☐	filetransfer	netmgr	Started
☐	ivtApp	Net1_AS_server1	Started
☐	petstore	Net1_AS_server1	Started

Figure 13-25 Starting and stopping an application from the administrative console

Note: In order to start an application, the application server that contains the application has to be started. If not, the application will show in the administrative console as unavailable and you won't be able to start it.

There is no command line option to start an application, and there is no *Initial State* setting either. The application will always be started when its Application Server starts.

Stopping an enterprise application

An application can be stopped using one of these options:

1. From the Administrative Console.
   a. Select **`Applications`** -> **`Manage Applications`**
   b. Place a check mark in the check box beside the application you want to stop and click **`Stop`**, as shown in Figure 13-25.

Viewing installed applications

The administrative console does not display the deployed servlets, JSPs or EJBs directly on the console. However you can use the console to display XML deployment descriptors for the enterprise application, Web modules and EJB modules.

To see the WAR files and JAR files associated with an enterprise application do the following:

1. From the console navigation tree, see Figure 13-20, select **`Applications`** -> **`Manage Applications`**.
2. Click on the application that you are interested in.
3. Under **`Configuration`** -> **`Additional Properties`**, select **`View Deployment Descriptor`**.

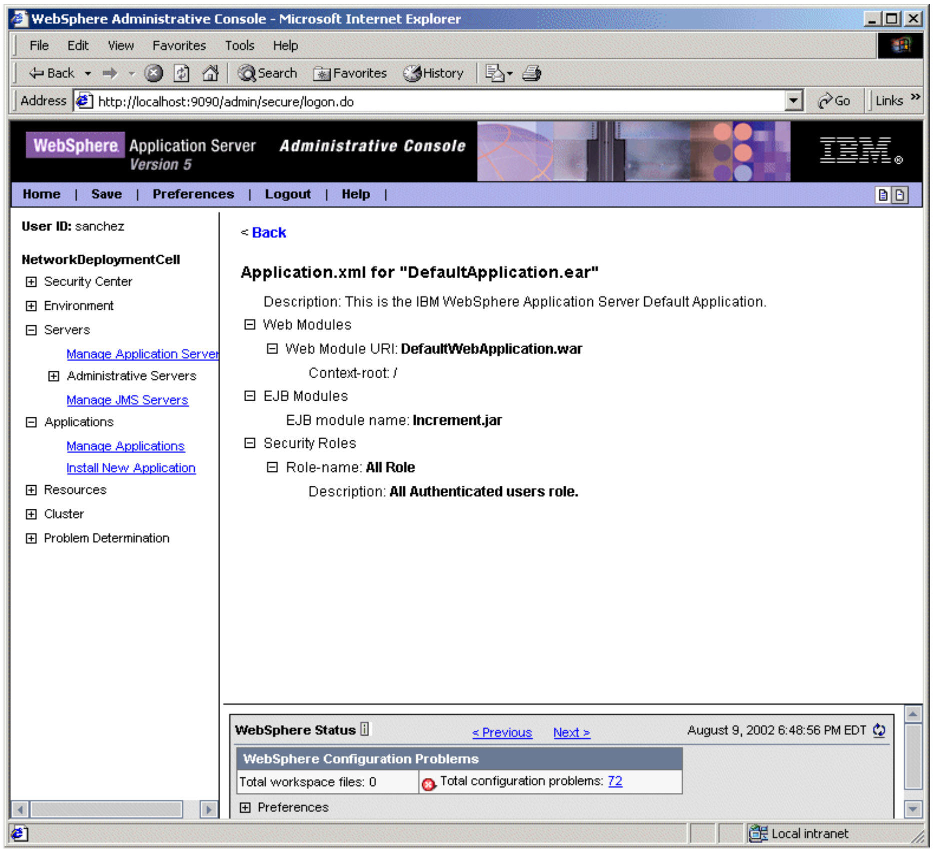
Figure 13-26 shows the Deployment Descriptor window for the *DefaultApplication* enterprise application. You can see:

a. The name and description of the enterprise application.

b. The Web modules or WAR files and their context roots.

c. The EJB modules and their associated JAR files.

d. The Security roles associated with the enterprise application.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: sanchez
NetworkDeploymentCell
Security Center
Environment
Servers
Manage Application Server
Administrative Servers
Manage JMS Servers
Applications
Manage Applications
Install New Application
Resources
Cluster
Problem Determination
< Back

Application.xml for "DefaultApplication.ear"

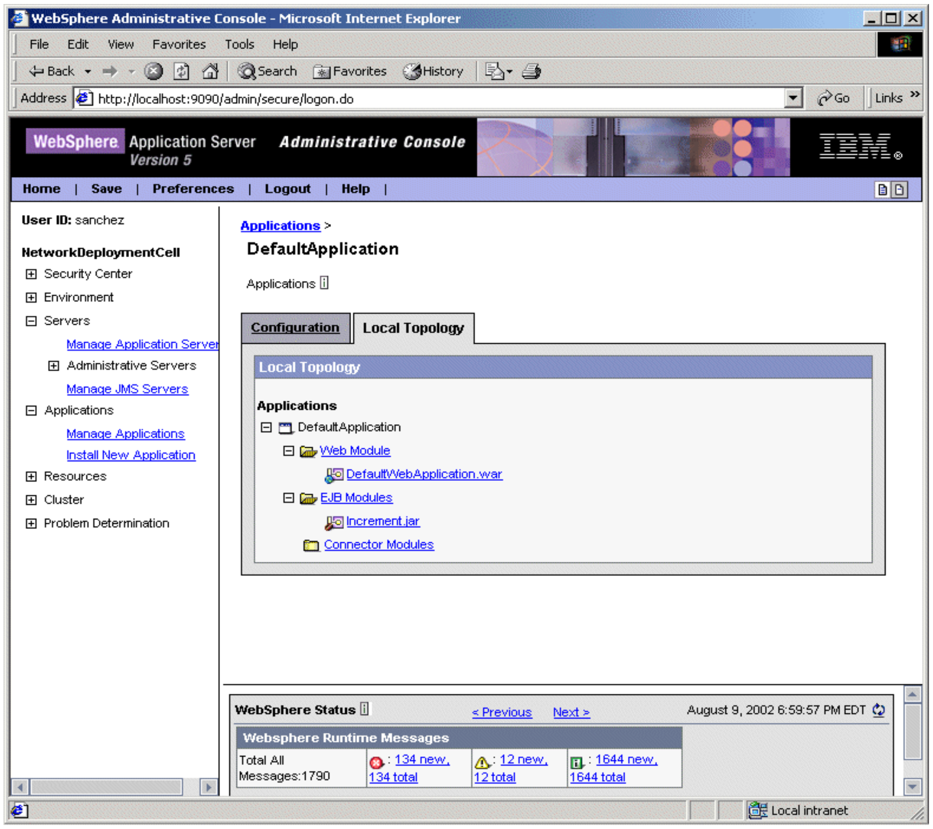
Description: This is the IBM WebSphere Application Server Default Application.
Web Modules
Web Module URI: **DefaultWebApplication.war**
Context-root: /
EJB Modules
EJB module name: **Increment.jar**
Security Roles
Role-name: **All Role**
Description: **All Authenticated users role.**

WebSphere Status < Previous Next > August 9, 2002 6:48:56 PM EDT
WebSphere Configuration Problems
Total workspace files: 0 Total configuration problems: 72
Preferences

Figure 13-26 Enterprise application deployment descriptor

4. The **`Local Topology`** tab in the administrative console page for this application also gives a graphical view of it, as shown in Figure 13-27. From this view you can access information for all the modules comprising the application.

Note: Local topology views are also available at the Node and Cell levels. Select **`Environment`** and **`Manage Nodes`**, **`Configure Cell`** respectively.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |

User ID: sanchez

NetworkDeploymentCell
- Security Center
- Environment
- Servers
  - Manage Application Server
  - Administrative Servers
  - Manage JMS Servers
- Applications
  - Manage Applications
  - Install New Application
- Resources
- Cluster
- Problem Determination

Applications >
DefaultApplication

Applications

Configuration | Local Topology

Local Topology

Applications
- DefaultApplication
  - Web Module
    - DefaultWebApplication.war
  - EJB Modules
    - Increment.jar
  - Connector Modules

WebSphere Status < Previous Next > August 9, 2002 6:59:57 PM EDT

Websphere Runtime Messages

Total All Messages:1790	: 134 new, 134 total	: 12 new, 12 total	: 1644 new, 1644 total

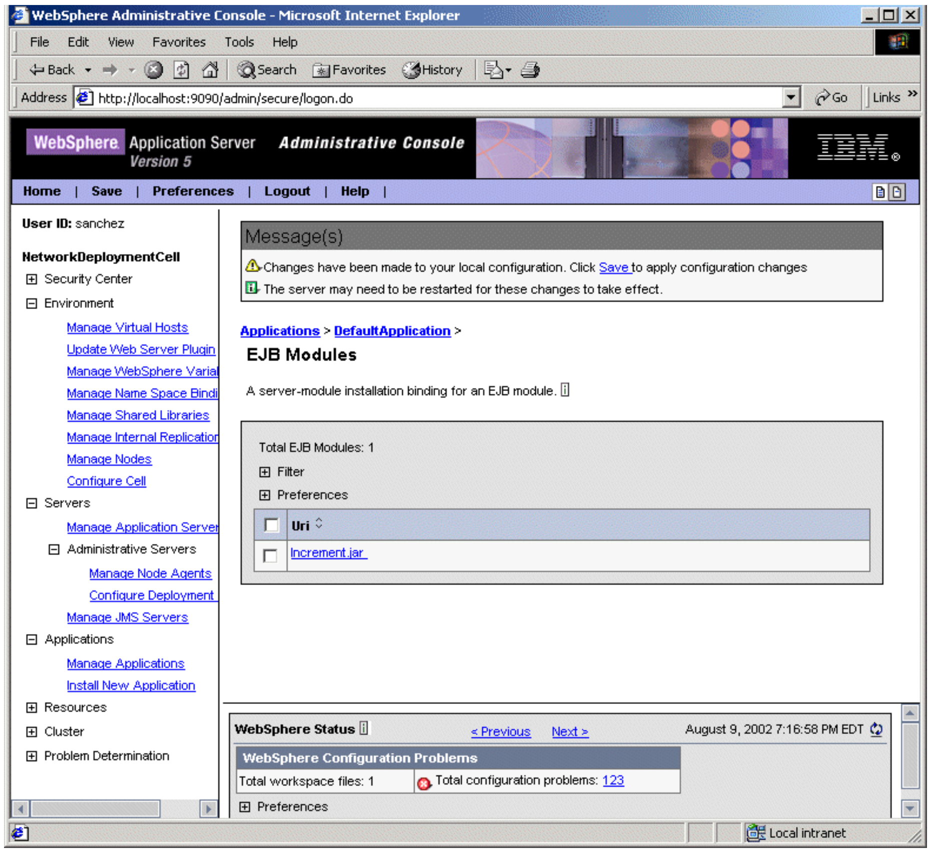
Figure 13-27 Topology view of the application

Viewing EJB modules

To see the EJBs that are part of an enterprise application:

1. From the console navigation tree, see Figure 13-20, select **`Applications`** -> **`Manage Applications`**.
2. Click on the application that you are interested in.
3. Under **`Configuration`** -> **Related Items**, select **`EJB Modules`**.

   This will display all of the EJB modules in the enterprise application, Figure 13-28 shows all the EJB modules defined for the *DefaultApplication* enterprise application. This enterprise application has only one EJB module, *Increment.jar*:



User ID: sanchez

Message(s)

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

The server may need to be restarted for these changes to take effect.

Applications > DefaultApplication >

EJB Modules

A server-module installation binding for an EJB module.

Total EJB Modules: 1

Filter

Preferences

	Uri
☐	Increment.jar

WebSphere Status < Previous Next > August 9, 2002 7:16:58 PM EDT

WebSphere Configuration Problems	
Total workspace files: 1	Total configuration problems: 123

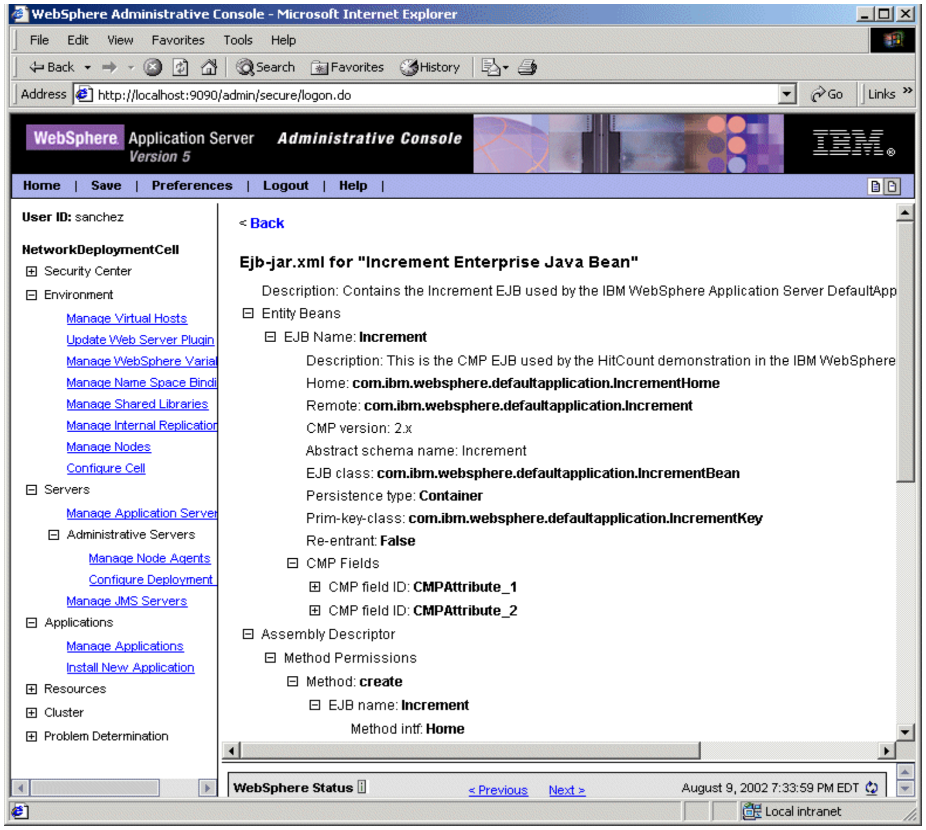
Preferences

Figure 13-28 View EJB modules in an application

4. Click on the module you are interested in to see General Properties.
5. Click **`View Deployment Descriptor`** under **`Configuration`** -> **`Additional Properties`**.

The View Deployment Descriptor window, partly shown in Figure 13-29, shows details of the EJB module, such as:

- The name of the EJB JAR file containing the EJBs.
- The EJBs in the module.
- The type of bean.
- The name of the bean and its interface classes.
- Method permissions, etc.



Ejb-jar.xml for "Increment Enterprise Java Bean"

Description: Contains the Increment EJB used by the IBM WebSphere Application Server DefaultApp
Entity Beans
  EJB Name: **Increment**
    Description: This is the CMP EJB used by the HitCount demonstration in the IBM WebSphere
    Home: **com.ibm.websphere.defaultapplication.IncrementHome**
    Remote: **com.ibm.websphere.defaultapplication.Increment**
    CMP version: 2.x
    Abstract schema name: Increment
    EJB class: **com.ibm.websphere.defaultapplication.IncrementBean**
    Persistence type: **Container**
    Prim-key-class: **com.ibm.websphere.defaultapplication.IncrementKey**
    Re-entrant: **False**
    CMP Fields
      CMP field ID: **CMPAttribute_1**
      CMP field ID: **CMPAttribute_2**
Assembly Descriptor
  Method Permissions
    Method: **create**
      EJB name: **Increment**
        Method intf: **Home**

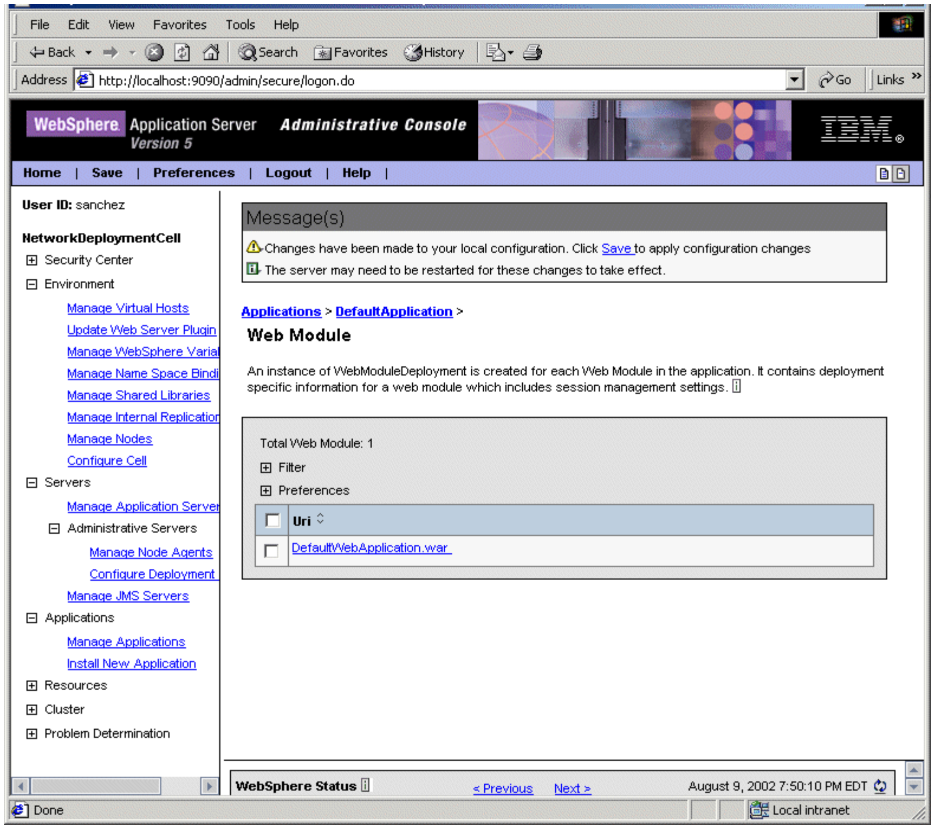
Figure 13-29 Part of the deployment descriptor for the Increment EJB module

Viewing Web modules

To see the servlets and JSPs that are part of an enterprise application:

1. From the console navigation tree, see Figure 13-20, select **`Applications`** -> **`Manage Applications`**.
2. Click on the application that you are interested in.
3. Under **`Configuration`** -> **Related Items**, select **`Web Modules.`**

   This will display all of the web modules defined for this enterprise application, Figure 13-30 shows all the Web modules defined for the *DefaultApplication* enterprise application. This enterprise application has only one web module, *DefaultWebApplication.war*:



Message(s)

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

The server may need to be restarted for these changes to take effect.

Applications > DefaultApplication >

Web Module

An instance of WebModuleDeployment is created for each Web Module in the application. It contains deployment specific information for a web module which includes session management settings.

Total Web Module: 1

Filter

Preferences

	Uri
☐	DefaultWebApplication.war

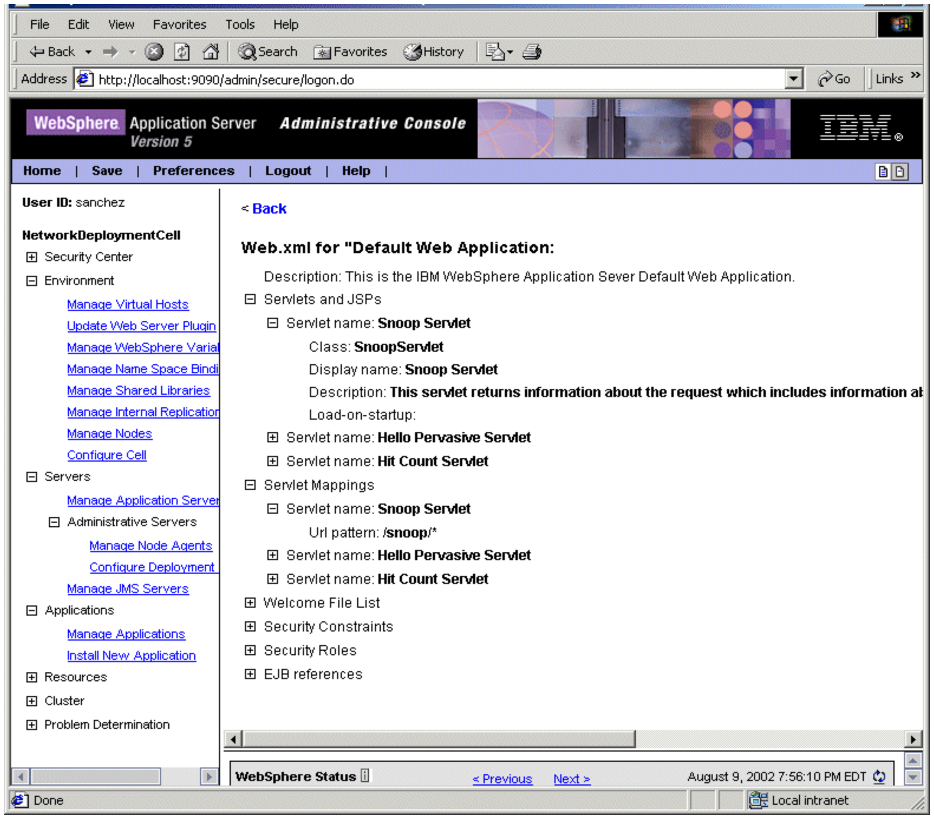
August 9, 2002 7:50:10 PM EDT

Figure 13-30 View web modules defined in an enterprise application

4. Click on the module you are interested in to see General Properties.
5. Click **`View Deployment Descriptor`** under **`Configuration`** -> **`Additional Properties`**.

The View Deployment Descriptor window, partly shown in Figure 13-31, shows 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.
- URLs for each servlet and JSP starting from the Web Module context-root.
- Security information about the servlets/JSPs.
- A list of EJBs referenced by the servlets, etc.



Web.xml for "Default Web Application:

Description: This is the IBM WebSphere Application Sever Default Web Application.
- Servlets and JSPs
  - Servlet name: **Snoop Servlet**
    - Class: **SnoopServlet**
    - Display name: **Snoop Servlet**
    - Description: **This servlet returns information about the request which includes information al**
    - Load-on-startup:
  - Servlet name: **Hello Pervasive Servlet**
  - Servlet name: **Hit Count Servlet**
- Servlet Mappings
  - Servlet name: **Snoop Servlet**
    - Url pattern: **/snoop/***
  - Servlet name: **Hello Pervasive Servlet**
  - Servlet name: **Hit Count Servlet**
- Welcome File List
- Security Constraints
- Security Roles
- EJB references

Figure 13-31 Part of the deployment descriptor for the DefaultApplication Web Module

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. The URL is partly defined in the deployment descriptor provided in the EAR file and partly defined in the deployment descriptor for the Web module containing the servlet or JSP.

To find the URL for a servlet or JSP:

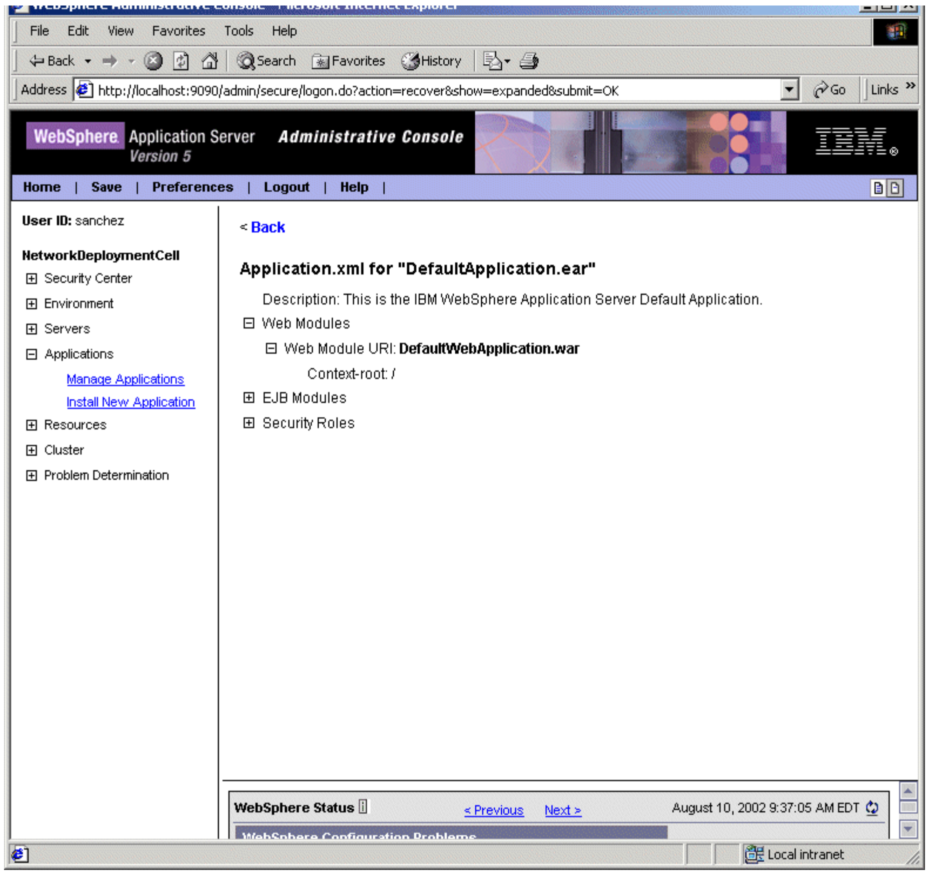
1. Find the context root of the Web module containing the servlet.
2. Find the URL for the servlet.
3. Find the virtual host where the Web module is installed.
4. Find the aliases by which the virtual host is known.
5. Combine the virtual host alias, context root and URL pattern to form the URL request of the servlet/JSP.

For example, to look up the URL for the `snoop` servlet:

1. Find the context root of the Web module `DefaultWebApplication` of the `DefaultApplication` enterprise application. This Web module contains the `snoop` servlet.
   a. From the console navigation tree, select **`Applications`** -> **`Manage Applications`**.
   b. Click on the application that you are interested in, in our case `DefaultApplication`.
   c. Under **`Configuration`** -> **`Additional Properties`**, select **`View Deployment Descriptor`**.

      Figure 13-32 shows the Deployment Descriptor window for the *DefaultApplication* enterprise application. You can see:

      i. There is only one Web module in this application, `DefaultWebApplication`.
      ii. The context-root for the `DefaultWebApplication` Web module is “/”.We will use this later.


Address http://localhost:9090/admin/secure/logon.do?action=recover&show=expanded&submit=OK

User ID: sanchez

NetworkDeploymentCell

< Back

Application.xml for "DefaultApplication.ear"

Description: This is the IBM WebSphere Application Server Default Application.

Web Modules

Web Module URI: DefaultWebApplication.war

Context-root: /

EJB Modules

Security Roles

WebSphere Status < Previous Next > August 10, 2002 9:37:05 AM EDT

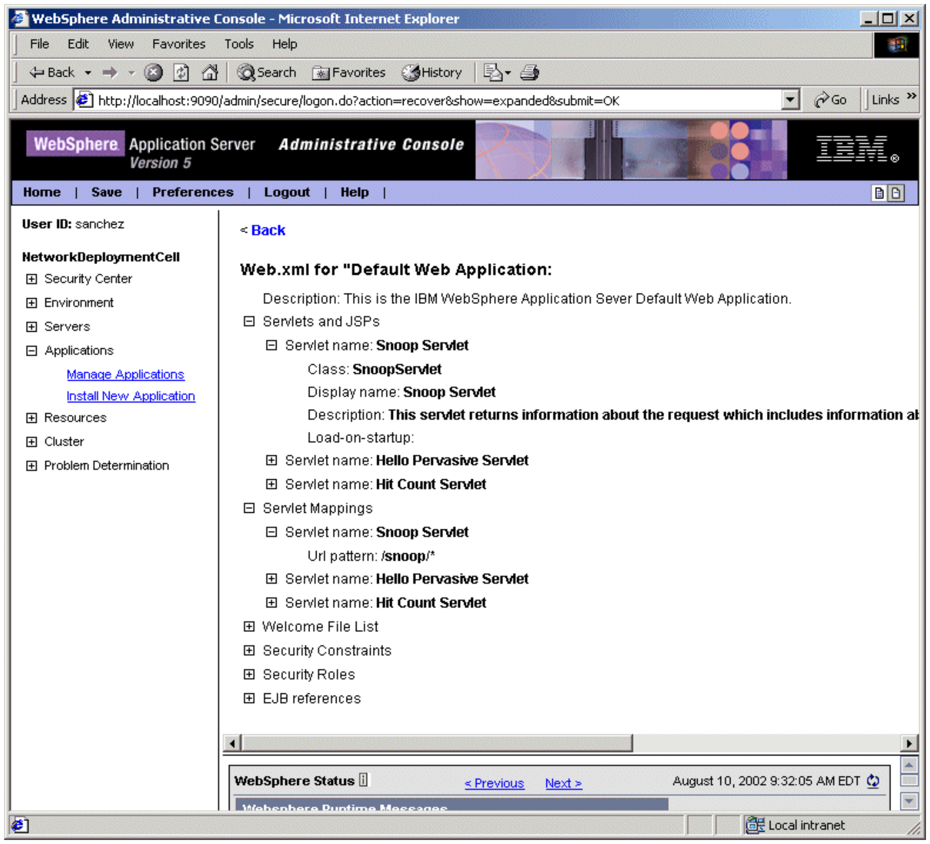
Figure 13-32 Deployment descriptor of the DefaultApplication enterprise application

2. Find the URL for the servlet `snoop`:
   a. Back in the administrative console page for the `DefaultApplication` enterprise application, under **`Configuration`** -> **Related Items**, select **`Web Modules`**.

      This will display all of the web modules defined for this enterprise application, This enterprise application has only one Web module, *DefaultWebApplication.war*:
   b. Click on the `DefaultWebApplication` Web module to see the general properties.
   c. Click **`View Deployment Descriptor`** under **`Configuration`** -> **`Additional Properties`**.

This displays the Web module properties window, as shown in Figure 13-33.

Note that the URL pattern for the servlet `snoop` starting from the Web module context root is "/snoop/*". The Web module context root was "/".


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:9090/admin/secure/logon.do?action=recover&show=expanded&submit=OK
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: sanchez
NetworkDeploymentCell
Security Center
Environment
Servers
Applications
Manage Applications
Install New Application
Resources
Cluster
Problem Determination
< Back

Web.xml for "Default Web Application:

Description: This is the IBM WebSphere Application Sever Default Web Application.
Servlets and JSPs
Servlet name: Snoop Servlet
Class: SnoopServlet
Display name: Snoop Servlet
Description: This servlet returns information about the request which includes information al
Load-on-startup:
Servlet name: Hello Pervasive Servlet
Servlet name: Hit Count Servlet
Servlet Mappings
Servlet name: Snoop Servlet
Url pattern: /snoop/*
Servlet name: Hello Pervasive Servlet
Servlet name: Hit Count Servlet
Welcome File List
Security Constraints
Security Roles
EJB references

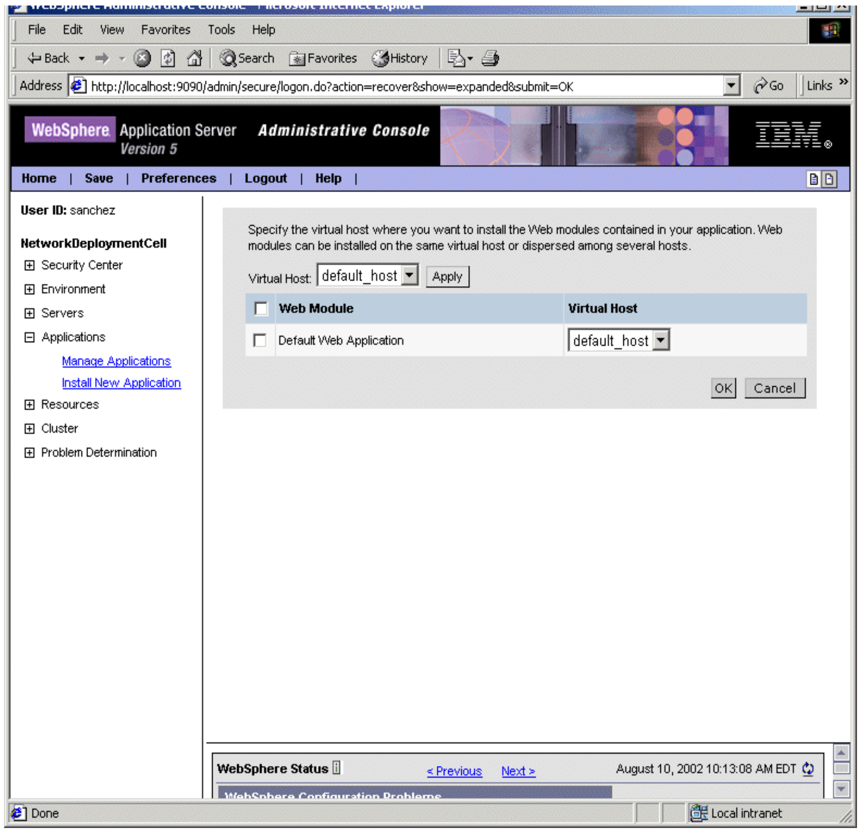
WebSphere Status < Previous Next > August 10, 2002 9:32:05 AM EDT
Websphere Runtime Messages
Local intranet

Figure 13-33 Deployment descriptor of DefaultWebApplication Web module

3. Find the virtual host where the `DefaultWebApplication` Web module is installed:

   a. Back in the administrative console page for the `DefaultApplication` enterprise application, under **`Configuration`** -> **`Additional Properties`**, select **`Map virtual hosts for web modules`**.

   This will display all of the Web modules contained in the enterprise application, and the virtual hosts in which they have been installed, Figure 13-34. Note that the `DefaultWebApplication` Web module has been installed on the `default_host` virtual host.


File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do?action=recover&show=expanded&submit=OK
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: sanchez
NetworkDeploymentCell
Security Center
Environment
Servers
Applications
Manage Applications
Install New Application
Resources
Cluster
Problem Determination
WebSphere Status
< Previous Next >
August 10, 2002 10:13:08 AM EDT
Done
Local intranet

Specify the virtual host where you want to install the Web modules contained in your application. Web modules can be installed on the same virtual host or dispersed among several hosts.

Virtual Host: default_host Apply

Web Module	Virtual Host
Default Web Application	default_host

OK Cancel

Figure 13-34 List of virtual hosts where the web modules of the DefaultApplication enterprise application have been installed

4. Find the host aliases for the `default_host` virtual host.

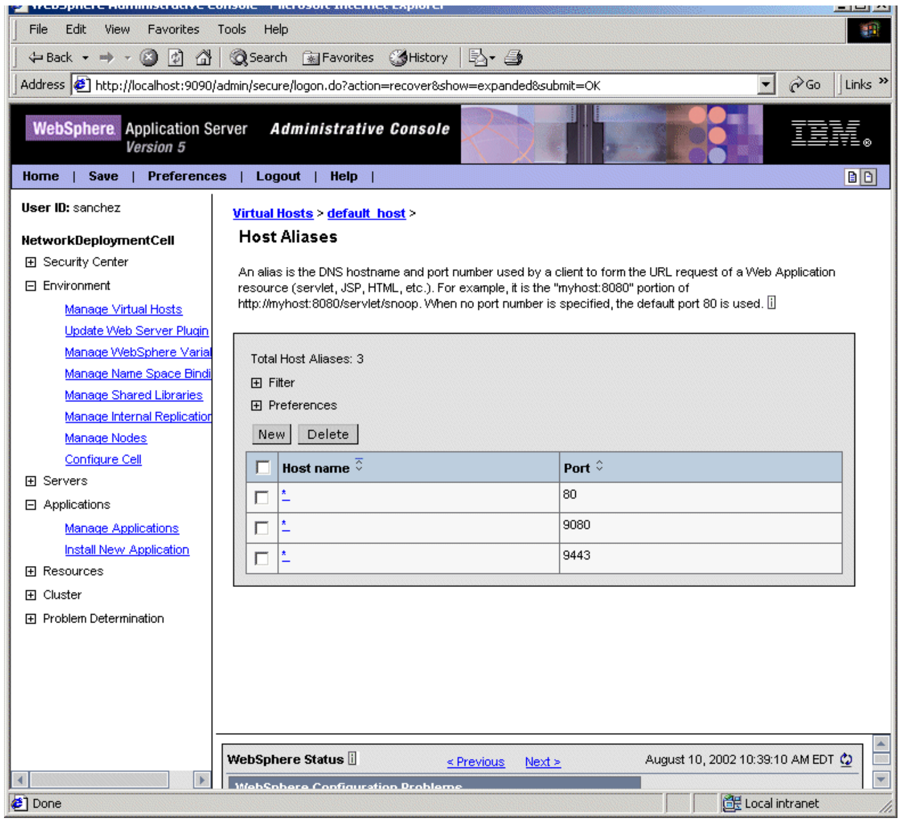
   a. From the console navigation tree, select **`Environment`** -> **`Manage Virtual Hosts`**.

   b. Click on the name of the virtual host, *default_host*.

   c. Under **`Configuration`** -> **`Additional Properties`**, select **`Host Aliases`**.

      This shows the list of aliases by which the `default_host` virtual host is known, as shown in Figure 13-35.

      Note that the aliases are composed of a DNS hostname and a port number. The host aliases for the `default_host` virtual host are *:80, *:9080 and *:9443, “*” meaning any hostname.



Virtual Hosts > default_host >

Host Aliases

An alias is the DNS hostname and port number used by a client to form the URL request of a Web Application resource (servlet, JSP, HTML, etc.). For example, it is the "myhost:8080" portion of http://myhost:8080/servlet/snoop. When no port number is specified, the default port 80 is used.

Total Host Aliases: 3

Filter

Preferences

New Delete

	Host name	Port
☐	*	80
☐	*	9080
☐	*	9443

Figure 13-35 Default_host virtual host aliases

5. Combine the virtual host alias, context root and URL pattern to form the URL request of the `snoop` servlet.

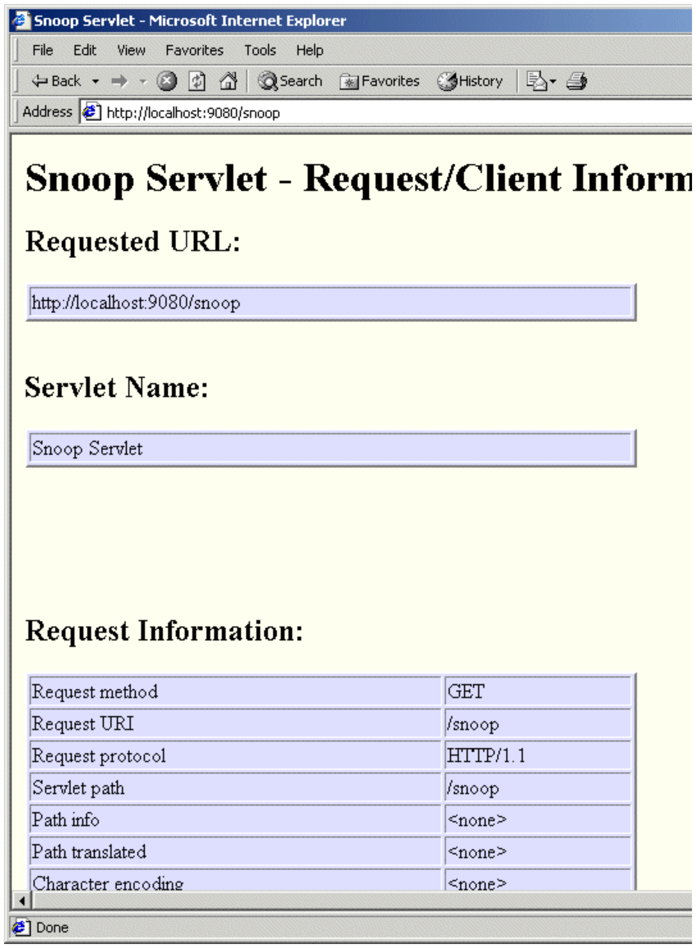
   You should be able to access the snoop servlet with a URL of the form:

```
http://*:80/snoop
http://*:9080/snoop
https://*:9443/snoop
```

   Figure 13-36, shows access to the `snoop` servlet using URL:

```
http://localhost:9080/snoop
```

   Make sure the application containing the resource, in this case the `DefaultApplication`, is up and running.


Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
History
Address
http://localhost:9080/snoop
Done

Snoop Servlet - Request/Client Inform

Requested URL:

http://localhost:9080/snoop

Servlet Name:

Snoop Servlet

Request Information:

Request method	GET
Request URI	/snoop
Request protocol	HTTP/1.1
Servlet path	/snoop
Path info	<none>
Path translated	<none>
Character encoding	<none>

Figure 13-36 Access to the snoop servlet

Note: Ports 9080 and 9443 are the default HTTP transport ports for the Web container of the application server hosting the DefaultApplication enterprise application. However you should use a dedicated Web server to access WebSphere resources. For information on how to configure the Web server refer to Chapter 16, “Configuring the Web server interface” on page 549

13.7.4 Miscellaneous

Regenerating Web server plug-in configuration

Changing certain WebSphere configuration properties makes it necessary to regenerate the Web server plug-in configuration file . You should regenerate the plug-in file after for example, installing or removing an enterprise application, adding or removing servlets and mappings from a particular application or after changing the configuration for the plug-in (a virtual host, a transport, etc).

For details on how to regenerate the plug-in configuration, see Chapter 16, “Configuring the Web server interface” on page 549.

Checking versions

Author Comment: Does this belong in PD chapter?

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 versions and build levels of the various components are in your environment.

WebSphere Application Server version

To check the version of the product installed on each of the nodes in the cell, and the Network Deployment product installed in the dmgr node (only one per cell), use one of these options.

1. Look in the product version properties file of the specific installation:

   ```
   Base: <WAS_ROOT>/properties/version/BASE.product
   Deployment manager: <WAS_ROOT>/properties/version/ND.product
   ```

   The file will contain content similar to that shown in Example 13-1.

Example 13-1 BASE.product content

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE product PUBLIC "productId" "product.dtd">
<product name="IBM WebSphere Application Server">
  <id>BASE</id>
  <version>5.0.0</version>
  <build-info date="8/14/02" level="m0232.11"></build-info>
</product>
```

2. Look in the SystemOut.log file of the specific installation:

   ```
   Base: <WAS_ROOT>/logs/nodeagent/SystemOut.log
   ```

```
Deployment manager: <WAS_ROOT>/logs/dmgr/SystemOut.log
```

The file will contain content similar to that shown in Example 13-2.

Example 13-2 Node agent SystemOut.log content

```
************ Start Display Current Environment ************
WebSphere Platform 5.0 [BASE 5.0.0 m0232.11] [ND 5.0.0 m0232.11]  running with
process name Net1Network\Net1_JH\Net1_JH and process id 1912
Host Operating System is Windows 2000, version 5.0
Java version = J2RE 1.3.1 IBM Windows 32 build cn131-20020731 (JIT enabled:
jitc), Java Compiler = jitc, Java VM name = Classic VM
...
...
************* End Display Current Environment *************
```

3. Execute the **`versionInfo`** command from the *bin* directory of the specific installation. Output similar to Example 13-3 will be generated.

Example 13-3 versionInfo output for base installation

```
$ cd <WAS_ROOT>/bin
$ versionInfo
------------------------------------------------------------------------
IBM WebSphere Application Server Version Report
------------------------------------------------------------------------
   Platform Information
------------------------------------------------------------------------
         Name: IBM WebSphere Application Server
         Version: 5.0

   Product Information
------------------------------------------------------------------------
         ID: BASE
         Name: IBM WebSphere Application Server
         Build Date: 8/14/02
         Build Level: m0232.11
         Version: 5.0.0

   Product Information
------------------------------------------------------------------------
         ID: ND
         Name: IBM WebSphere Application Server for Network Deployment
         Build Date: 8/14/02
         Build Level: m0232.11
         Version: 5.0.0
------------------------------------------------------------------------
End Report
------------------------------------------------------------------------
```

Note: When executed from the Deployment Manager *bin* directory, **`versionInfo`** only lists the Deployment Manager version details. When executed from the bin directory of any of the nodes, both the local node's and the DeploymentManager version details are listed.

JDK version

To figure out which version of the JDK you have installed in your environment use one of the following options:

1. Look in the SystemOut.log file:

   ```
   Base: <WAS_ROOT>/logs/nodeagent/SystemOut
   Deployment manager: <WAS_ROOT>/logs/dmgr/SystemOut
   ```

2. Run **`java -fullversion`** from the command line:

   ```
   <WAS_ROOT>/java/bin/java -fullversion
   ```

IBM HTTP Server version

To check the version of your IBM HTTP Server or Apache Web server do the following:

On Windows platforms, run **`apache -v`**. For example:

```
"D:\IBM HTTP Server\apache.exe -v”
Server version: IBM_HTTP_SERVER/1.3.24 Apache/1.3.24 (Win32)
Server built:   Jun 12 2002 16:30:01
```

On UNIX platforms, run **`httpd -v`**.

14

Configuring the environment

In this chapter, the following topics are described. The topics which are specific to the Network Deployment are indicated as ND only.

- Manage virtual hosts
- Update Web server plugin
- Manage WebSphere variables
- Naming
- Shared libraries
- Internal replication domain (ND only)

14.1 Manage virtual hosts

A virtual host is a filter used by the WebSphere plug-in to determine whether each HTTP request should be handled by WebSphere or not. A virtual host consists of host alias(es) and a host alias consists of a hostname and a port number. If “*” is specified as a hostname, all hostnames and IP addresses which the Web server can receive will be applied.

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 setting 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 request 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, an HTTP error will be returned to the user.

There are two virtual hosts which have already defined after the installation, default_host and admin_host. The default_host is to be used for user applications, such as webbank, and requests come from Web browsers via Web server. The default_host is to be used for a default application server called server1, which includes applications such as adminconsole, DefaultApplication. It is configured to match requests to ports 80, 9080, and 9443. The admin_host is to be used for administration of the WebSphere, and requests come from web browsers directly. It is configured to match requests to ports 9090 and 9043. Port 9443 and 9043 are to be used secured access, namely SSL connection.Those virtual hosts are summarized in Table 14-1 on page 511.

Table 14-1 Pre-defined virtual hosts

virtual host name	hostname	port
default_host	*	80
	*	9080
	*	9443
admin_host	*	9090
	*	9043

Each virtual host is associated with Web module(s). This association is established when an application is installed. It also can be changed via administration console after the application is installed. By associating a virtual host with a Web module, requests that match the host aliases for the virtual host should be processed by servlets/JSPs in this Web module.The plug-in also checks the URI of the request against the URIs for the Web module to determine whether the Web module can handle them or not.

A single virtual host can be associated with multiple Web modules unless each application has unique URIs. If there are same URIs among applications, different virtual hosts must be created and associated with each of the applications.

14.1.1 Create virtual host

In most cases, only the default_host is associated with user applications to serve requests. There are some cases that multiple virtual hosts can/should be created, for example,

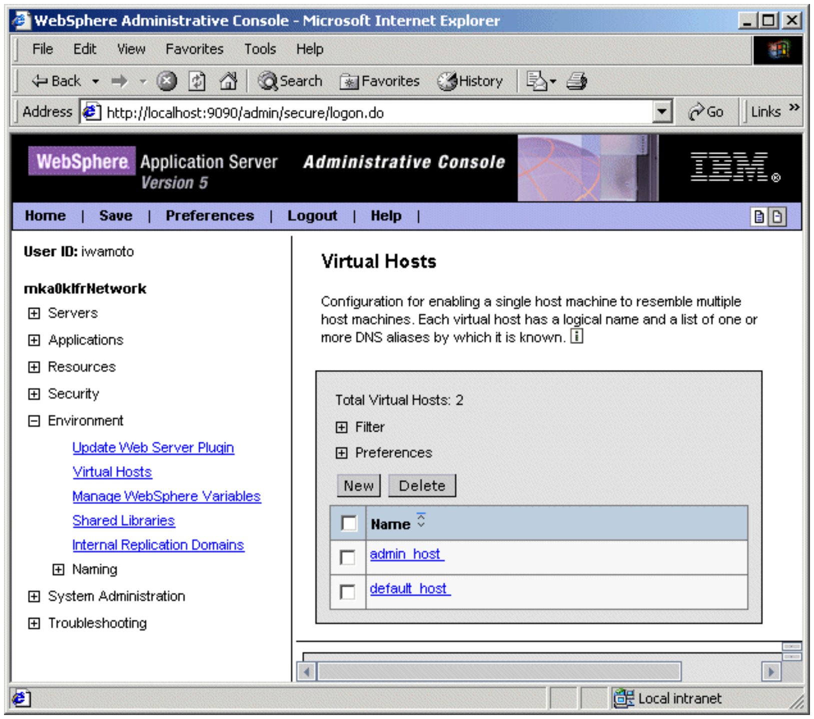
- Applications having conflict URIs
- Support extra ports such as port 443 for SSL
- Clearly keep independence of each virtual host for applications/servers

After a virtual host is created, Web server plug-in needs to be updated. Refer to “Update Web server plugin” on page 517.

The configuration of virtual host is applied to entire cell.

To create a new virtual host:

1. Click **Environment -> Manage Virtual Hosts -> New.**


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: iwamoto
mka0klfrNetwork
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variables
Shared Libraries
Internal Replication Domains
Naming
System Administration
Troubleshooting
Virtual Hosts
Filter
Preferences
New
Delete
Local intranet

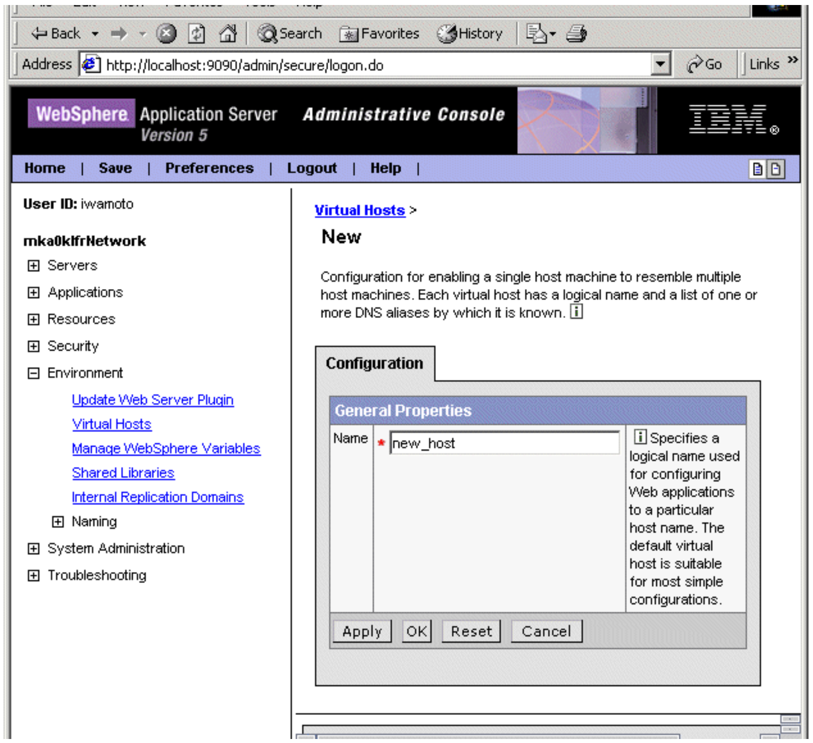
Configuration for enabling a single host machine to resemble multiple host machines. Each virtual host has a logical name and a list of one or more DNS aliases by which it is known.

Total Virtual Hosts: 2

	Name
☐	admin_host
☐	default_host

Figure 14-1 Virtual Host

2. Enter new virtual host name to be created (ex. new_host) and click **Apply**



Virtual Hosts >

New

Configuration for enabling a single host machine to resemble multiple host machines. Each virtual host has a logical name and a list of one or more DNS aliases by which it is known.

Configuration

General Properties

Name	* new_host	Specifies a logical name used for configuring Web applications to a particular host name. The default virtual host is suitable for most simple configurations.

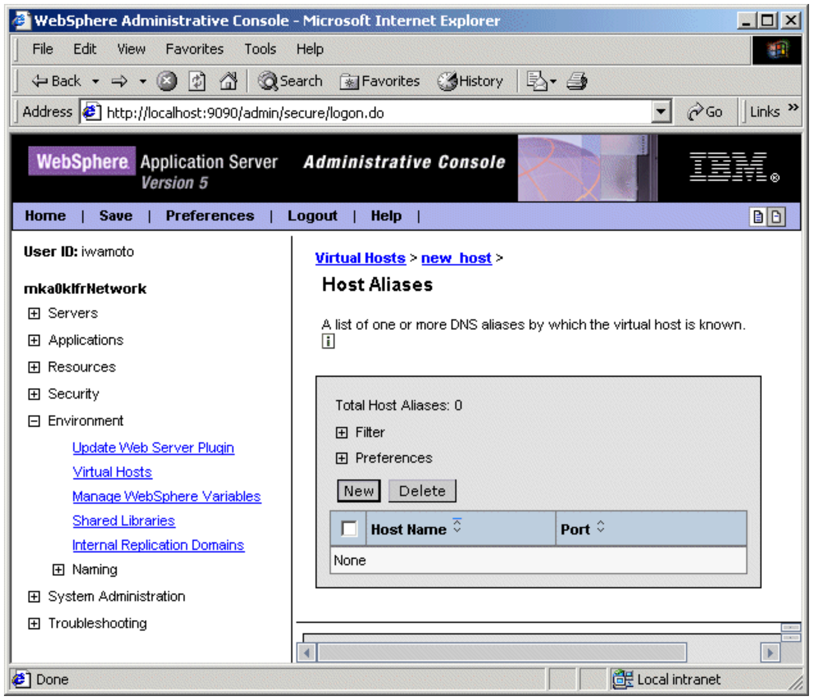
Figure 14-2 New Virtual Host

14.1.2 Host aliases

After creating a new virtual host, host alias(es) should be added. A host alias consists of a DNS hostname and port number.

To create host aliases:

1. Click **Host Aliases**.
2. Click **New**


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
User ID: iwamoto
mka0klfrNetwork
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variables
Shared Libraries
Internal Replication Domains
Naming
System Administration
Troubleshooting
Virtual Hosts > new_host >
Host Aliases
Total Host Aliases: 0
Filter
Preferences
New Delete
Done
Local intranet

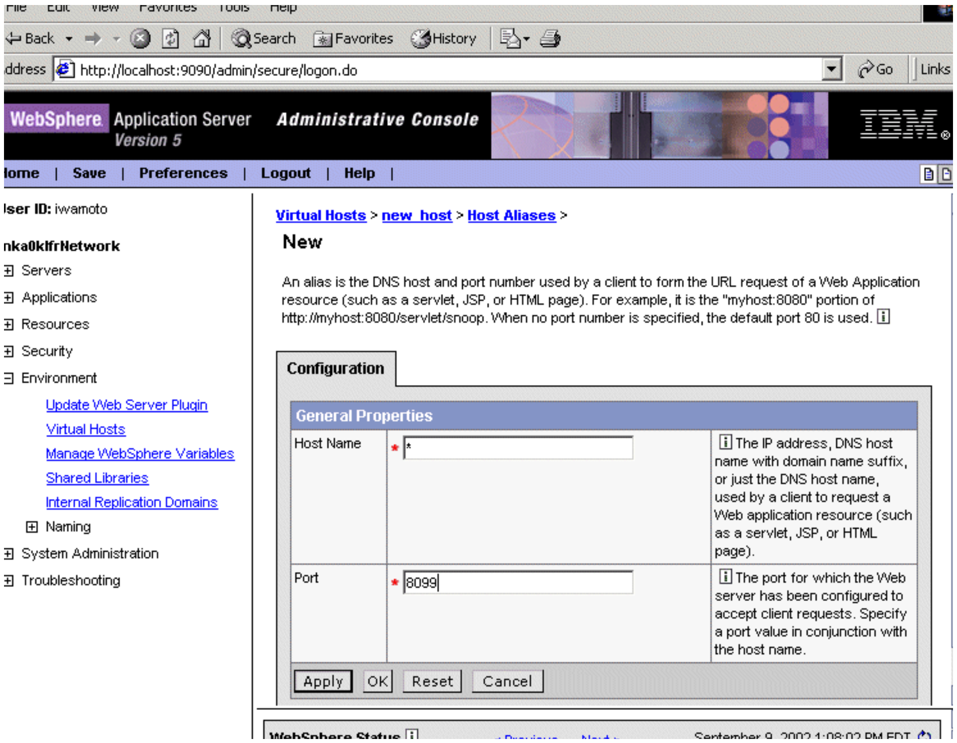
A list of one or more DNS aliases by which the virtual host is known.

	Host Name	Port
None		

Figure 14-3 Host Aliases

3. Enter **Host name** and **Port** respectively and click **Apply.**

Simple wild cards can be used in the host aliases. A* may be used for the host name or the port or both. It means that any request will match this rule. If multiple host aliases need to be added, repeat step 2) and 3).



Virtual Hosts > new_host > Host Aliases >

New

An alias is the DNS host and port number used by a client to form the URL request of a Web Application resource (such as a servlet, JSP, or HTML page). For example, it is the "myhost:8080" portion of http://myhost:8080/servlet/snoop. When no port number is specified, the default port 80 is used.

Configuration

General Properties

Host Name	* *	The IP address, DNS host name with domain name suffix, or just the DNS host name, used by a client to request a Web application resource (such as a servlet, JSP, or HTML page).
Port	* 8099	The port for which the Web server has been configured to accept client requests. Specify a port value in conjunction with the host name.

Figure 14-4 New Host Alias

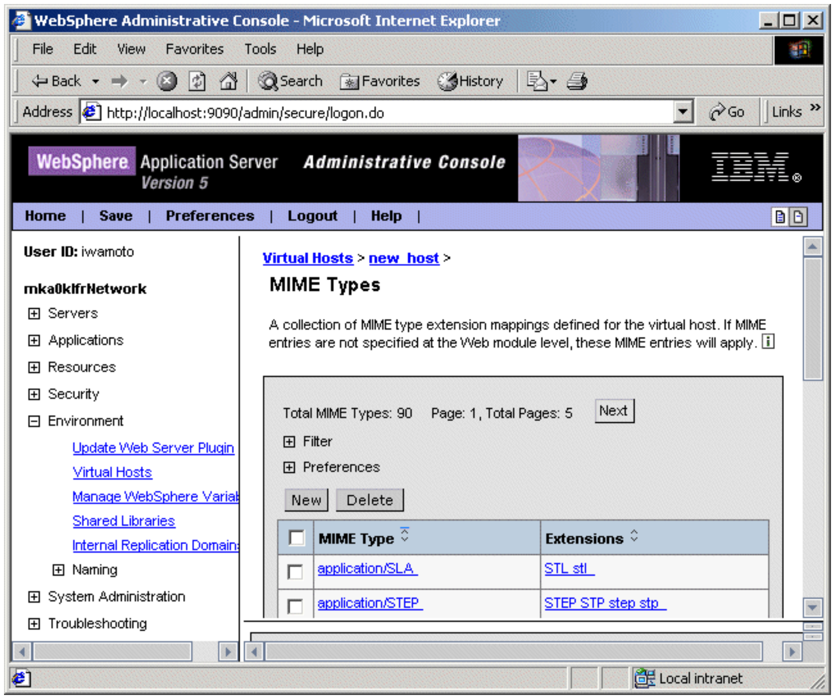
Note: If the virtual host is used in a cluster environment, all host aliases used by servers in the cluster should be registered in the virtual host.

14.1.3 MIME types

Multi-Purpose Internet Mail Extensions (MIME) mappings associate a file name extension with a type of data file (text, audio, image).

To add MIME types:

1. Click **MIME Types**
2. Click **New**


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: iwamoto
mka0klfrNetwork
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variab
Shared Libraries
Internal Replication Domain
Naming
System Administration
Troubleshooting
Local intranet

Virtual Hosts > new_host >

MIME Types

A collection of MIME type extension mappings defined for the virtual host. If MIME entries are not specified at the Web module level, these MIME entries will apply.

Total MIME Types: 90 Page: 1, Total Pages: 5 Next

Filter

Preferences

New Delete

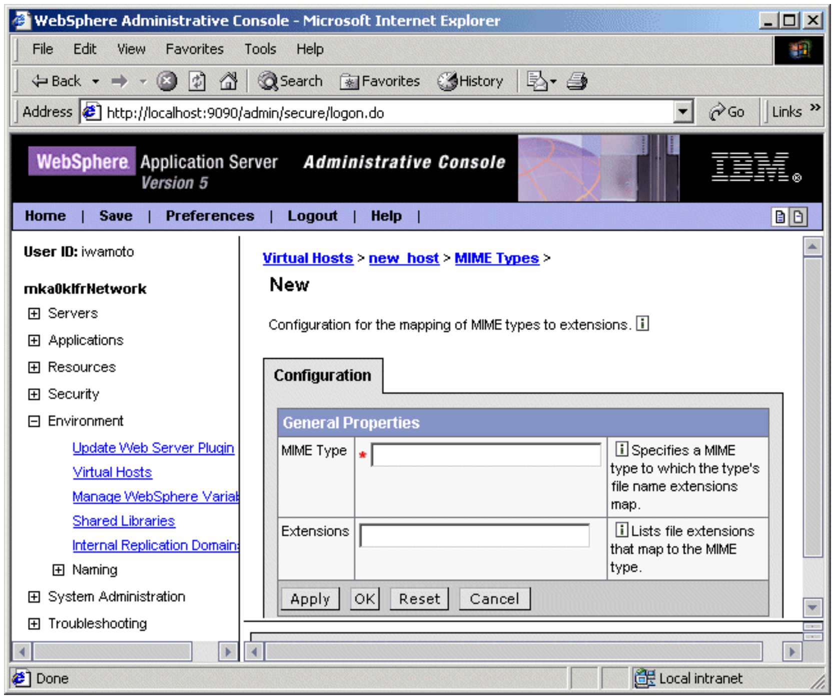
	MIME Type	Extensions
☐	application/SLA	STL stl
☐	application/STEP	STEP STP step stp

Figure 14-5 MIME Types

3. Enter **MIME Types** and **Extensions** respectively and click **Apply**

If multiple MIME types need to be added, repeat step 2) and 3).

Author Comment: When new virtual host is created, no MIME types are defined in it whereas default MIME types are defined in V4.0. The Number of defined MIME types in default_host also reduced. Does this mean that the specification changed from V4.0 to V5.0?


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: iwamoto
mka0klfrNetwork
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variables
Shared Libraries
Internal Replication Domains
Naming
System Administration
Troubleshooting
Virtual Hosts > new_host > MIME Types >
New
Configuration
General Properties
Apply OK Reset Cancel
Done
Local intranet

Configuration for the mapping of MIME types to extensions.

MIME Type	*	Specifies a MIME type to which the type's file name extensions map.
Extensions		Lists file extensions that map to the MIME type.

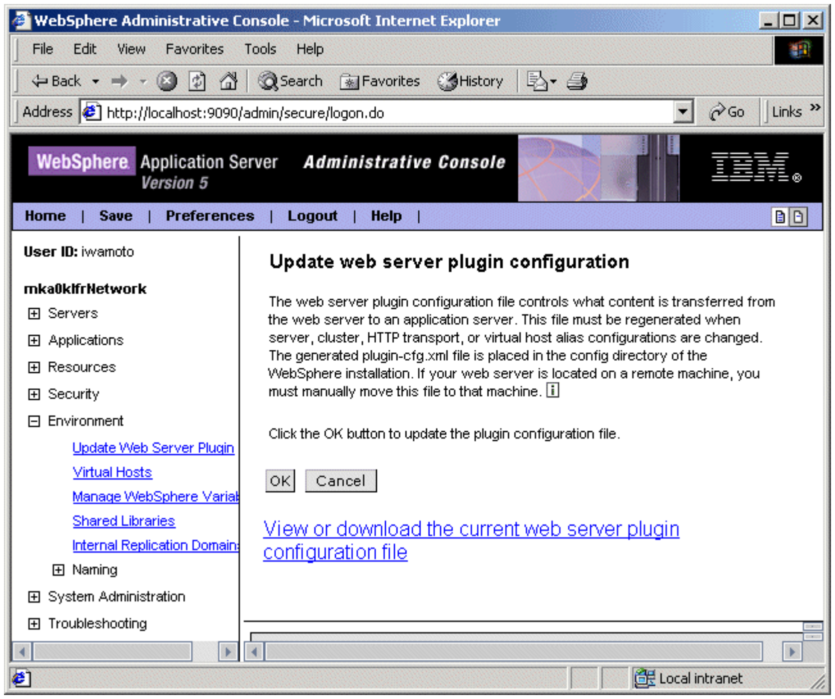
Figure 14-6 New MIME Type

14.2 Update Web server plugin

After a virtual host, server, http transport, or cluster is changed, Web server plugin should be updated so that Web server can handle requests properly. Refer to *Chapter 16 Configuring the Web server interface* for detailed explanation.

To update Web server plugin:

1. Click **Environment -> Update Web Server Plugin** and click **OK**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: iwamoto
mka0klfrNetwork
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variat
Shared Libraries
Internal Replication Domain
Naming
System Administration
Troubleshooting
OK
Cancel
Local intranet

Update web server plugin configuration

The web server plugin configuration file controls what content is transferred from the web server to an application server. This file must be regenerated when server, cluster, HTTP transport, or virtual host alias configurations are changed. The generated plugin-cfg.xml file is placed in the config directory of the WebSphere installation. If your web server is located on a remote machine, you must manually move this file to that machine.

Click the OK button to update the plugin configuration file.

View or download the current web server plugin configuration file

Figure 14-7 Update Web Server Plugin

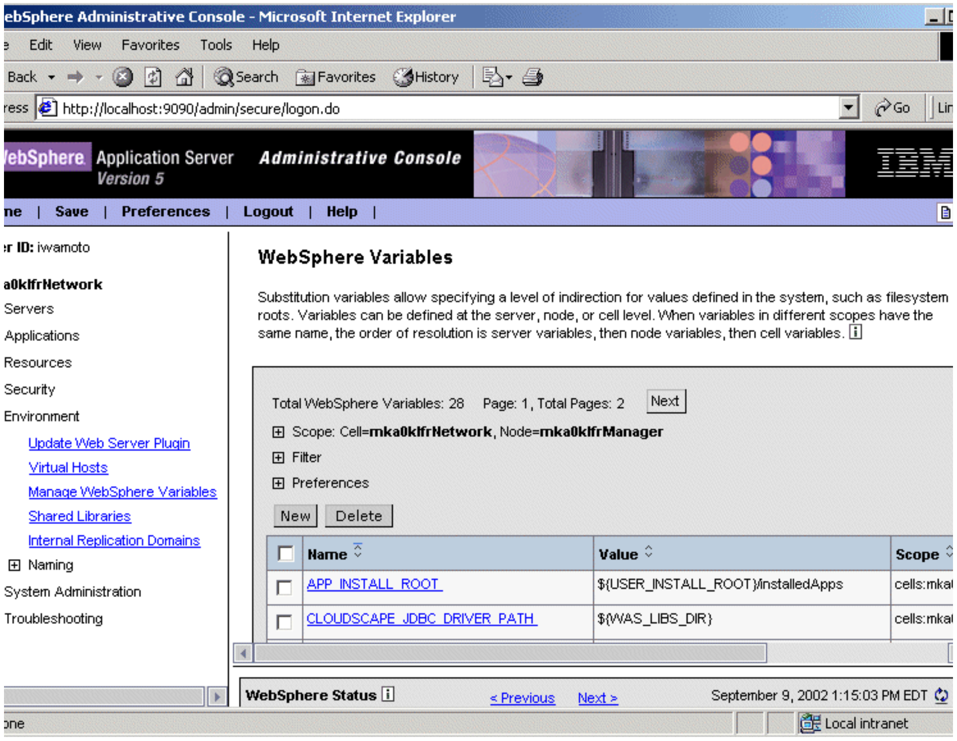
Note: Web server has to be restarted after Web server plugin has been updated. If Web server runs on a different machine from WebSphere machine, plugin-cfg.xml file should be copied to the machine manually.

14.3 Manage WebSphere variables

A WebSphere Variable is a set of a variable and a value. This prevents same value, such as WebSphere installation directory, from appearing in many places in the configuration. For example, DB2 installation directory should be defined here so that the variable can be used to define a JDBC provider of DB2.

To set a WebSphere variable:

1. Click **Environment -> Manage WebSphere Variables** and click **New**.


ebSphere Administrative Console - Microsoft Internet Explorer
e Edit View Favorites Tools Help
Back Search Favorites History
ress http://localhost:9090/admin/secure/logon.do
Go Lin
ebSphere Application Server Version 5 Administrative Console
IBM
ne | Save | Preferences | Logout | Help |
r ID: iwamoto
a0klfrNetwork
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variables
Shared Libraries
Internal Replication Domains
Naming
System Administration
Troubleshooting
WebSphere Status
< Previous
Next >
September 9, 2002 1:15:03 PM EDT
one
Local intranet

WebSphere Variables

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

Total WebSphere Variables: 28 Page: 1, Total Pages: 2 Next

Scope: Cell=**mka0klfrNetwork**, Node=**mka0klfrManager**

Filter

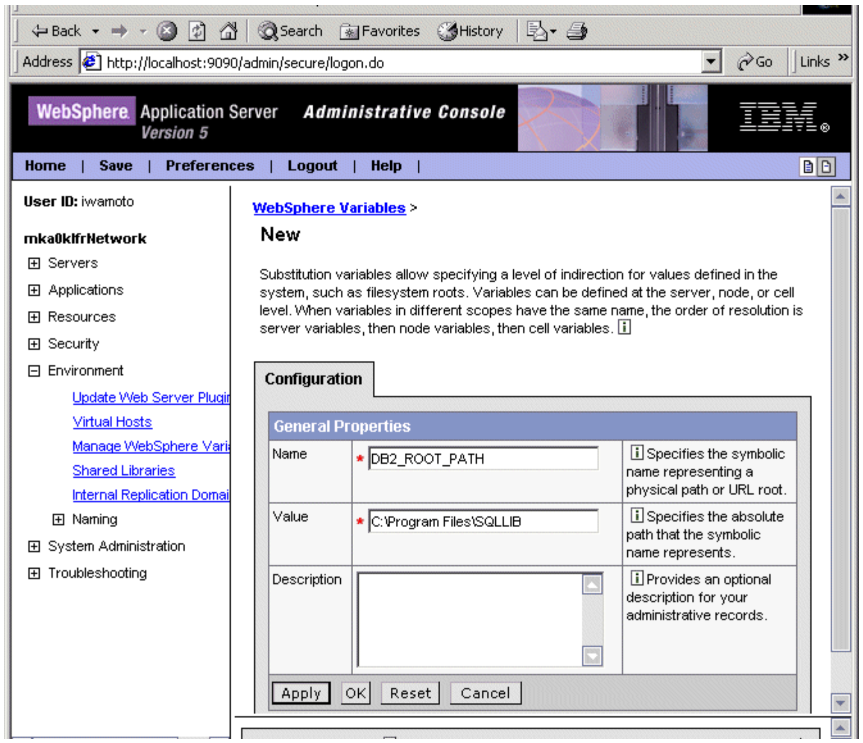
Preferences

New Delete

	Name	Value	Scope
☐	APP_INSTALL_ROOT	${USER_INSTALL_ROOT}/installedApps	cells:mka
☐	CLOUDSCAPE_JDBC_DRIVER_PATH	${WAS_LIBS_DIR}	cells:mka

Figure 14-8 WebSphere Variables

2. Enter **Name** and **Value** and click **Apply**. In this example, **`DB2_ROOT_PATH`** is used as name and **`c:\Program Files\SQLLIB`** is used as value.



Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

Name	* DB2_ROOT_PATH	Specifies the symbolic name representing a physical path or URL root.
Value	* C:\Program Files\SQLLIB	Specifies the absolute path that the symbolic name represents.
Description		Provides an optional description for your administrative records.

Figure 14-9 New WebSphere Variable

14.4 Naming

14.4.1 Name space bindings

Objects can be added to the namespace without writing code. Name space binding can be done for the following four types:

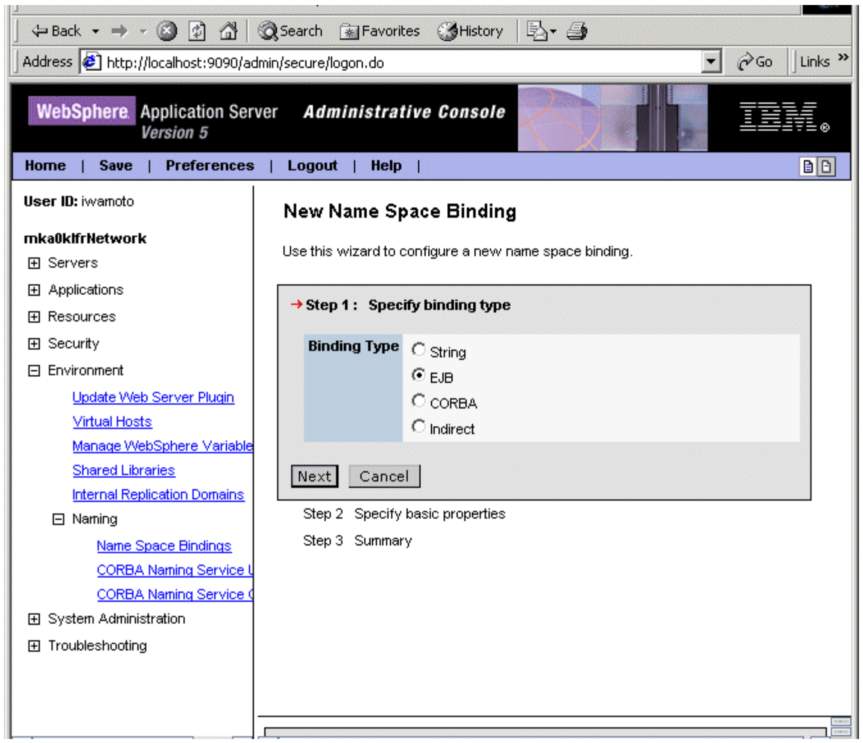
- String
- EJB
- CORBA
- Indirect

Please refer to the Chapter 6, “Naming” on page 105 for general meaning of naming and to the 6.5, “Configured bindings” on page 123 for each binding type.

For example, for WebSphere 4.0 EJB client to access to WebSphere 5.0 EJB, this binding is useful. The following step illustrates how to configure the example in Figure 6-8 on page 142.

To create configured binding:

1. Click **Environment -> Naming -> Manage Name Space Bindings** and click **New**.



http://localhost:9090/admin/secure/logon.do

User ID: iwamoto

mka0klfrNetwork

New Name Space Binding

Use this wizard to configure a new name space binding.

→ Step 1 : Specify binding type

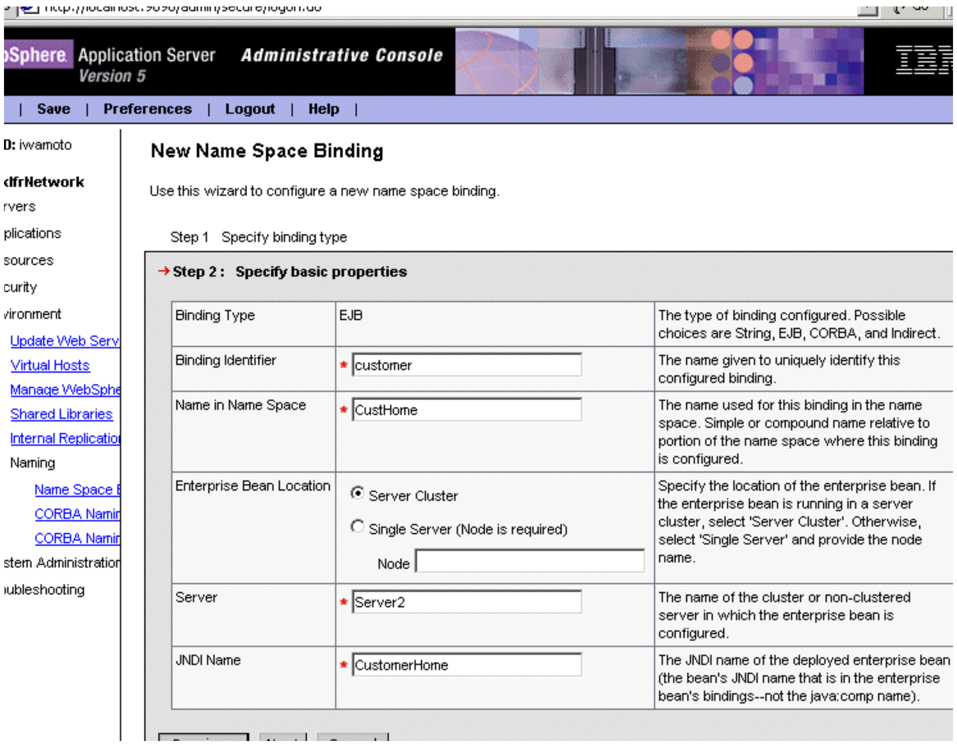
Binding Type: String [✓ EJB] CORBA Indirect

Step 2 Specify basic properties

Step 3 Summary

Figure 14-10 Name Space Binding

2. Choose **EJB** and Click **Next**



New Name Space Binding

Use this wizard to configure a new name space binding.

Step 1 Specify binding type

→ Step 2 : Specify basic properties

Binding Type	EJB	The type of binding configured. Possible choices are String, EJB, CORBA, and Indirect.
Binding Identifier	* customer	The name given to uniquely identify this configured binding.
Name in Name Space	* CustHome	The name used for this binding in the name space. Simple or compound name relative to portion of the name space where this binding is configured.
Enterprise Bean Location	[● Server Cluster] ○ Single Server (Node is required) Node	Specify the location of the enterprise bean. If the enterprise bean is running in a server cluster, select 'Server Cluster'. Otherwise, select 'Single Server' and provide the node name.
Server	* Server2	The name of the cluster or non-clustered server in which the enterprise bean is configured.
JNDI Name	* CustomerHome	The JNDI name of the deployed enterprise bean (the bean's JNDI name that is in the enterprise bean's bindings--not the java:comp name).

Figure 14-11 Name Space Binding - EJB

3. Enter **`customer`** as Binding Identifier, **`CustHome`** as Name in Name Space, **`Server2`** as Server, and **`CustomerHome`** as JNDI Name.
4. Click **Next**
5. Click **Finish**

14.5 Shared libraries

A shared library consists of its library name and a classpath which contains its jar file name. It also includes a native path which is a path to any native libraries if they are required by the shared library.

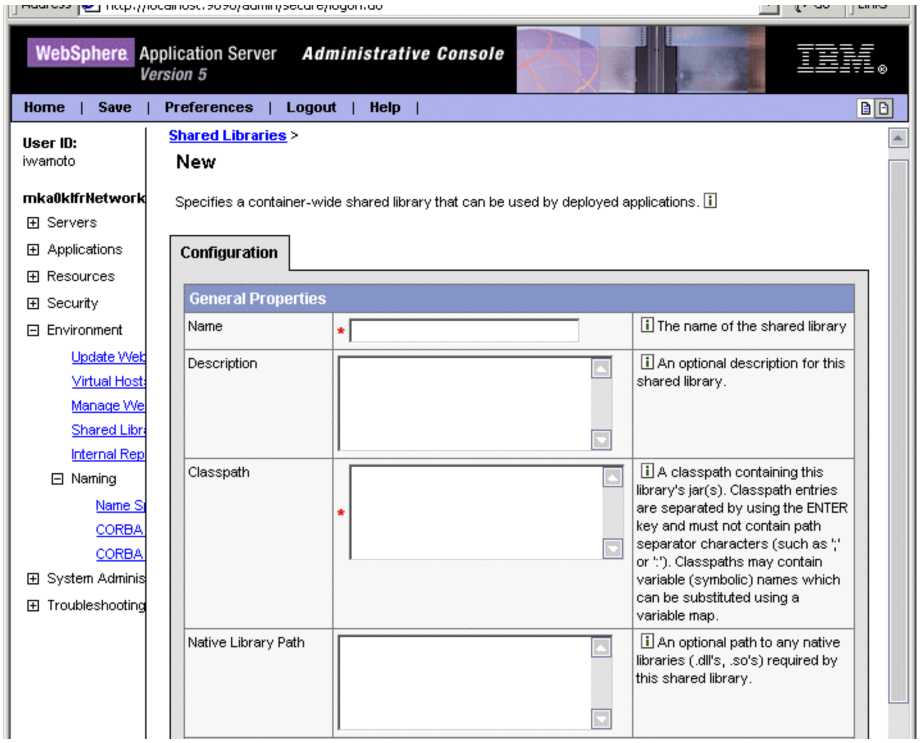
A library that is used by multiple applications should be defined here. It is very useful for it to be defined here because its information, such as its classpath and its native path, can be kept at one place. Each application that uses the library is able to refer to it just by its name.

In order for an application to use a shared library, there are two steps:

1. Define a shared library
2. Associate it to the application

To define a shared library:

1. Click **Environment -> Shared Libraries -> New**



Shared Libraries >

New

Specifies a container-wide shared library that can be used by deployed applications.

General Properties

Name	*	The name of the shared library
Description		An optional description for this shared library.
Classpath	*	A classpath containing this library's jar(s). Classpath entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Classpaths may contain variable (symbolic) names which can be substituted using a variable map.
Native Library Path		An optional path to any native libraries (.dll's, .so's) required by this shared library.

Figure 14-12 Manage Shared Library

2. Enter **name**, **classpath**, and **native path** and click **Apply**.

To use a shared library by an application,

1. Click **Applications -> Applications ->** application_name **-> Libraries**.

Please refer to the Chapter 20, “Deploying an application” on page 835 for an application configuration.

14.6 Internal replication domains

Multiple WebSphere 5.0 server instances in a replication domain can share session information without session database. It is called memory-to-memory replication. Refer to the Chapter 17, “Configuring session management” on page 551 for detailed explanation and configuration.

Note: This functionality is applied only to WAS Network Deployment.

15

Server Configuration

An application server is the component which provides the J2EE compliant runtime environment to enterprise applications. The application server provides a Web container, EJB container and various services for enterprise applications. The application server makes enterprise applications available to clients. Finally the application server itself is a managed process that offers JMX MBeans to manage its resources.

In this chapter, the following topics are described. The topics which are specific to the Network Deployment environment are indicated as ND only:

- Creating an application server
- Managing an application server
- Managing JMS servers (ND only)
- Managing clusters (ND only)
- Viewing a cluster topology (ND only)

Author Comment: Look at Ascension's chapter and make sure that the format is consistent across all the configuration chapters. The session management chapter used the WebSPhere 4.0 Handbook as a reference for format, may want to look to see if Ascension did the same thing.

15.1 Creating application servers

In any WebSphere Application Server node a server1 application server is created by default. In a single server environment this application server hosts both the administrative console and the sample enterprise applications. In a Network Deployment environment the administrative console is removed from server1 on the node and installed in the deployment manager application server (dmgr) instead.

In this section we wil discuss the steps necessary to create additional application servers. First, invoke the administrative console, expand the **Servers** branch from the navigation tree, and click **Manage Aplication Servers**. A list of application servers defined in the cell will show in the workspace.

From the application servers workspace, click **New** to lauch the **Create New Application Server** guide as shown in Figure 15-1.

The following are the fields found in Step 1 of the Create New Application Server guide.

- Select Node

  Select on which node the application should be created. The drop down list shows all available nodes in the cell. As only one node is available in a single server environment, only one node is listed for this environment.

- Server Name

  Type a name used for identifying the application server in the cell.

- HTTP Ports

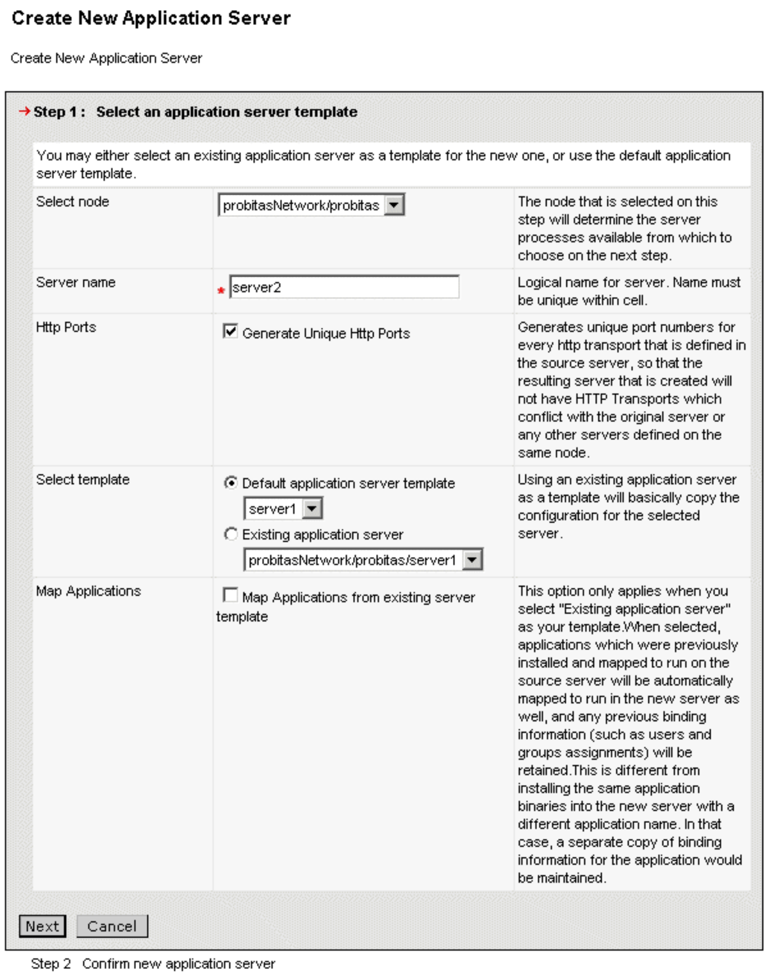
  Select whether the TCP ports used for connecting from a client to the Web container on this application server will be carried over from the source server (the server specified in the Select Template field) or should be generated unique to this node. Copying ports can be useful for creating identical application servers on different nodes (eg. for clustering purposes). Check the **Generate Unique Http Ports** checkbox if the source application server and the new application server needs to run simultanously on the same node or the new application server should have a set of unused ports generated.

- Copy Applications

  Select whether enterprise applications will be carried over from the source server (the server specified in the Select Template field). Copying enterprise applications can be useful for creating identical application servers (eg. for clustering purposes).

- Select Template

An existing server to act as template for creating your new application server needs to be specified. All configuration settings for the template application server will be used for creating this new application server.



Create New Application Server

Create New Application Server

→ **Step 1 : Select an application server template**

You may either select an existing application server as a template for the new one, or use the default application server template.

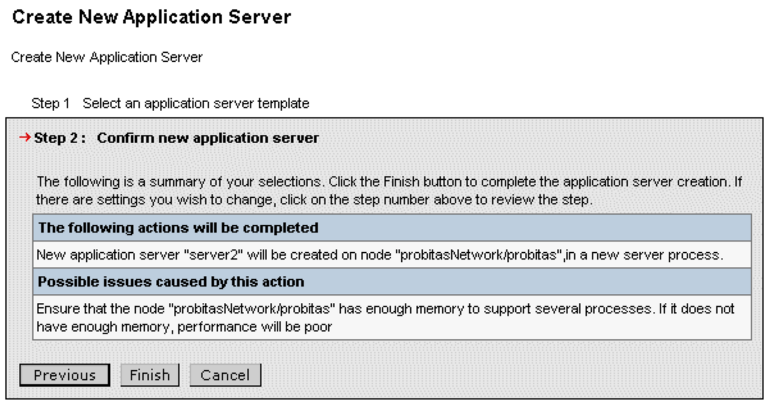
Select node	probitasNetwork/probitas	The node that is selected on this step will determine the server processes available from which to choose on the next step.
Server name	* server2	Logical name for server. Name must be unique within cell.
Http Ports	[✓] Generate Unique Http Ports	Generates unique port numbers for every http transport that is defined in the source server, so that the resulting server that is created will not have HTTP Transports which conflict with the original server or any other servers defined on the same node.
Select template	(●) Default application server template: server1 ( ) Existing application server: probitasNetwork/probitas/server1	Using an existing application server as a template will basically copy the configuration for the selected server.
Map Applications	[ ] Map Applications from existing server template	This option only applies when you select "Existing application server" as your template.When selected, applications which were previously installed and mapped to run on the source server will be automatically mapped to run in the new server as well, and any previous binding information (such as users and groups assignments) will be retained.This is different from installing the same application binaries into the new server with a different application name. In that case, a separate copy of binding information for the application would be maintained.

Step 2 Confirm new application server

Figure 15-1 Create a new application server

Select the desired options and fill in a unique name identifying this server in the cell. Click **Next** and Step 2 of the Create New Application Server guide appears as shown in Figure 15-2

Once you select finish, the administrative console will prompt to save the changes. After saving the changes the new application server appears in the list of application servers defined for the cell. This is shown in Figure 15-3.



Create New Application Server

Create New Application Server

Step 1 Select an application server template

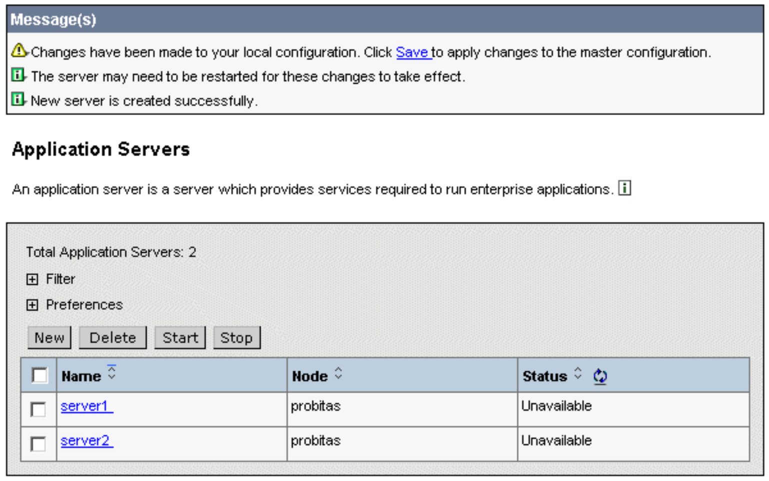
→ Step 2 : Confirm new application server

The following is a summary of your selections. Click the Finish button to complete the application server creation. If there are settings you wish to change, click on the step number above to review the step.

The following actions will be completed
New application server "server2" will be created on node "probitasNetwork/probitas",in a new server process.

Possible issues caused by this action
Ensure that the node "probitasNetwork/probitas" has enough memory to support several processes. If it does not have enough memory, performance will be poor

Figure 15-2 Confirm new application server



Message(s)

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

The server may need to be restarted for these changes to take effect.

New server is created successfully.

Application Servers

An application server is a server which provides services required to run enterprise applications.

Total Application Servers: 2

Name	Node	Status
server1	probitas	Unavailable
server2	probitas	Unavailable

Figure 15-3 New Application Server and save changes

The new application server created using the Create New Application Server guide inherited most configuration settings from the template server specified. In the next section we are going discuss all of the configuration options available for WebSphere 5.0 application servers.

15.2 Managing application servers

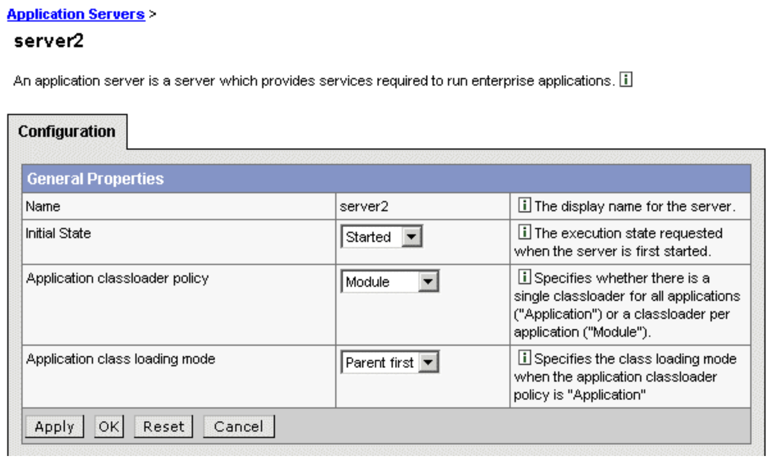
In a network deployment environment basic administration of application servers is performed from the administrative console. In a single server environment the `wsadmin` command interface is used for administration (alternatively shell scripts, such as startServer, stopServer and serverStatus is used). The `wsadmin` command interface and shell scripts are discussed in Chapter 23, “Command line administration and scripting” on page 971 and Appendix A, “Command-line tools” on page 975.

From the application servers workspace in the administrative console a list of existing application servers in the cell is found and buttons for perfoming basic adminstration tasks is shown. Go to the application servers workspace by expanding the **Servers** branch from the navigation tree of the administrative console and click **Manage Aplication Servers**. The node on which the application server is installed is shown in the application server workspace. Status information such as started, stopped and unavailable is shown here only for application servers in a network deployment environment.

To stop, start or delete one or more application servers put a checkmark in the checkbox next to the name of one or more application servers in the workspace. Click the appropriate button to perform the desired task. Only New and Delete administration tasks are available in the administrative console of a single server environment.

To view additional information about a particular application server or to further configure a server, click on the server's name in the Name column of the application server workspace. The configuration page of the application server is shown. If the application server is currently runng a Runtime tab is also available from this pane. Click on the **Runtime** tab. The Process id of the application server Java Runtime (JVM) is listed in the general properties part of this page along with cell and node information. In the network deployment environment a Product Information entry is listed in the additional properties part of the workspace. Click **Product Information** to get version and build information as well as information on components and e-fixes applied to this application server.

Go back to the configuration page of the application server by clicking the **Configuration** tab. The pane is shown in Figure 15-4. From this pane the application server is configured. The configuration options available in the general properties page are:


Application Servers >
server2

An application server is a server which provides services required to run enterprise applications.

Configuration

General Properties

Name	server2	The display name for the server.
Initial State	Started	The execution state requested when the server is first started.
Application classloader policy	Module	Specifies whether there is a single classloader for all applications ("Application") or a classloader per application ("Module").
Application class loading mode	Parent first	Specifies the class loading mode when the application classloader policy is "Application"

Figure 15-4 General Properties for COnfiguration

- Initial State

  Specifies the desired execution state of the application server when the WebSphere node containing the application server starts. In case of a system reboot, this value will be used to determine if the application server is to be restarted on system startup.

- Application class loader policy

  Application classloaders consist of EJB modules, dependency JAR files, resource adapters and shared libraries. This selection will specify if any classloader islolation is to take place between applications running on this application server.

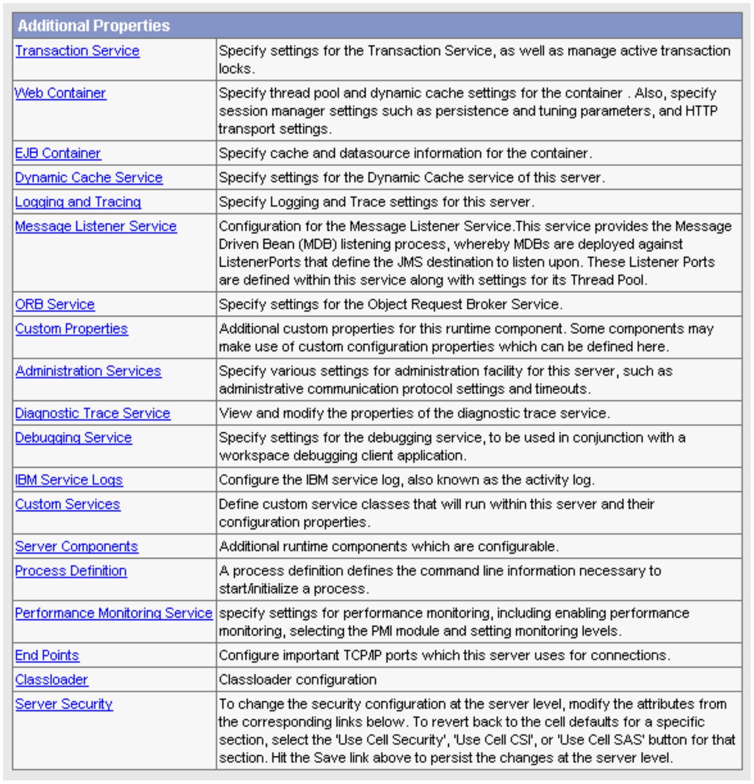
  If **SINGLE** is selected, only one object is created for managing loading of all EJB modules, dependency JAR files, resource adapters and shared files required by the applications on the application server. The benefit is that of low memory usage as shared classes are only loaded once, the drawback is the requirement for all applications to agree on a common version of all classes.

  Choosing **MULTIPLE** creates a classloader object for every application in the application server, isolating applications classes (EJB modules, dependency JAR files, resource adapters and shared libraries) from one another.

> **Note:** The WAR classloader policy for WEB modules is specified on an a per application basis. The application class loader policy configured here, at the application server level, only affects EJB modules, dependency JAR files, resource adapters and shared libraries for applications residing on this application server.

- Application class loading mode

  The class loading mode allows overriding the delegation in which classes are located and loaded. The standard J2EE classloader scheme is **PARENT_FIRST** in which an application first delegates classloading to its parent classloader before trying to locate and load a class itself. The **PARENT_LAST** reverses the J2EE standard classloading behaviour and makess any classloader try to locate and load a class before delegation takes place. The PARENT_LAST policy allows an application classloader to override and provide its own version of a class that exists in the parent classloader.



Additional Properties	
Transaction Service	Specify settings for the Transaction Service, as well as manage active transaction locks.
Web Container	Specify thread pool and dynamic cache settings for the container . Also, specify session manager settings such as persistence and tuning parameters, and HTTP transport settings.
EJB Container	Specify cache and datasource information for the container.
Dynamic Cache Service	Specify settings for the Dynamic Cache service of this server.
Logging and Tracing	Specify Logging and Trace settings for this server.
Message Listener Service	Configuration for the Message Listener Service.This service provides the Message Driven Bean (MDB) listening process, whereby MDBs are deployed against ListenerPorts that define the JMS destination to listen upon. These Listener Ports are defined within this service along with settings for its Thread Pool.
ORB Service	Specify settings for the Object Request Broker Service.
Custom Properties	Additional custom properties for this runtime component. Some components may make use of custom configuration properties which can be defined here.
Administration Services	Specify various settings for administration facility for this server, such as administrative communication protocol settings and timeouts.
Diagnostic Trace Service	View and modify the properties of the diagnostic trace service.
Debugging Service	Specify settings for the debugging service, to be used in conjunction with a workspace debugging client application.
IBM Service Logs	Configure the IBM service log, also known as the activity log.
Custom Services	Define custom service classes that will run within this server and their configuration properties.
Server Components	Additional runtime components which are configurable.
Process Definition	A process definition defines the command line information necessary to start/initialize a process.
Performance Monitoring Service	specify settings for performance monitoring, including enabling performance monitoring, selecting the PMI module and setting monitoring levels.
End Points	Configure important TCP/IP ports which this server uses for connections.
Classloader	Classloader configuration
Server Security	To change the security configuration at the server level, modify the attributes from the corresponding links below. To revert back to the cell defaults for a specific section, select the 'Use Cell Security', 'Use Cell CSI', or 'Use Cell SAS' button for that section. Hit the Save link above to persist the changes at the server level.

Figure 15-5 Additional Properties for the Application Server

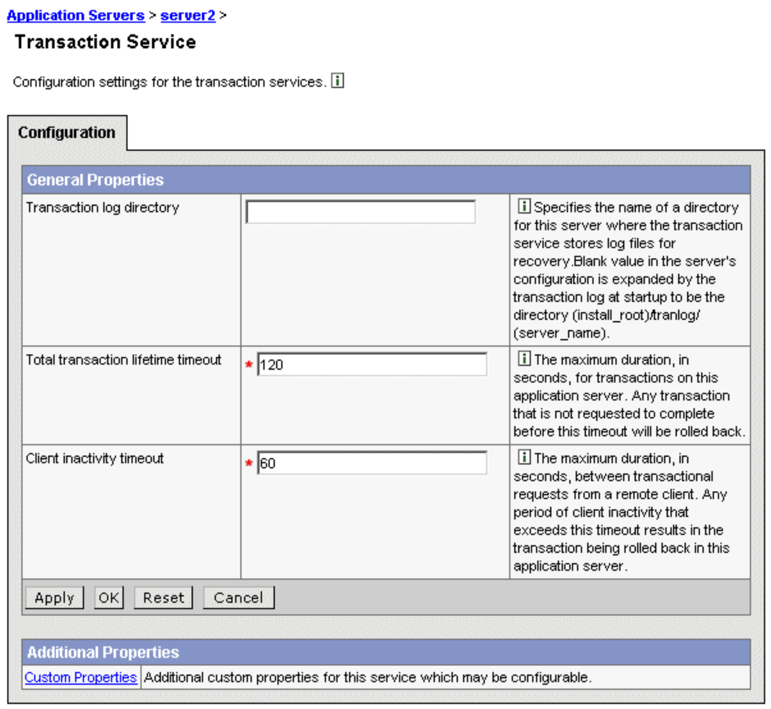
Additional Properties:Transaction Service

This page allows you to specify settings for the Transaction Service, as well as manage active transaction locks.

- Transaction Log Directory: Specifies the name of a directory for this server where the transaction service stores log files for recovery. If no value is provided for the location of this file, its location defaults to (install_root)/tranlog/(server_name) at server startup.

Author Comment: Can you supply a server address to have the file logged to a seperate server?

- Total Transaction Lifetime Timout: The maximum duration, in seconds, for transactions on this application server. Any transaction that is not requested to complete before this timeout will be rolled back in this application server. The default value of this field is 120 seconds. As a best practice you will want to examine the types of transactions you will be monitoring and determine the total time it should take to complete these transactions. Any existing business intelligence may help with this decisions.
- Client Inactivity Timeout: The maximum duration, in seconds, between transactional requests from a remote client. Any period of client inactivity that exceeds this timeout results in the transaction being rolled back in this application server.



Application Servers > server2 >

Transaction Service

Configuration settings for the transaction services.

Configuration

General Properties

Transaction log directory		Specifies the name of a directory for this server where the transaction service stores log files for recovery.Blank value in the server's configuration is expanded by the transaction log at startup to be the directory (install_root)/tranlog/(server_name).
Total transaction lifetime timeout	* 120	The maximum duration, in seconds, for transactions on this application server. Any transaction that is not requested to complete before this timeout will be rolled back.
Client inactivity timeout	* 60	The maximum duration, in seconds, between transactional requests from a remote client. Any period of client inactivity that exceeds this timeout results in the transaction being rolled back in this application server.

Additional Properties

Custom Properties	Additional custom properties for this service which may be configurable.

Figure 15-6 Transaction Services

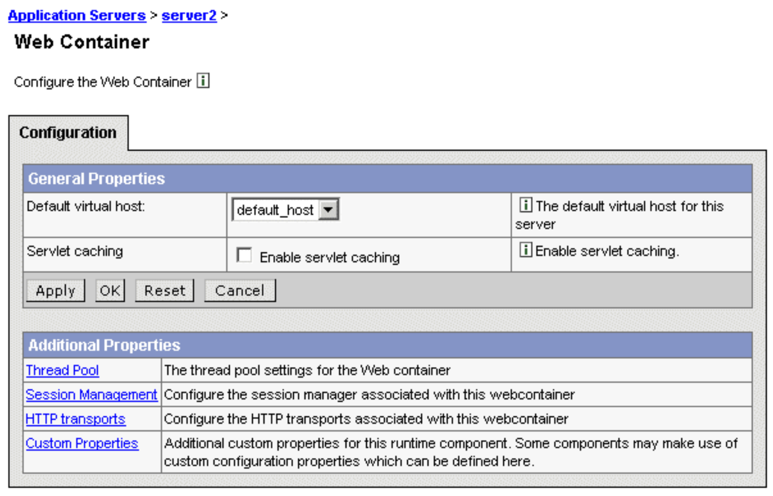
15.2.1 Additional Properties:Web Container

This page allows you to specify thread pool and dynamic cache settings for the container . Also, allows you to specify session manager settings such as persistence and tuning parameters, and HTTP transport settings. The web container will serve application requests for Servlets and JSPs. The following options are configurable for this container.

- Thread Pool: The Thread pool allows you to specify how many concurrent threads you have accessing the web container on this server. By controlling the amount of threads you can prevent “denial of service” attacks on the application server. On this page you can specify the range of threads allowed in a pool. The default values are between 10 and 50 threads. Once the thread is created, you can specify a timeout that WebSPhere will use to remove threads from the pool, based on inactivity. You also have an option to allow tWebSPhere to create threads beyond the maximun size. Be careful of this option as you are now enabling the possibility for thread attacks on the web container.

- Session Management: Session MAnagement setting allow you to determine how the web container will manage conversational data from the client. The following options are configurable:
  - Session Tracking mechanism: you have the option of enabling SSL ID tracking, enabling cookies, and enabling URL rewritting. There is also the ability to enable protocol switch rewriting. This option allow the web container to recieve requests over HTTP and then redirect them to HTTPS.
  - Maximum in memory session count: Specifies the maximum number of sessions to maintain in memory. A common attack on web containers is the spawning of sessions on the web container until the web container runs out of memory. BY setting this limit you elimiate the possibility have haveing an infinite amount of sessions created in the web container. This also identifies the amount of sessions in memory and the same number of sessions there for will be help persistently in the case of an application server failure.
  - Overflow: This option identifies whether to allow the number of sessions in memory to exceed the value specified by Max In Memory Session Count property. This is valid only in non-persistent sessions mode. When you have decided not to use the option for persistent sessions.
- HTTP Transports: Allows you to configure the transports associated with the web container.When you select HTTP Transports you can create a new transport by providing the host, port, and whether you choose to enable SSL for this port. You can also edit those atributes on any existing transport.
- Custom Properties:Allows you to specify name/value pairs for configuring internal system properties. Some components may make use, prgramaticaly of custom configuration properties which can be defined here. It is not common to pass information to the web container this way, however the J2EE

specification indicates this as a requirement. Most configuration information can be handled progrmmatically or through the deployment descriptor.


Application Servers > server2 >

Web Container

Configure the Web Container

Configuration

General Properties

Default virtual host:	default_host	The default virtual host for this server
Servlet caching	☐ Enable servlet caching	Enable servlet caching.

Additional Properties

Thread Pool	The thread pool settings for the Web container
Session Management	Configure the session manager associated with this webcontainer
HTTP transports	Configure the HTTP transports associated with this webcontainer
Custom Properties	Additional custom properties for this runtime component. Some components may make use of custom configuration properties which can be defined here.

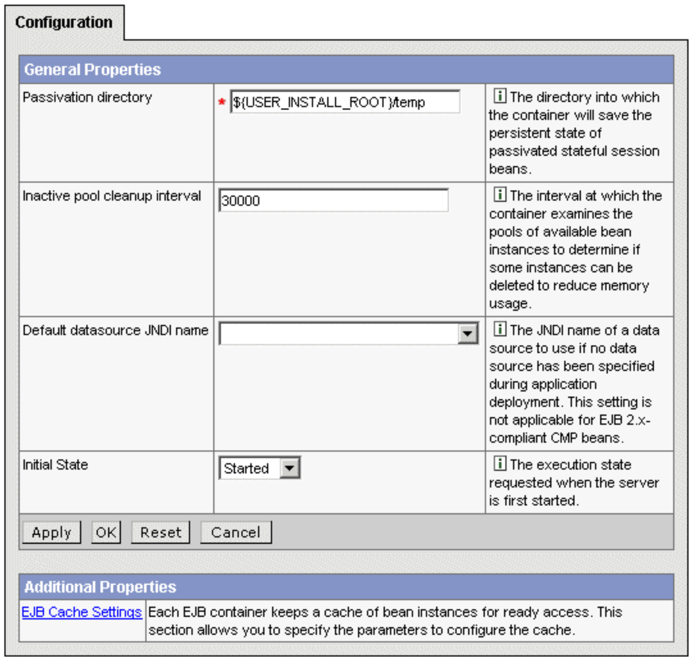
Figure 15-7 Web Container Attributes

15.2.2 Additional Properties:EJB Container

This page allows you configurer the services provides by the EJB container.

- Passivation Directory: This attribute provides the directory that you can use to store the persistent state of passivated, stateful session EJBs. If you are using the EJB container to manage session data, you should give WebSphere the ablity to swap data to disk when necessary. This directory will tell WebSphere where to hold ejb session data when it passivates and activates beans from the pool.
- Inactive pool cleanup interval: Because WebSphere will build a pool of EJBs to satisfy incoming requests, we need to tell it when to remove beans from this pool to preserve resources. This attributes allows you to define the interval at which the container examines the pools of available bean instances to determine if some instances can be deleted to reduce memory usage.
- Default Datasource JNDI name: Here you can set a default datasource to use for EJBs that have no individual datasource defined .This setting is not applicable for EJB 2.x-compliant CMP beans.
- Intial state: This attributes allows you to identify the state of the container when WebSphere is started. In the case that you have to recycle the appliction server, this attribute will be used to determine whether to start the ejb container at that time, or wait for a manual start.

- EJB Cache settings: You can set up two types of cache settings in WebSphere:
  - Cleanup interval: This attribute allows you to set the interval at which the container attempts to remove unused items from the cache in order to reduce the total number of items in cache to the value we set in the cache size attribute.
  - Cache size: This attribute specifies the number of buckets in the active instance list within the EJB container. This attributes will be used by websphere to determine how large the chache will be and when to remove components from the cache to reduce its size.



General Properties

Passivation directory	* ${USER_INSTALL_ROOT}/temp	The directory into which the container will save the persistent state of passivated stateful session beans.
Inactive pool cleanup interval	30000	The interval at which the container examines the pools of available bean instances to determine if some instances can be deleted to reduce memory usage.
Default datasource JNDI name		The JNDI name of a data source to use if no data source has been specified during application deployment. This setting is not applicable for EJB 2.x-compliant CMP beans.
Initial State	Started	The execution state requested when the server is first started.

Additional Properties

EJB Cache Settings	Each EJB container keeps a cache of bean instances for ready access. This section allows you to specify the parameters to configure the cache.

Figure 15-8 EJB Container

15.2.3 Additional Properties:Dynamic Cache service

This page allows you to specify settings for the Dynamic Cache service of this server. There is also a monitor that you can use to monitor you dynamic cahce settings. It is an enterprise applicaiton that you will have to install. It can be found at install_root/installableApplications directory. The application is named CacheMonitor.ear. The following options are configurable for the dynamic cache service.

- Startup state: Specifies whether the dynamic cache is enabled.
- Cache size: Specifies a positive integer as the value for the maximum number of entries the cache holds.
- Enter the cache size value in this field. Values are usually in the thousands, with no set maximum or minimum.
- Default priority: Specifies the default priority for cache entries, determining how long an entry stays in a full cache. The dafault value is 1.
- Disk offload: Specifies whether disk offload is enabled.
- By default, the dynamic cache only maintains, at most, the number of entries configured in memory. If new entries are created while the cache is full, the priorities configured for each cache entry, along with a least recently used algorithm, are used to remove entries from the cache. As an alternative, you can configure disk offload, and rather than removing the entries from memory, have them copied onto the file system (the location is configurable) for later.
- Cache replications:
- External Cache groups: This option allows you to create and modify external cache groups for the purpose of managing cache external to the application server.The external cache group name needs to match the “externalcache” property as defined in the servlet or JSP’s cachespec.xml file.

  When external caching is enabled, the cache matches pages with its URIs and pushes matching pages to the external cache. The entries can then serve from the external cache, instead of the application server. This ability creates a significant savings in performance

  – Create a new external cache group

  – Specify filters and preferences

Figure 15-9 Dynamic Caching Service

Author Comment: Currently looking for a sample that uses Dynacache so that we can use a screen capture for it. If there is no applicaiton installed the page will dipay and exception “null”, that the JSP could not find the cache properties it was looking for.

15.2.4 Additional Properties:Logging and Tracing

This page allows you to specify Logging and Trace settings for the server.

- Diagnostic Trace

  Configuration vs runtime: The trace service page provides two tabs for viewing configuraton and runtime tracing information. The configuration tab provides the following:

  – Enable Trace: This option provides the ability to turn trace on or off.

  – Trace Specification: This is the trace string that will be used to determine what to trace. Once the trace string as been supplied, you will need to provide the location where the trace will be dumped. You can specify the trace string by simply typing it into the text area, or you can choose “modify” and this will lauch a GUI tool for specifying the trace. As a result of using the GUI your trace string will be built for you.

– Trace Output: This identifies how you want your trace information saved. You can choose to hold the trace in memory by selecting a buffer and setting the size of that buffer. You can also choose to save the trace data to a file. You can select the name and location of the file, the size of the file, and the amount of historical/back up files you would like to create for the trace.

– Trace Output format: You have three options for the format of the trace file. You can choose basic,advanced and log analyzer.

  - The *Basic* (Compatible) selection preserves only basic trace information. Select this option to minimize the amount of space taken up by the trace output.
  - The *Advanced* selection preserves more specific trace information. Select this option to see detailed trace information for use in troubleshooting and problem determination.
  - The *Log Analyzer* selection preserves trace information in a format that is compatible with the Log Analyzer tool. Select this option if you want to use the trace output as input to the Log Analyzer tool. The benefit to using the Log Analyzeer tool is that you can take advantage of the symptom database that IBM WebSphere provides to help in problem determination and isolation.

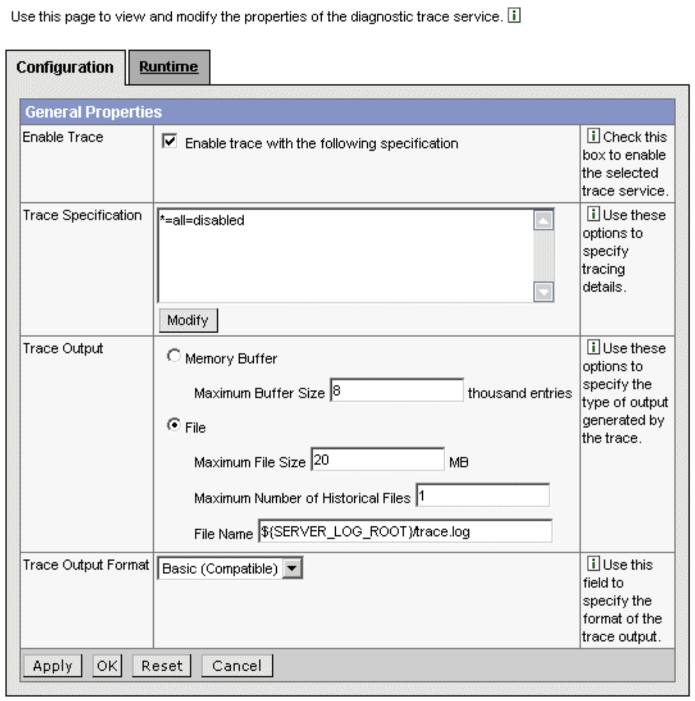
The runtime tab provides the ability to configure what you want done with trace information at runtime. The runtime tab provides the following:

– Save Trace

– Trace Specification

– Trace Output

You can use the runtime tab to make dynamic trace changes to a server that is running. The configuration tab will be what the server uses at startup.


Use this page to view and modify the properties of the diagnostic trace service.

General Properties

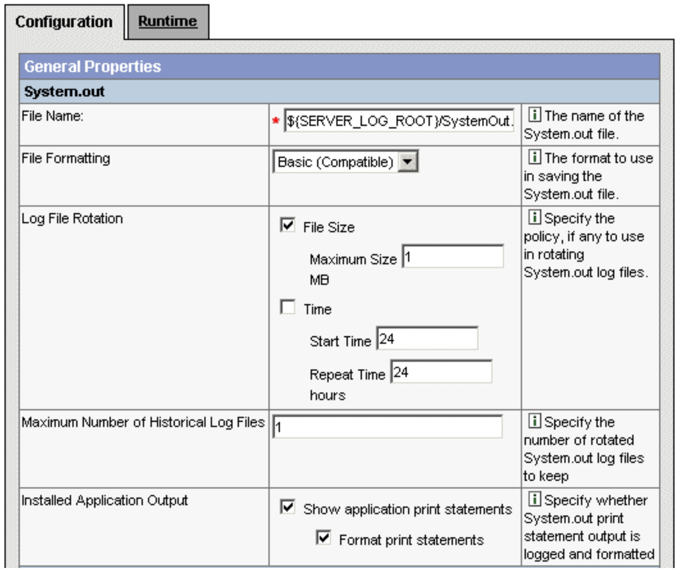
Enable Trace	☑ Enable trace with the following specification	Check this box to enable the selected trace service.
Trace Specification	*=all=disabled Modify	Use these options to specify tracing details.
Trace Output	○ Memory Buffer Maximum Buffer Size 8 thousand entries ● File Maximum File Size 20 MB Maximum Number of Historical Files 1 File Name ${SERVER_LOG_ROOT}/trace.log	Use these options to specify the type of output generated by the trace.
Trace Output Format	Basic (Compatible)	Use this field to specify the format of the trace output.

Figure 15-10 Diagnostic Trace Settings

- JVM Logs

  This page allows you to view and modify the settings for the JVM's System.out and System.err logs. The information logged to these files will include application specific error and output messages.There are two tabs on this page. The first tab is configuration. It contains information on how to set the file name, the format of the file, the rotation, how many historical files to save, and well as whatto log about application installation. You can do this for both system.out and system.err.

  The next tab on this page is the Runtime tab. Once the files have been created, you can use the runtime tab to view the contents of each file. If you have not yet created these files the dropdown list will say "could not find file".



System.out

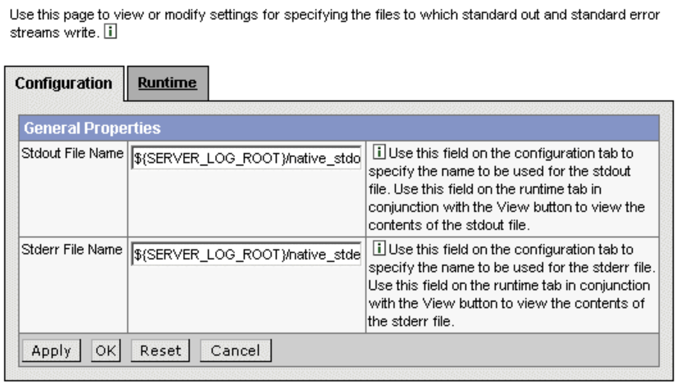
File Name:	* ${SERVER_LOG_ROOT}/SystemOut.	The name of the System.out file.
File Formatting	Basic (Compatible)	The format to use in saving the System.out file.
Log File Rotation	✓ File Size Maximum Size 1 MB ☐ Time Start Time 24 Repeat Time 24 hours	Specify the policy, if any to use in rotating System.out log files.
Maximum Number of Historical Log Files	1	Specify the number of rotated System.out log files to keep
Installed Application Output	✓ Show application print statements ✓ Format print statements	Specify whether System.out print statement output is logged and formatted

Figure 15-11 JVM Logging for System.out

– Process Logs: Process logs allow you to manage the output and errors from the server process. This type of information will be associated with the server, not the applications running on that server. These logs are process oriented. The output will be based on what the system considers a process.

  Standard out file name: allows you to specify the native file name to hold the process specific output messages.

  Standard err file name: allows you to specify the native file name for the process specific error messages.


Use this page to view or modify settings for specifying the files to which standard out and standard error streams write.

Stdout File Name	${SERVER_LOG_ROOT}/native_stdo	Use this field on the configuration tab to specify the name to be used for the stdout file. Use this field on the runtime tab in conjunction with the View button to view the contents of the stdout file.
Stderr File Name	${SERVER_LOG_ROOT}/native_stde	Use this field on the configuration tab to specify the name to be used for the stderr file. Use this field on the runtime tab in conjunction with the View button to view the contents of the stderr file.

Figure 15-12 Process Logs

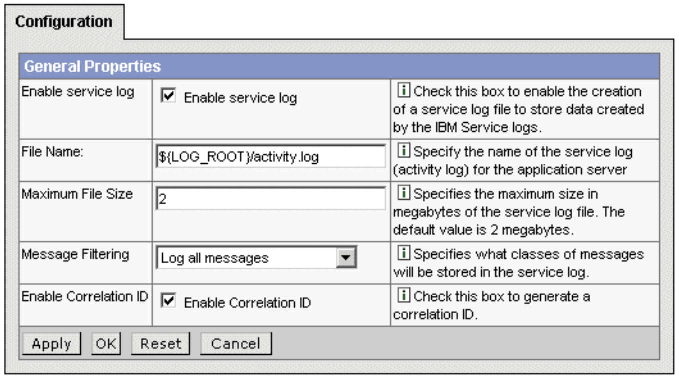
– IBM Service Logs

The IBM service log is also known as the activitiy log. The activity log captures events and dispays a history of Application Server activities. Some of the entires in the log are informational, while others reports on system exceptions, such as CORBA or communication exceptions.

The activity log is a binary log file that can be viewed using the Log Analyzer tool that ships with WebSphere. You can choose to enable or disable this logging service, where to place the file, the maximum size of the file, what kind of messages to you want to log, or enable a correlation id. The correlation Id can be used to correlate activity to a particular client request, or correlate activities on multiple application servers.

Note: You can use remote file systems to manage your log files. However, the Log Analyzer tool cannot browse to remote files. Log Analyzer is discussed in...

Author Comment: Do we discuss troubleshooting/ Log Analyzer anywhere, if so... what chapter is it in? Or should I add a section called “Trouble shooting the application server“ here



General Properties		
Enable service log	✓ Enable service log	Check this box to enable the creation of a service log file to store data created by the IBM Service logs.
File Name:	${LOG_ROOT}/activity.log	Specify the name of the service log (activity log) for the application server
Maximum File Size	2	Specifies the maximum size in megabytes of the service log file. The default value is 2 megabytes.
Message Filtering	Log all messages	Specifies what classes of messages will be stored in the service log.
Enable Correlation ID	✓ Enable Correlation ID	Check this box to generate a correlation ID.

Figure 15-13 IBM Service Logs

15.2.5 Additional Properties:Message Listener Service

This page provides the configuration for the Message Listener Service.This service provides the Message Driven Bean (MDB) listening process, whereby MDBs are deployed against ListenerPorts that define the JMS destination to listen upon. These Listener Ports are defined within this service along with settings for its Thread Pool.

General properties

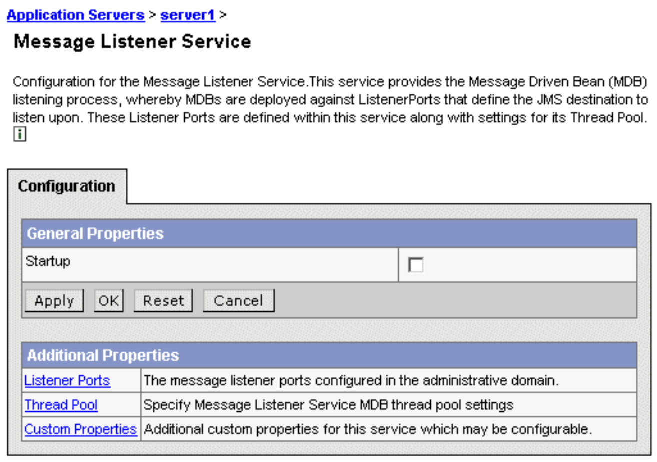
Startup: This checkbox specifies whether the server will attempt to start the Message Listener service automatically when the server starts. If the selection is cleared the message listener service is not started automatically. If message-driven beans are to be used on the application server, the system administrator must start the service manually or select this property then restart the application server.

Additional properties

Listener Ports:These are for Message Driven Beans to listen upon for messages. Each port specifies the JMS Connection Factory and JMS Destination that an MDB,deployed against that port,will listen upon.

Thread Pools: This option allows the administrator to create a thread pool that will be available for use by the beans at runtime. This pool allows components of the server to reuse threads to eliminate the need to create new threads at runtime. Creating new threads is typically a time and resource intensive operation. This dialog goves you the ability to set up the thread pool by assigning a range of threads the pool wil be able to create, when the pool should timeout inactive threads, and whether or not you want the pool to create new threads, in the case that it has more requests that the maximum size allows.

Custom Properties: Here you can supply custom properties that you would like accessible programmatically to your messaging beans. By supplying the properties here, the context object in the JVM will allow you beans operating under this context to access these properties through access methods.


Application Servers > server1 >

Message Listener Service

Configuration for the Message Listener Service.This service provides the Message Driven Bean (MDB) listening process, whereby MDBs are deployed against ListenerPorts that define the JMS destination to listen upon. These Listener Ports are defined within this service along with settings for its Thread Pool.

Configuration

General Properties	
Startup	☐

Additional Properties	
Listener Ports	The message listener ports configured in the administrative domain.
Thread Pool	Specify Message Listener Service MDB thread pool settings
Custom Properties	Additional custom properties for this service which may be configurable.

Figure 15-14 Message Listener Interface

15.2.6 Additional Properties:ORB Service

This page allows you to specify settings for the Object Request Broker Service.

General properties

The configuration tab list the general setting you will want to configure for the ORB service running on WebSphere. The following lists the attributes that you can select:

- Request timeout: This option should be used to specify the number of seconds you want the ORB service to wait before timing out a request message This value defaults to 180.
- Request retries count: this value specifies the number of times that the ORB attempts to send a request if the server fails.The ability to retry a request can enable application recovery from transient network failures. This value defaults to 1.

- Request reties delay. this value specifies the number of milliseconds to wait between retries. This value defaults to 0.
- Connection cache maximum: this value is used with the connection cache mimimum to set a range that the ORB service can use to keep a certain number of connection available in the cache. The maximum is the largest number of connections allowed to occupy the connection cache for the service. This value defaults to 240.
- Connection cache minimum:This attribute specifies the number of connections allowed to occupy the connection cache. The minimum value defaults to 100.
- ORB Tracing: this option allows you to enable the trace service at the ORB level. This will provide trace information on GIOP messages generated by the ORB service. Many ORB exceptions may be difficult to isolate. This tracing option will help you isolate those situations where you have recieved a GIOP message and need to troubleshoot the problem.

  This setting effects two system properties, com.ibm.CORBA.Debug, and com.ibm.CORBA.CommTrace. By deafult this option is not enabled, if you choose to enable it you will have to ensure that the above system properties have been set to true. If they are not both set to true you will not be able to trace the GIOP messages.
- Locate request timeout: This value specifies the number of seconds that a LocateRequest message will wait for a response.

  The LocateRequest message enables clients to determine whether an object reference is known, whether the server in question can process a request to this object, and if not, the server to which requests for this object are made. The information obtained by LocateRequest is also provided by Request, but in the case of a Location Forward situation, it enables clients to avoid a potentially lengthy transmission of parameters associated with a particular request.
- Force-tunnel:This option controls how the client ORB attempts to use HTTP tunneling. The values of this optional property can be:
  - Always: Use HTTP tunneling immediately, without trying TCP connections first.
  - Never: Disable HTTP tunneling. If a connection fails, a CORBA system exception will be thrown. (COMM_FAILURE)
  - WhenRequired:Use HTTP tunneling if the TCP connections fail.
- Tunnel agent URL: This must be specified if the force-tunnel option as chosen to use HTTP tunneling. The URL must be properly formatted for the tunneling to work. An example of the proper format is below

`http://w3.mycorp.com:81/servlet/com.ibm.CORBA.services.IIOPTunnelServlet`

or, for applets, `http://applethost:port/servlet/com.ibm.CORBA.services.IIOPTunnelServlet`

- Pass by reference: When parameters are passed locally, they are passed "by value" using the standard IIOP copy semantics. Use this property to select "pass by reference" instead. Depending on your application design, using pass by reference might introduce side effects, such as unexpected or unpredictable behavior. Use this option with caution.

Additional Properites

Thread Pools

Custom properties

- com.ibm.CORBA.ConnectionInterceptorName

Author Comment: I think we should have an explanation of each of these cutsom props and why they might be needed

- com.ibm.CORBA.FragmentSize
- com.ibm.CORBA.RasManager
- com.ibm.CORBA.WSSSLClientSocketFactoryName
- com.ibm.CORBA.WSSSLServerSocketFactoryName
- com.ibm.CORBA.enableLocateRequest
- javax.rmi.CORBA.UtilClass

15.2.7 Additional Properties:Administration Services

This page allows you to specify various settings for administration facility for this server, such as administrative communication protocol settings and timeouts.

15.2.8 Additional Properties:Diagnostic Trace Service

This page allows you to view and modify the properties of the diagnostic trace service.

15.2.9 Additional Properties:Debugging Service

This page allows you to specify settings for the debugging service, to be used in conjunction with a workspace debugging client application.

15.2.10 Additional Properties:IBM Service Logs

This page allows you to configure the IBM service log, also known as the activity log.

15.2.11 Additional Properties:Custom Services

This page allows you to define custom service classes that will run within this server and their configuration properties.

15.2.12 Additional Properties:Server Components

This page provides additional runtime components which are configurable.

15.2.13 Additional Properties:Process Definition

A process definition defines the command line information necessary to start/initialize a process.

15.2.14 Additional Properties:Performance Monitoring Service

This page allows you to specify settings for performance monitoring, including enabling performance monitoring, selecting the PMI module and setting monitoring levels.

- End Points: Configure important TCP/IP ports which this server uses for connections.
- Classloader: Classloader configuration
- Server Security: To change the security configuration at the server level, modify the attributes from the corresponding links below. To revert back to the cell defaults for a specific section, select the ’Use Cell Security’, ’Use Cell CSI’, or ’Use Cell SAS’ button for that section. Hit the Save link above to persist the changes at the server level.

15.3 Managing JMS servers

15.4 Managing clusters

Config - workload weight

After installing IBM WebSphere Application Server Network Deployment, you need to use the addnode command to set up your first application server.The main purpose of the **addnode** tool is to create a node agent for the new node and incorporate the new node into the cell. Once the node has been added you can use the administrative console create application servers. An application server is hosted on a node, that is why before you can create manage applications servers you must create a node for the application server to be associated with.The default server that is created is server1.

15.5 The cluster toplogy

16

Configuring the Web server interface

In this chapter we introduce/provide/describe/discuss ...

In this chapter:
In this chapter, the following topics are discussed/described:
This chapter provides/describes/discusses/contains the following:

- ...
- ...
- ...
- Sample level 2 “n.n” chapter heading
  (created by **Special > Cross-Reference > Format: Head > Insert**)
- Sample next level 2 heading

17

Configuring session management

This chapter dicusses HTTP session support in WebSphere Application Server V5.0. 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 a 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.

17.1 Version 4.0 vs. Version 5.0 session management

The following improvements have been made to session management of WebSphere V5.0:

- Servlet 2.3 support: The new features are:

Author Comment: are the servlet 2.3 features directly related to session management?

  - Servlet Filtering
  - Application Life-cycle Listeners and events
  - Enhanced Internationization
  - Overall API Changes
- Session manager configuration enhancement: In WebSphere V4.0, session manager configuration is at the application server level. In WebSphere V5.0, it can be defined at the following levels:
  - Application Server level
  - Application level
  - Web module level

  We will discuss this in “Session manager configuration” on page 553.
- Session scope enhancement

  As per the Servlet 2.3 specification, session scope is defined per Web application. In WebSphere 5.0, the IBM extensions provide the option to have session scope defined per Enterprise application. We will discuss this in s“Session scope” on page 556.
- Session enhancements
  - Session affinity has been improved (“Session affinity” on page 570).
  - Access to the session can be serialized (“Advanced settings for session management” on page 567).
  - There is an option to propagate exceptions back to servlet/JSP or not (“Advanced settings for session management” on page 567)
- Session Persistence

  There is a new feature, WebSphere Internal Messaging, which is in-memory replication of HTTP session state between clustered Web application servers. This is discussed in “Persistent session management” on page 576.

Note: This new feature is only available for WebSphere Network Deployment environments.

- Session counters in Tivoli performance viewer: New session counters are added in the runtime to analyze the session manager runtime. See Chapter 21, “Monitoring and tuning your runtime environment” on page 839.

Author Comment: need a specific link when the monitoring chapter is done.

17.2 Session manager configuration

In WebSphere 4.0, session management configuration was at the application server level, meaning, all applications used the same session manager settings. When an application would benefit from customized settings, such as session timeout, persistence, etc., or from using its own database for persistence to reduce the load generated by multiple applications, the only way to achieve this was by putting it on a separate application server was needed.

In WebSphere V5.0, this is no longer a problem. Session management can be defined at the following levels:

- Application server: this is the default level. Configuration at this level is applied to all Web modules within the server.
- Application: Configuration at this level is applied to all Web modules within the application.
- Web module: Configuration at this level is applied only to that Web module.

17.2.1 Accessing session management properties

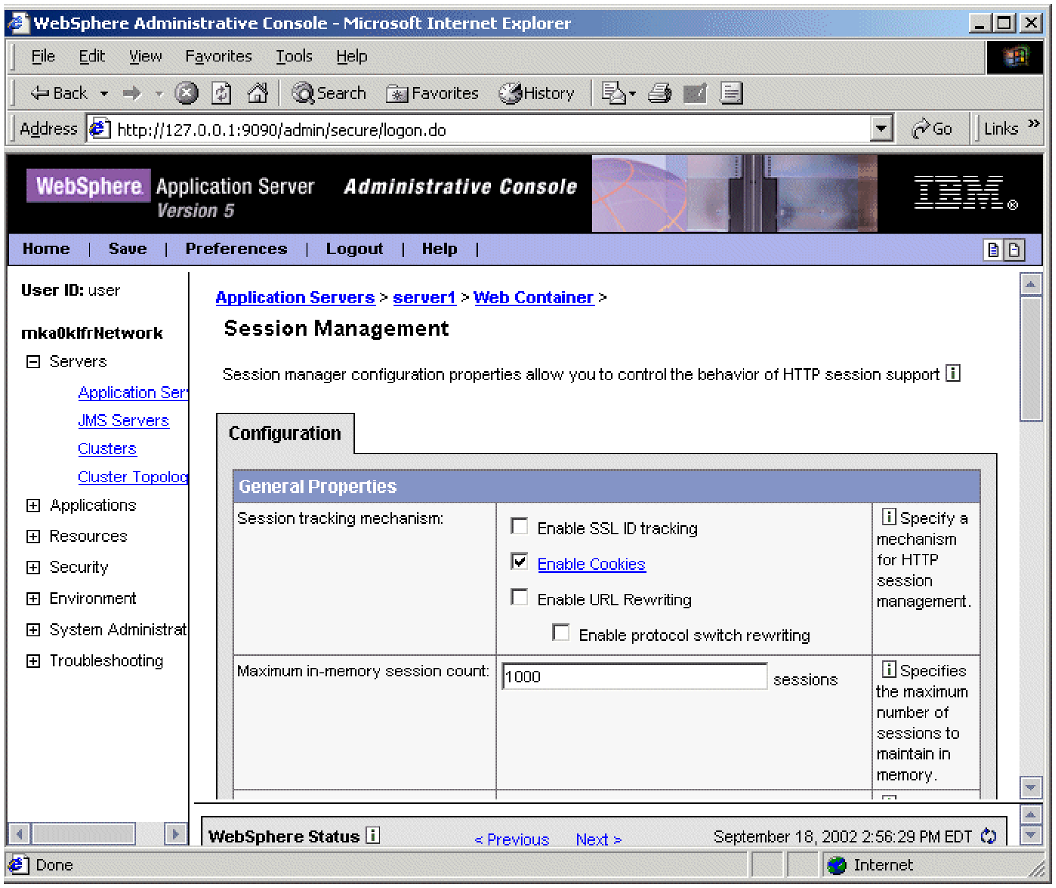
All configuration settings can be done using the administrative console.

Application server session management properties

To access session management properties at the application server level from the administrative console:

1. Click **Servers -> Application Servers**.
2. Click **<application server name>**. In this example, `server1` is selected.
3. Click **Web Container**.

4. Click **Session Management**. The window shown in Figure 17-1 on page 554 will appear.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: user
mka0klfrNetwork
Servers
Application Ser
JMS Servers
Clusters
Cluster Topolog
Applications
Resources
Security
Environment
System Administrat
Troubleshooting
WebSphere Status
< Previous
Next >
September 18, 2002 2:56:29 PM EDT
Done
Internet

Application Servers > server1 > Web Container >

Session Management

Session manager configuration properties allow you to control the behavior of HTTP session support

Configuration

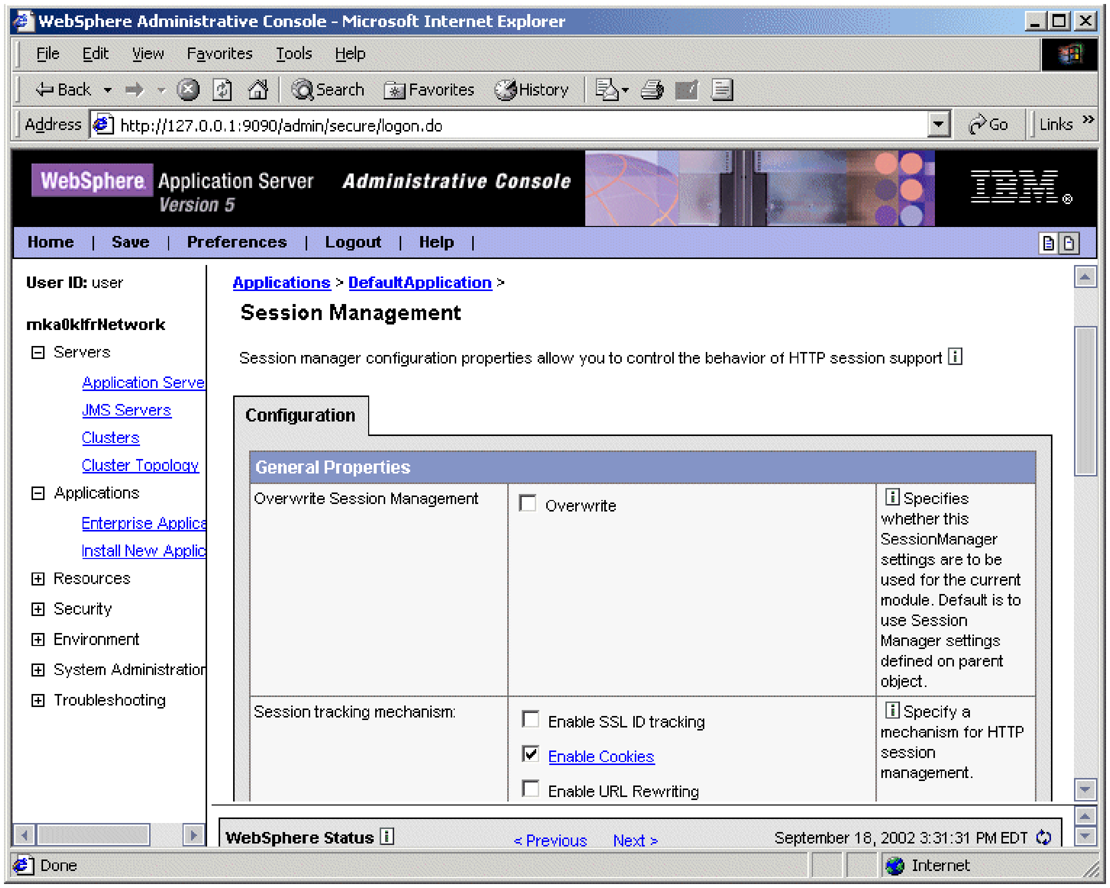
General Properties		
Session tracking mechanism:	☐ Enable SSL ID tracking ☑ Enable Cookies ☐ Enable URL Rewriting ☐ Enable protocol switch rewriting	Specify a mechanism for HTTP session management.
Maximum in-memory session count:	1000 sessions	Specifies the maximum number of sessions to maintain in memory.

Figure 17-1 Session Management window

Application session management properties

To access session management properties at the application level from the administrative console:

1. Click **Applications>Enterprise Applications**.
2. Click **<application name>**. In this example, `DefaultApplication` is selected.
3. Click **Session Management**. The window similar to one shown in Figure 17-2 on page 555 will appear.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: user
mka0klfrNetwork
Servers
Application Serve
JMS Servers
Clusters
Cluster Topology
Applications
Enterprise Applica
Install New Applic
Resources
Security
Environment
System Administration
Troubleshooting
WebSphere Status
< Previous Next >
September 18, 2002 3:31:31 PM EDT
Done
Internet

Applications > DefaultApplication >

Session Management

Session manager configuration properties allow you to control the behavior of HTTP session support

Configuration

General Properties		
Overwrite Session Management	☐ Overwrite	Specifies whether this SessionManager settings are to be used for the current module. Default is to use Session Manager settings defined on parent object.
Session tracking mechanism:	☐ Enable SSL ID tracking ☑ Enable Cookies ☐ Enable URL Rewriting	Specify a mechanism for HTTP session management.

Figure 17-2 Session Management Window at the application level

There is another property, Overwrite Session Management**,** in addition to the same properties at the application server level. Selecting this means that any properties configured at the application server level can be overwritten at the application level.

Web module session management properties

To access session management properties at the Web module level from administrative console:

1. Click **Applications -> Enterprise Applications**.
2. Click **<application name>**. In this example, `DefaultApplication` is selected.
3. Click **Web Modules**.

4. Click **<Web module name>**. In this example, `DefaultWebApplication.war` is selected.
5. Click **Session Management**. The window similar to one shown in Figure 17-2 on page 555 will appear.

The same properties appeared as at the application level. This means that any properties configured either at the application server level or at the application level can be overwritten at the Web module level by selecting the Overwrite Session Management property.

17.3 Session scope

The Servlet 2.3 specification defines session scope at the Web application level, meaning, session information can only be accessed by a single Web application. However, there may be times when there is a logical reason for multiple Web applications to share information, for example, a user name.

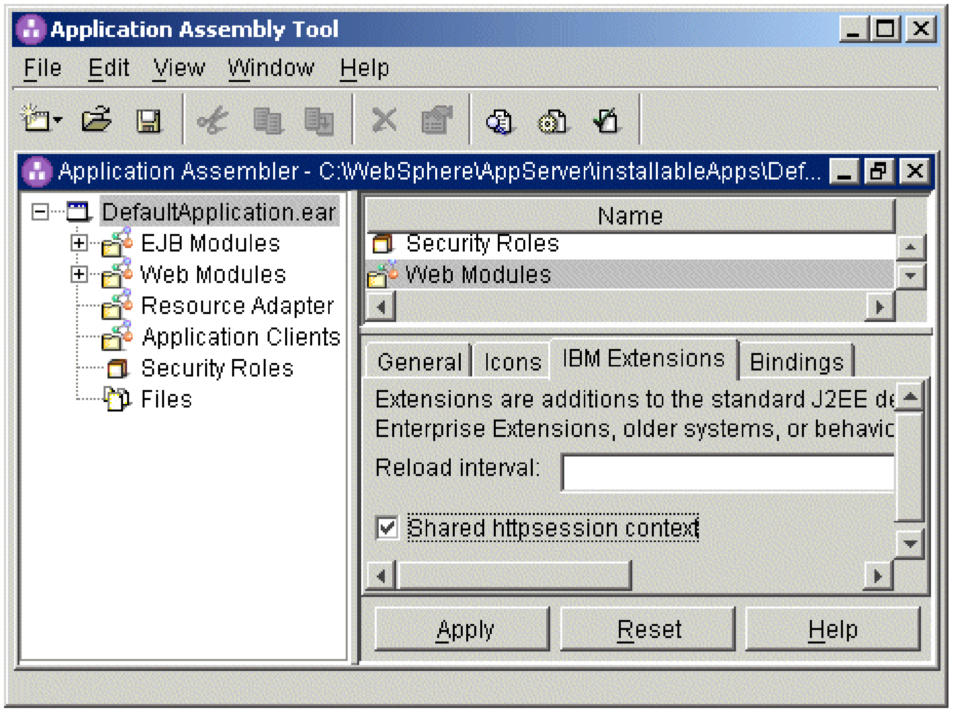
WebSphere V5.0 provides an IBM extension to the specification allowing session information to be shared among Web applications within an enterprise application. This option is offered as an extension to the application deployment descriptor. No code change is necessary to enable this option. This option is specified during application assembling.

Note: Since session information is shared within the enterprise application you can not use the Overwrite Session Management property at the Web module level when this option is selected.

To enable this option from application assembly tool (AAT):

1. Open ear file. In this example, `DefaultApplication.ear` file is opened.
2. Click the name of opened ear file.
3. Click **IBM Extensions** tab.
4. Select **Shared httpsession context**.

Author Comment: Can you specify this option from studio? Does this mean all apps that use this have to go thru the AAT?



Extensions are additions to the standard J2EE de
Enterprise Extensions, older systems, or behavic

Reload interval:

[✓] Shared httpsession context

Figure 17-3 Shared HTTP session context

5. Click **Apply** to reflect the change.

17.4 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.

17.4.1 Choosing a session tracking mechanism

WebSphere supports three approaches to track sessions:

- SSL session identifiers
- Cookies
- URL encoding/rewriting

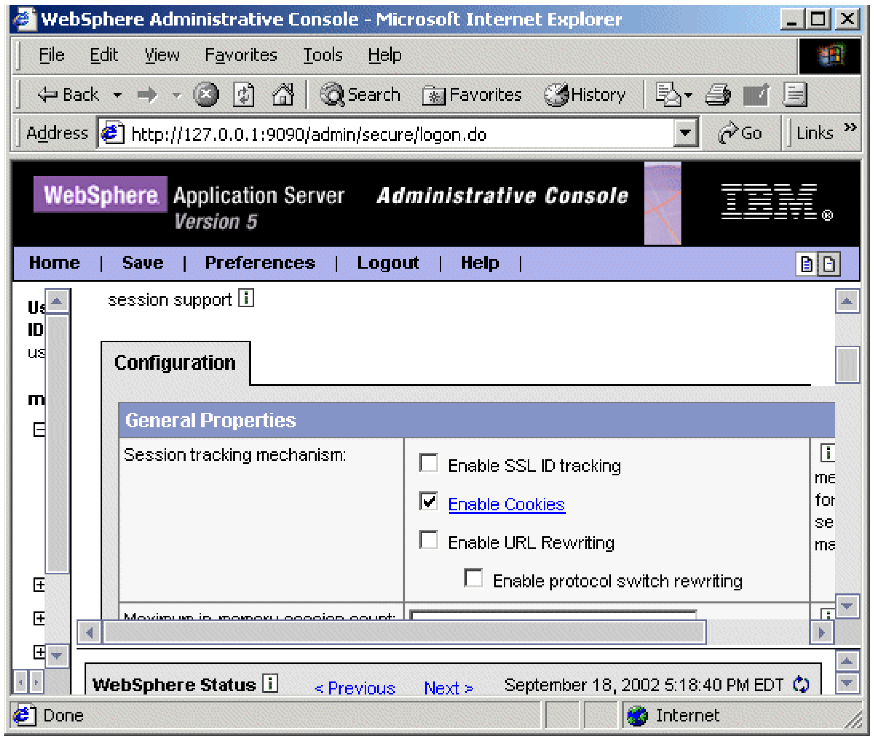
It is possible to select all three options for a Web application. If you do this:

- SSL session identifiers are used in preference to cookie and URL rewriting.
- Cookies 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 window at your preferred level, as described in “Accessing session management properties” on page 553.
2. Select the session tracking mechanism that you require.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
session support
Configuration
General Properties
WebSphere Status < Previous Next > September 18, 2002 5:18:40 PM EDT
Done
Internet

Session tracking mechanism:
- [ ] Enable SSL ID tracking
- [✓] Enable Cookies
- [ ] Enable URL Rewriting
  - [ ] Enable protocol switch rewriting

Figure 17-4 Session Tracking Mechanism window

3. Click **OK**.

4. Restart the application server or the cluster.

17.4.2 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 SUN ONE Web Server.

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 clustered 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.

17.4.3 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.

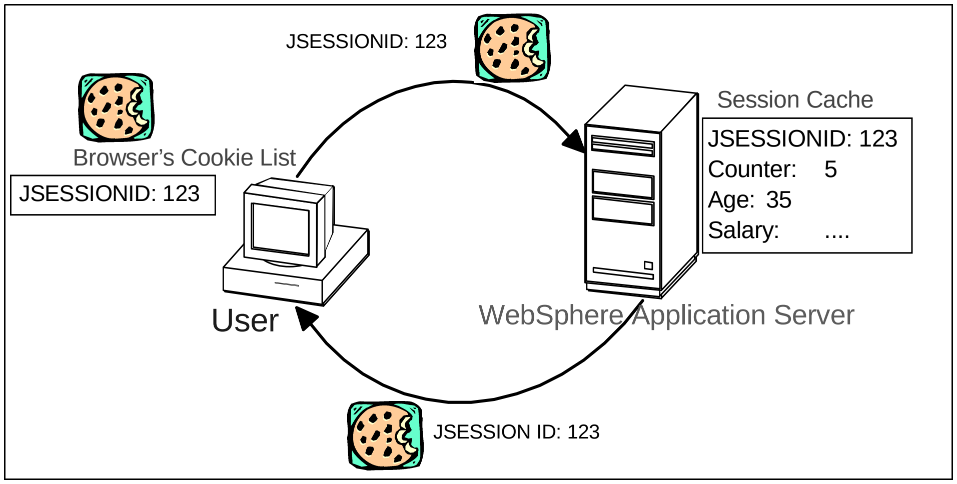


Figure 17-5 Cookie overview

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.

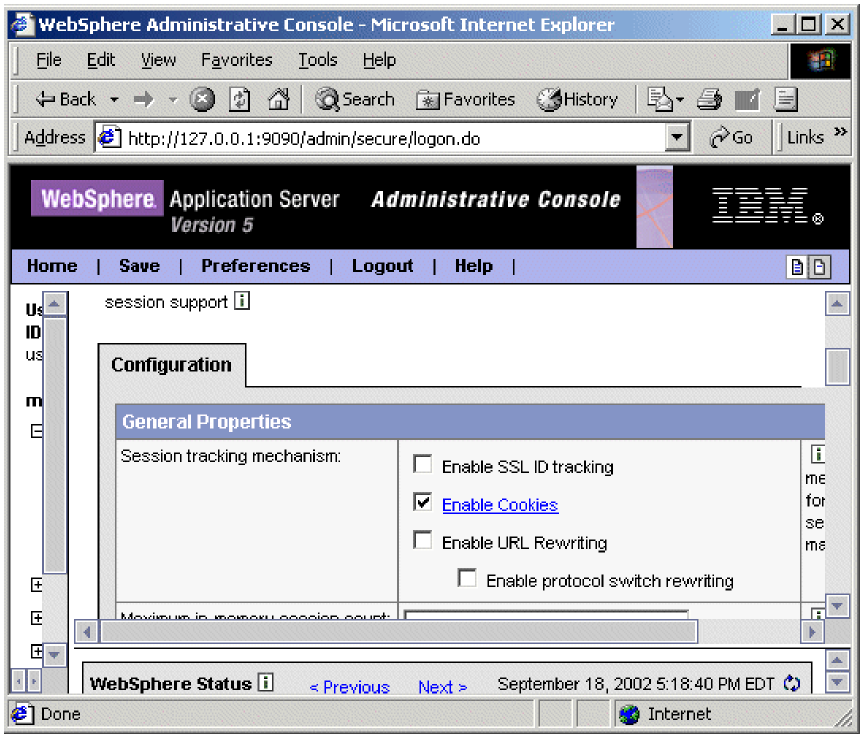
Cookie disadvantages

The main disadvantage with cookies is that some users, either by choice or mandate, disable them from within their browser.

Cookie settings

To configure session management using cookies from administrative console:

1. Open the Session Manager window at your preferred level.
2. Select **Enable Cookies** as the session tracking mechanism (check the box).

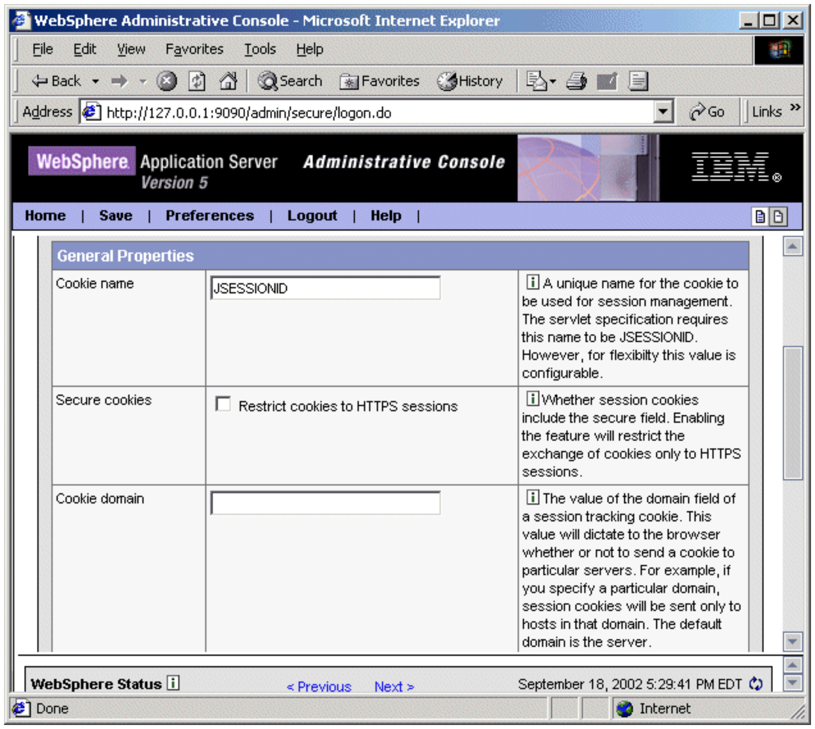

WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
session support
Configuration
General Properties
WebSphere Status < Previous Next > September 18, 2002 5:18:40 PM EDT
Done Internet

Session tracking mechanism:

- [ ] Enable SSL ID tracking
- [✓] Enable Cookies
- [ ] Enable URL Rewriting
  - [ ] Enable protocol switch rewriting

Figure 17-6 Session Tracking Mechanism

3. Click **Enable Cookies** (select the hot link).


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
WebSphere Status
< Previous Next >
September 18, 2002 5:29:41 PM EDT
Done
Internet

General Properties

Cookie name	JSESSIONID	A unique name for the cookie to be used for session management. The servlet specification requires this name to be JSESSIONID. However, for flexibilty this value is configurable.
Secure cookies	☐ Restrict cookies to HTTPS sessions	Whether session cookies include the secure field. Enabling the feature will restrict the exchange of cookies only to HTTPS sessions.
Cookie domain		The value of the domain field of a session tracking cookie. 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, session cookies will be sent only to hosts in that domain. The default domain is the server.

Figure 17-7 Cookie settings

Edit these properties to meet your needs. If **Enable URL Rewriting** is selected, **Enable protocol switch rewriting** can be also selected. See "Enable protocol switch rewriting" on page 565.

4. Click **OK**.
5. Click **OK**.
6. Restart the application server or the cluster.

The followings are the options for cookies:

- Cookie name

  The cookie name for session management should be unique. The default cookie name is JSESSIONID, which is required by the Servlet 2.3 specification for all cookie-based session IDs. However, this value can be configured for flexibility.

- Secure cookies

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.

Author Comment: so how is session management done on the http sessions?

- 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.

Author Comment: so how is session management done for the other domains?

Note: 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.

Author Comment: an example would help here .. ?

- Cookie maximum age

  The amount of time that the cookie will live in the client browser. There are 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
```

17.4.4 URL encoding/rewriting

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 17-1 shows a Web page link with URL encoding.

Example 17-1 Web page link with URL encoding

```
<a href="/store/catalog;$jsessionid=DA32242SSGE2">
```

When the user clicks this link (Example 17-1) 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 17-2, to encode the returning content.

Example 17-2 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 17-3.

Example 17-3 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 17-4.

Example 17-4 URL encoding in a JSP

```
<% response.encodeURL ("/store/catalog"); %>
```

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.

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.

17.5 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. 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, 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:

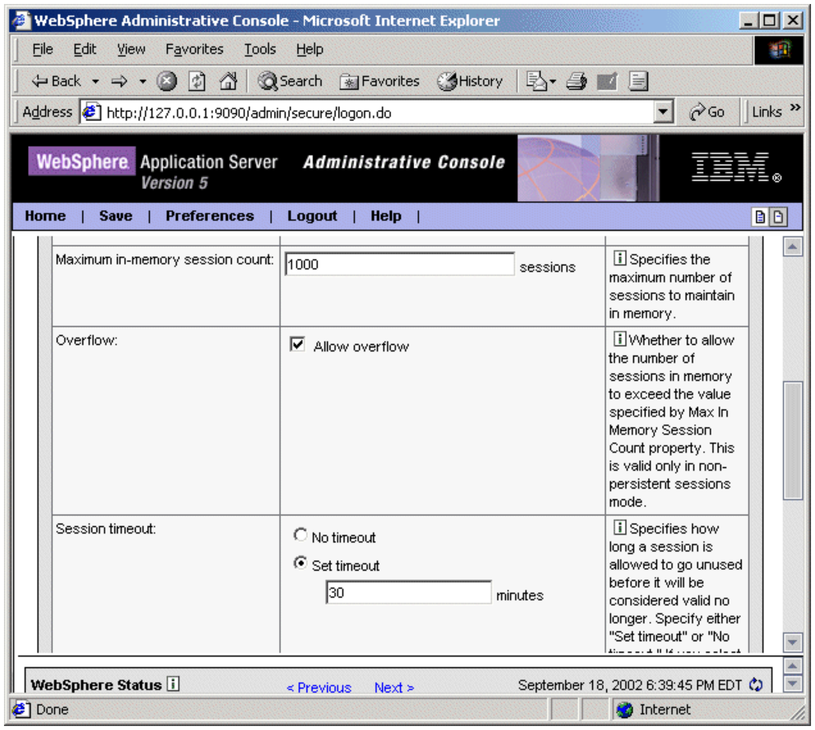
- The site receives greater traffic than anticipated, generating a large number of sessions held in memory.
- A malicious attack occurs against the site where a user deliberately manipulates their browser so the application creates a new session repeatedly for the same user.

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 21, "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.

17.6 Advanced settings for session management

The Session Management window, shown in Figure 17-8 on page 567 and Figure 17-9 on page 569, allows the administrator to tune a number of parameters that are important for both local or persistent sessions:


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |

Maximum in-memory session count:	1000 sessions	Specifies the maximum number of sessions to maintain in memory.
Overflow:	☑ Allow overflow	Whether to allow the number of sessions in memory to exceed the value specified by Max In Memory Session Count property. This is valid only in non-persistent sessions mode.
Session timeout:	○ No timeout ● Set timeout 30 minutes	Specifies how long a session is allowed to go unused before it will be considered valid no longer. Specify either "Set timeout" or "No

Figure 17-8 Session Management window (2)

- 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. Select **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.

- 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 17-5, the IBM HttpSession extension can be used to react if sessions exceed the maximum number of sessions specified when overflow is disabled.

Example 17-5 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…
}
```

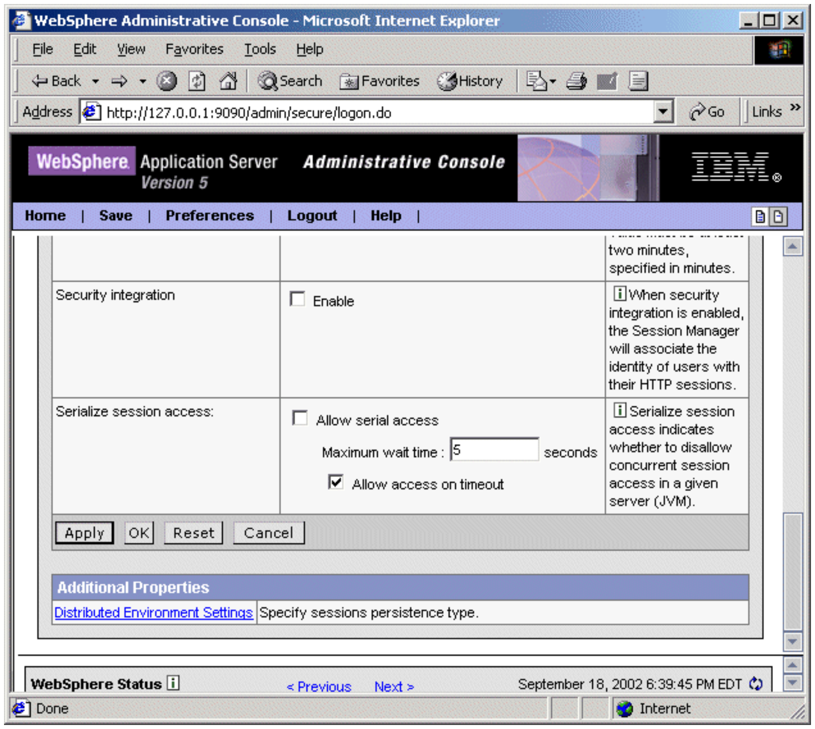
- Session timeout

  If **set timeout** is selected, when a session isn’t accessed for this many minutes it can be removed from the in-memory cache and if persistent sessions are used, from the persistent store. 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.

  If **no timeout** is selected, a session will be never removed from the memory unless explicite invalidation has been performed by the servlet. This may cause memory leak when the user closes the window without performing logout from the system. This option might be useful when sessions should be kept for a while until explicite invalidation has been done, for example, an employee leaves his company. In order to use this option, it is necessary to make sure that enough memory or space in a persistent store is kept to accommodate all sessions.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |

two minutes, specified in minutes.

Security integration	☐ Enable	When security integration is enabled, the Session Manager will associate the identity of users with their HTTP sessions.
Serialize session access:	☐ Allow serial access Maximum wait time : 5 seconds ☑ Allow access on timeout	Serialize session access indicates whether to disallow concurrent session access in a given server (JVM).

Additional Properties

Distributed Environment Settings	Specify sessions persistence type.

WebSphere Status < Previous Next > September 18, 2002 6:39:45 PM EDT
Done Internet

Figure 17-9 Session Management window (3)

- Security integration

  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.

- Serialize session access

  In WebSphere V4.0, sessions can be accessed concurrently. This causes that multiple threads can access to the same session at the same time and it may cause some inconsistency. It is the programmer's responsibility to serialize the access to the session to avoid the problem. In WebSphere V5.0, there is an option to provide a serialized access to the session in a given JVM. This ensures thread safe access when the session is accessed by multiple threads. No special code is necessary for using this option. This option is not recommended when framesets are used heavily because this option might affect performance.

The optional property **Maximum wait time,** the maximum amount of time a servlet request waits on an HTTP session before continuing execution**,** can be specified. The default is 2 minutes.

If **Allow access on timeout** option is true, multiple servlet requests that have timed out concurrently, execute concurrently. If it is false, servlet execution aborts.

17.7 Session affinity

The Servlet 2.3 specification requires that an HTTP session be:

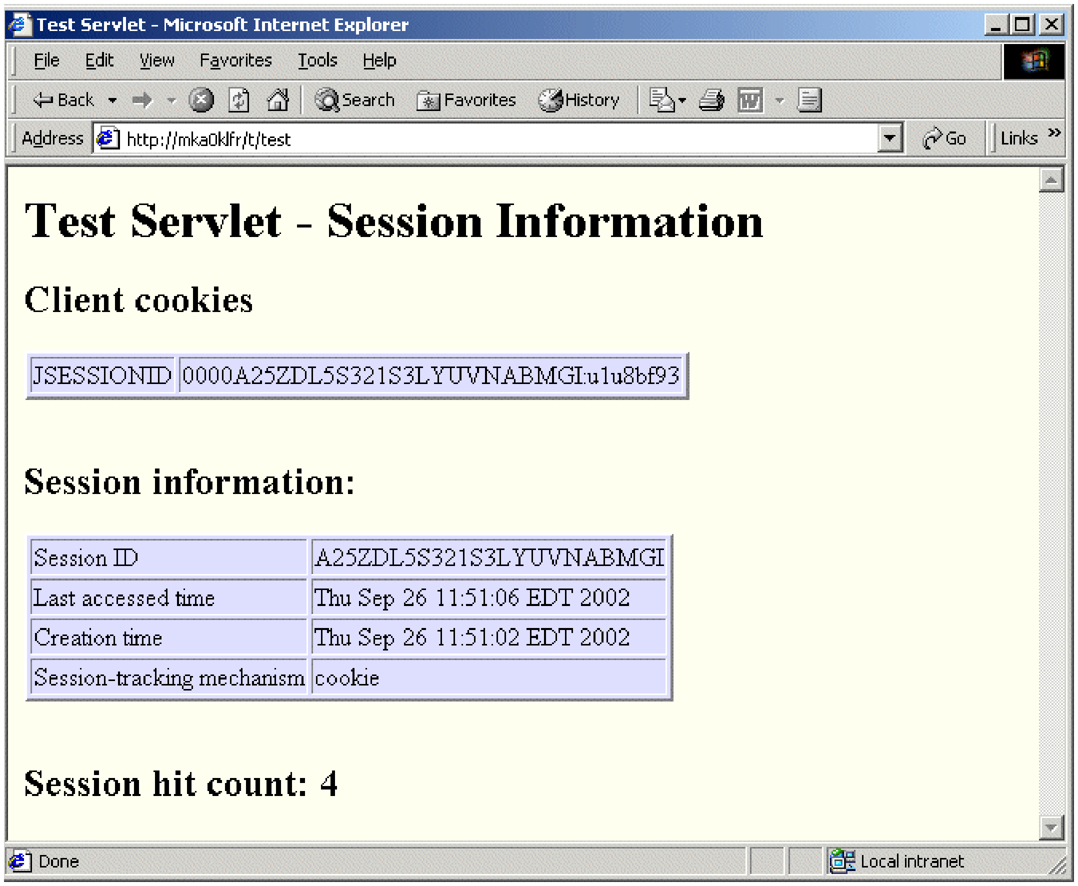
- Accessible only to the Web application that created the session. The session ID can be shared across Web applications, but not the session data.
- Handled by a single JVM for that application at any one time.

This means that in a clustered 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 cluster member fails or has been shut down. For more information see 17.7.1, "Session affinity and failover" on page 573.

Author Comment: reference to clones is outdated

WebSphere is able to assure that session affinity is maintained in the following way: 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 cluster member handling the session, and forwards the request.

This can be seen in Figure 17-10, 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 17-6, 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.


Test Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://mka0klfr/t/test
Go Links
Done
Local intranet

Test Servlet - Session Information

Client cookies

JSESSIONID	0000A25ZDL5S321S3LYUVNABMGI:u1u8bf93

Session information:

Session ID	A25ZDL5S321S3LYUVNABMGI
Last accessed time	Thu Sep 26 11:51:06 EDT 2002
Creation time	Thu Sep 26 11:51:02 EDT 2002
Session-tracking mechanism	cookie

Session hit count: 4

Figure 17-10 Session ID containing the server ID and cache ID

The JSESSIONID cookie can be divided into 4 parts: cache ID, session ID, separator, and clone ID. The following table shows the mappings of them based on the example in Figure 17-10. A clone ID is an ID of a cluster member.

Author Comment: another clone comment ..
=> (From Arihiro) In the XML file, the word CloneID is explicitely used.... So, I can’t change this here.

Table 17-1 cookie mapping

content	value in the example
Cache ID	0000
Session ID	A25ZDL5S321S3LYUVNABMGI
separator	:
Clone ID	u1u8bf93

Example 17-6 Server ID from plugin-cfg.xml file

```
<?xml version="1.0"?>
<Config>
    ......
    <ServerCluster Name="MyCluster">
        <Server CloneID="u1u8bf93" LoadBalanceWeight="2"
Name="MyClusterServer1">
            <Transport Hostname="9.24.104.124" Port="9082"
Protocol="http"/>
            <Transport Hostname="9.24.104.124" Port="9445"
Protocol="https">
                <Property name="keyring"
value="C:\WebSphere\AppServer\etc\plugin-key.kdb"/>
                <Property name="stashfile"
value="C:\WebSphere\AppServer\etc\plugin-key.sth"/>
            </Transport>
        </Server>
        <Server CloneID="u1u8bgj3" LoadBalanceWeight="2"
Name="MyClusterServer2">
            <Transport Hostname="9.24.104.124" Port="9083"
Protocol="http"/>
            <Transport Hostname="9.24.104.124" Port="9446"
Protocol="https">
                <Property name="keyring"
value="C:\WebSphere\AppServer\etc\plugin-key.kdb"/>
                <Property name="stashfile"
value="C:\WebSphere\AppServer\etc\plugin-key.sth"/>
            </Transport>
        </Server>
```

```
            <PrimaryServers>
                <Server Name="MyClusterServer1"/>
                <Server Name="MyClusterServer2"/>
            </PrimaryServers>
        </ServerCluster>
        ......
    </Config>
```

Note: Session affinity can still be broken if the cluster member 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 persistent store. So don't rely on the value of the session cookie remaining the same for a given session.

17.7.1 Session affinity and failover

Server clusters provide a solution for failure of an application server. Sessions created by cluster members in the server cluster share a common persistent session store. Therefore, any cluster member in the server cluster has the ability to see any user's session saved to persistent storage. If one of the cluster members fail, the user may continue to use their session information from another cluster member in the server cluster. This is known as failover. Failover works regardless of whether the nodes reside on the same machine or several machines. Please refer to Figure 17-11.

Author Comment: need to search whole chapter and fix references to clones

Author Comment: Need to add a reference to WLM red book.

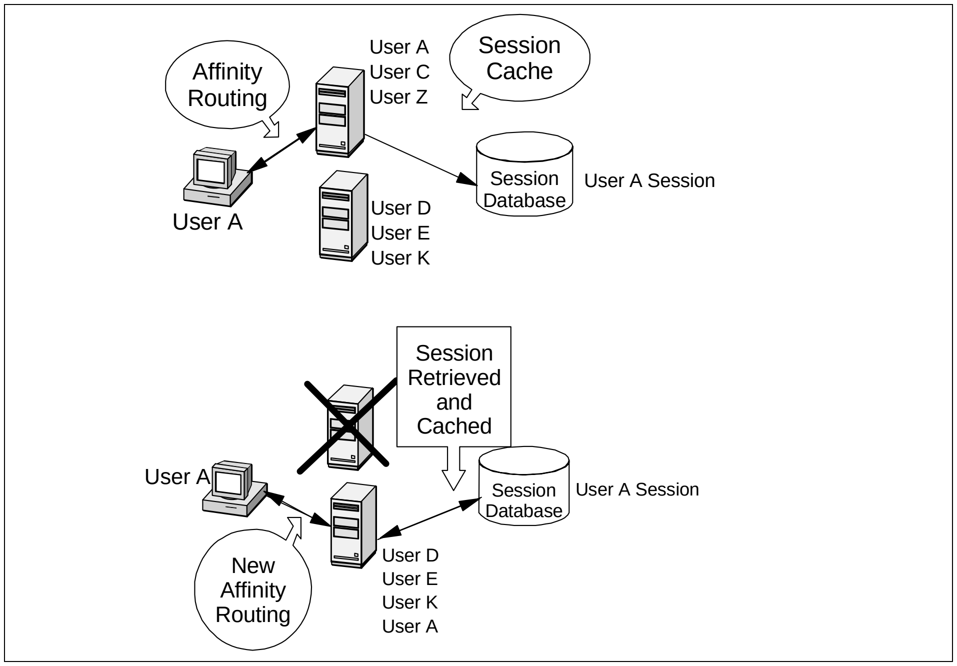


Figure 17-11 Session affinity and failover

> **Note:** According to the Servlet 2.3 specification, only a single cluster member may control/access a given session at a time.

After a failure, WebSphere redirects the user to another cluster member, and the user's session affinity switches to this replacement cluster member as well. After the initial read from the persistent store, the replacement cluster member 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 cluster member that is randomly selected from the list of available cluster members. From then onwards, requests for that session will go to the selected cluster member. The requests for the session will go back to the failed cluster member when the failed cluster member 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 cluster member is recovering from a crash, there exists a window where concurrent requests for the same session could end up in different cluster members. This is because the Web server is multi-processed and

each process separately maintains its own retry timer value and list of available cluster members. The end result is that requests being processed by different processes may end up being sent to more than one cluster member after at least one process has determined that the failed cluster member is up.

To avoid or limit exposure due to this scenario, if your cluster members 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 cluster members.

2. A server overload occurring that can cause requests belonging to the same session to go to different cluster members. This can occur even if all the cluster members are up and running. For each cluster member, 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 cluster member is not available. This failure will be handled in the same way by the plug-in as an actual JVM crash. See for **XXX** setting retry timeout.

Author Comment: The reference to the chapter which includes the definition of plugin file is needed to inserted above. But it seems that there is no such chapter in this redbook......

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 21, “Monitoring and tuning your runtime environment” on page 839.

17.7.2 Server order based session affinity

In WebSphere V4.0, when a cluster member is marked dead, the plug-in directs the request to one of the available cluster members randomly. In this case, there is a chance that two concurrent requests for a session might end up in different cluster members. To avoid this in WebSphere V5.0, plug-in maintains the cluster member list in order and will pick the cluster member next in its list to avoid the breaking of session affinity. This is also useful in case of URL-rewriting where cluster member information does not get updated if rewritten url's aren't updated.

17.8 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 persistent store.

Administrators should enable persistent session management when:

- The user's session data must be recovered by another cluster member after a cluster member in a cluster 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 store, the administrator controls the number of sessions allowed in memory at any given time.

There are two ways to configure session persistence in WebSphere V5.0. In addition to the database persistence, a new feature, WebSphere Internal Messaging, which is an in-memory replication of session state between clustered Web applications, can be configured. See "WebSphere Internal Messaging" on page 580 for the details.

All information stored in a persistent session store 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 store.

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.

If database persistence is used, 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 17.8.9, “Support for single or multi-row schemas” on page 604 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 store. 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.

17.8.1 Database persistence

Enable database persistence

It is assumed in this section that the following tasks have already completed before enabling database persistence:

1. Create a session database
2. Create a JDBC provider
3. Create a data source

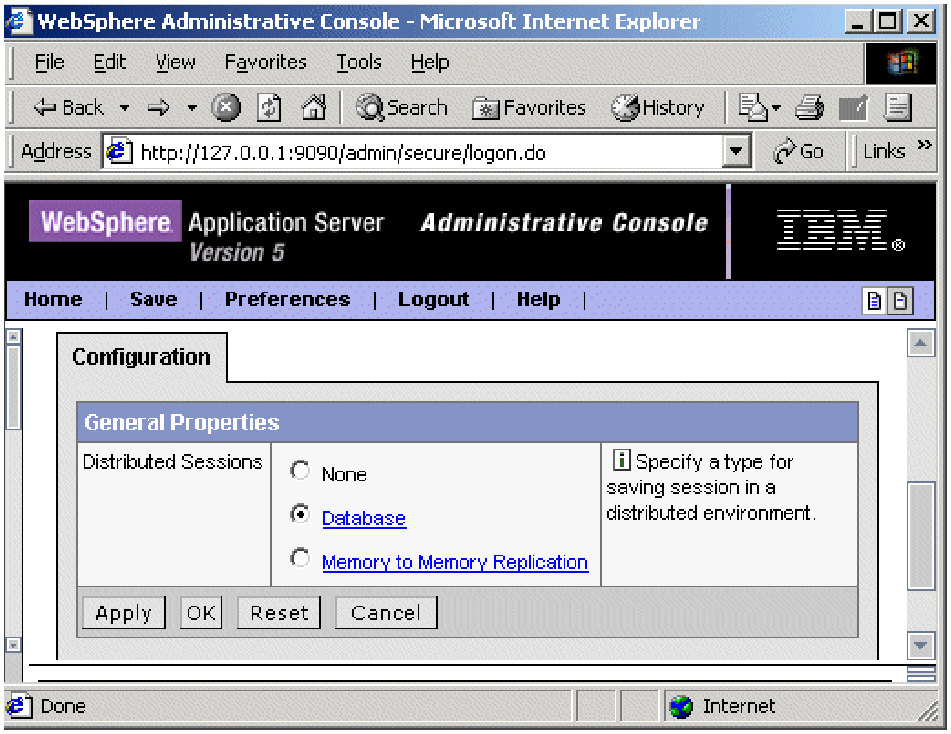
In this example, it is assumed that datasource JNDI name is `jdbc/Sessions`. See “JDBC and JDBC Resources” on page 611 for explanation of JDBC and JDBC resources, and “Creating a datasource” on page 617 for creating and configuring both a JDBC provider and its data source.

In order to enable database persistence:

Note: The following example illustrates the steps to enable database persistence at the application server level. In WebSphere V5.0, session management setting can be performed at the enterprise application level and the Web application level. See “Session manager configuration” on page 553.

Repeat the following steps for each application server.

1. Click **Servers -> Application Servers**
2. Click **<application server>**. An application server which will be configured should be clicked here.
3. Click **Web Container**
4. Click **Session Management**
5. Click **Distributed Environment Settings**
6. Select **Database** and click **Database**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
Configuration
General Properties
Apply OK Reset Cancel
Done
Internet

Distributed Sessions: None [✓ Database] Memory to Memory Replication

Specify a type for saving session in a distributed environment.

Figure 17-12 Distributed Environment Setting (database)

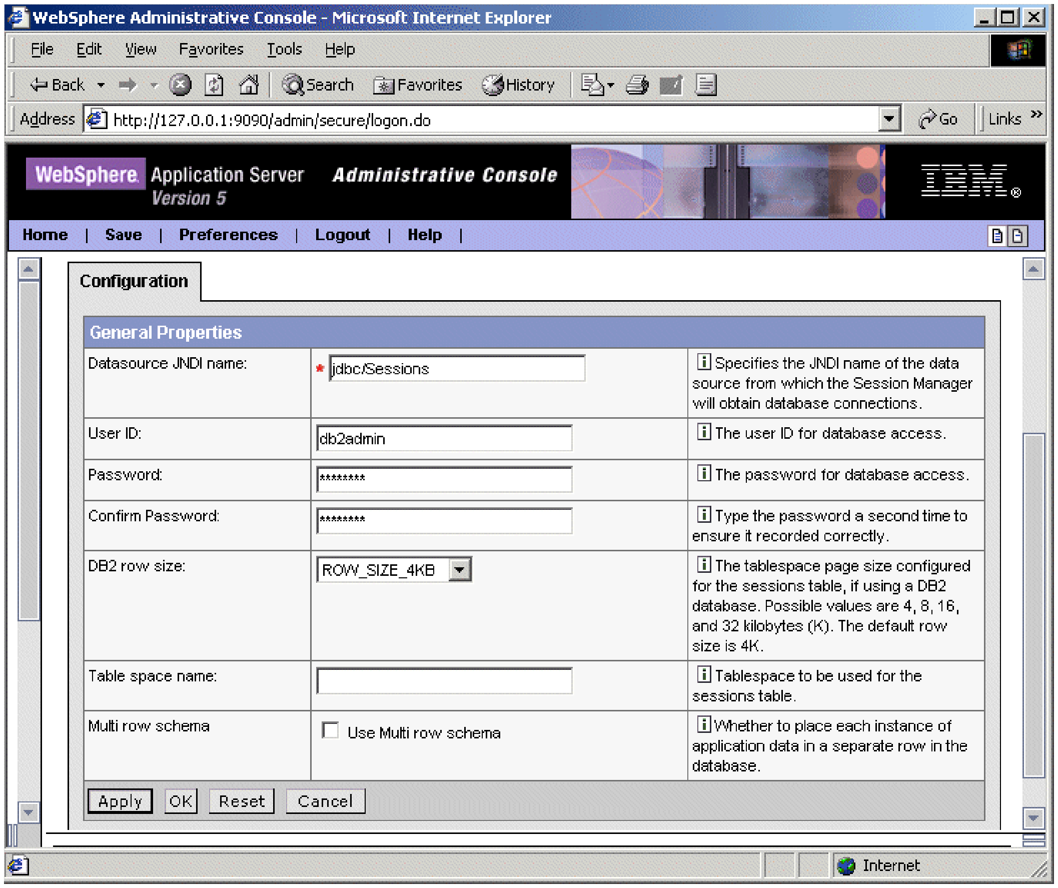
7. Enter values for properties if necessary:
   a. Enter Datasource JNDI name. The data source must be non-JTA enabled data source.
   b. If required, set the user ID, password, and confirm password.

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 17.8.8, “Using larger DB2 page sizes” on page 603 for more information.

d. If DB2 row size is other than 4KB, you are required to enter the name of table space. See “Using larger DB2 page sizes” on page 603.

e. If you intended using a multi-row schema, Select . See 17.8.9, “Support for single or multi-row schemas” on page 604 for more information.

8. Click **OK**


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
Configuration
General Properties
Apply OK Reset Cancel
Internet

Datasource JNDI name:	* jdbc/Sessions	Specifies the JNDI name of the data source from which the Session Manager will obtain database connections.
User ID:	db2admin	The user ID for database access.
Password:	********	The password for database access.
Confirm Password:	********	Type the password a second time to ensure it recorded correctly.
DB2 row size:	ROW_SIZE_4KB	The tablespace page size configured for the sessions table, if using a DB2 database. Possible values are 4, 8, 16, and 32 kilobytes (K). The default row size is 4K.
Table space name:		Tablespace to be used for the sessions table.
Multi row schema	☐ Use Multi row schema	Whether to place each instance of application data in a separate row in the database.

Figure 17-13 database

9. Click **Save**, **Save**

10. Restart the application servers.

Note: The persistent session database performs best if it is not shared with other database. This eliminates contention for resources, such as connections, which impacts performance.

17.8.2 WebSphere Internal Messaging

The WebSphere Internal Messaging is a new feature of the WebSphere V5.0. It enables to share the sessions between the application servers without using a database.

The figure in Figure 17-14 on page 581 shows basic topologies of both database persistent sessions and WebSphere Internal Messaging. Basic topology of WebSphere Internal Messaging is same as that of a database persistent sessions. In database persistent sessions, sessions are stored to and/or retrieved from a database by Web containers. Sessions created by an application server is retrieved from a database by another application server when the application server failed. This functionality achieves the failover and users can continue to use the applications with no aware of the application server failure. In WebSphere Internal Messaging, sessions are stored in the memory of an application server instead of in a database. It provides the same functionality as database persistent session does. Separate threads handle this functionality within an existing application server process. A new feature can be configured more flexibly, such as peer-peer and client-server. See “Topology” on page 582 for detail.

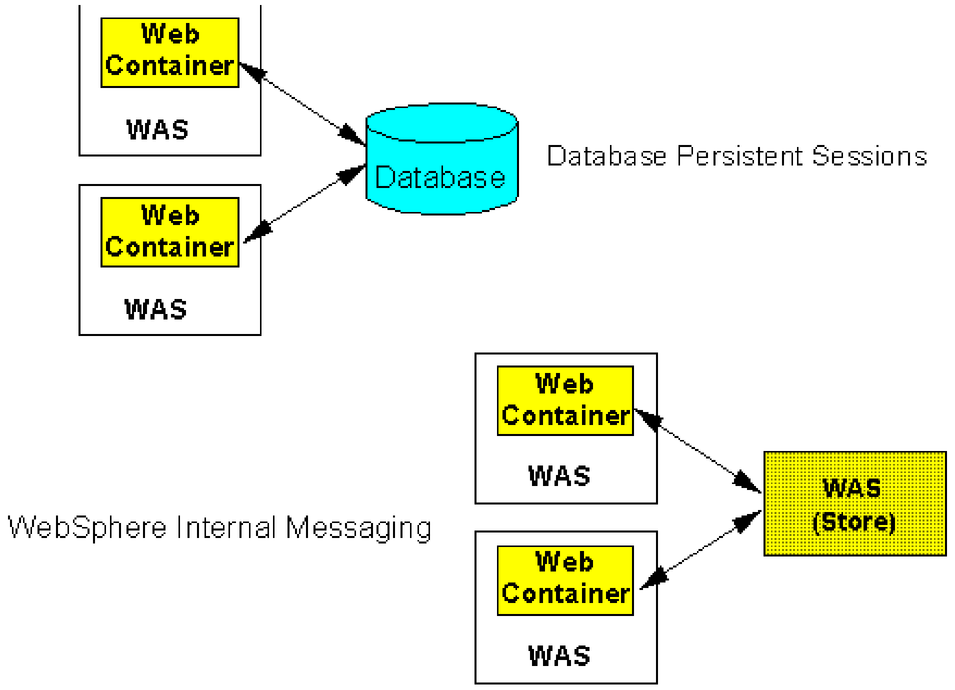


Figure 17-14 Persistent Sessions

There are some benefit over the use of a database for session persistence. There are overhead and cost of setting up and maintaining a real time, production database, such as preparing a machine, installing and configuring a database, starting and so on. SPOF(Single Point Of Failure) is a key issue of the session persistence and certain cost is required to solve this issue at the database level. The performance of the use of the WebSphere Internal Messaging is better than the use of a database.

All features available in database persistence is available in WebSphere Internal Messaging except DB2 variable row size and multi-row features, where those features are specific to a database.

Session can be either stored as bytes or an Object in persistent store. Storing as bytes removes the dependency on the classpath setting. See “Replication domain settings” on page 591 for detail.

Transferring sessions between application servers can be encrypted with DES or Trile DES. Regenerating the encryption key can be performed via adminconsole. It is recommended to regenerate the key at regular intervals to enhance security.

This is configured as the **Encryption Type** property in the window shown in Figure 17-17 on page 587.

Partitions can be set up to allow application servers to listen only on configured channels. Partitions reduce overhead of each application server because not all application servers need to store all sessions in the domain. See “Partioning” on page 594 for detail.

Note: This feature is only available in WebSphere Network Deployment.

Topology

The WebSphere Internal Messaging can be setup as peer-to-peer or client-server. Its default is peer-to-peer.

The figure shown in Figure 17-15 on page 583 is an example of peer-to-peer topology. Each application server stores sessions in its own memory. It also stores sessions to and retrieves session from other application servers. In other word, each application server acts as client by retrieving sessions from other application servers and each application server acts as server by providing sessions to other application servers.

The advantage of this topology is that no additional processes and/or products are required to avoid single point of failure. This reduces the time and cost for additional processes and/or products to configure and maintain.

One of the disadvantages of this topology is that it consumes large amount of memory resources by keeping all sessions in the domain. This is not useful when many users login the system at the same time. For example, assuming that a single session consumes 10KB and one million users have logged in the system. In this case, each application server consumes 10GB memory in order to keep all sessions in its own memory. This is independent to how many application servers exist in the domain. Other disadvantage is that keeping all sessions in each application server may have performance impact because every changes performed to sessions need to be sent to all application servers. Partitions can be used to avoid those problems. See “Partioning” on page 594 for detail about partitions.

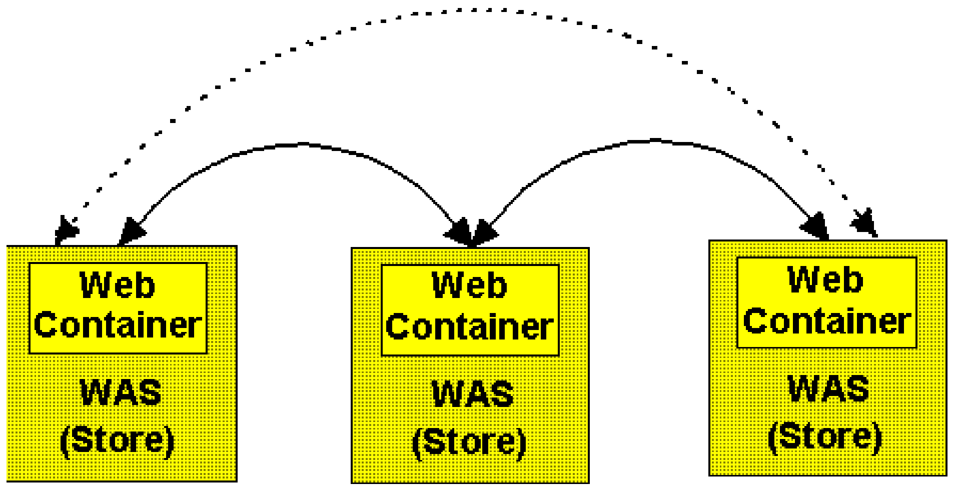


Figure 17-15 Example of Peer-to-Peer topology

The figure shown in Figure 17-16 on page 584 is an example of client-server topology. There are two types of application servers, client and server. Some application servers, which act as server, store sessions in their own memory and provide sessions to clients. Those application servers are dedicated replication servers, which just store sessions but do not respond to the users' requests. Other application servers, which act as client, store sessions to the servers and also retrieve sessions from the servers. Those application servers respond to the users' requests and store only sessions of the users with whom they interact. Configuration for client-server topology can be performed in the window shown in Figure 17-20 on page 591.

The advantage of this topology is that it clearly distinguishes the role of client and server, only the servers have to keep all sessions in their memory and only the clients have to interact with users. This reduces the consumption of memory resource of each application server. There are less performance impact since the changes to the sessions have to be sent only to the servers.

One of the disadvantages of this topology is that more application servers have to be configured and maintained in addition to the application servers which interact with users. This leads more time and cost is required for this topology. Other disadvantage is that a server can be single point of failure. It is recommended that multiple servers should be configured for the domain in order to avoid this problem. Since servers still have to maintain all sessions, there are some kinds of impact of performance and resource consumption. The partitions can be used to resolve these problems. See “Partioning” on page 594 for detail.

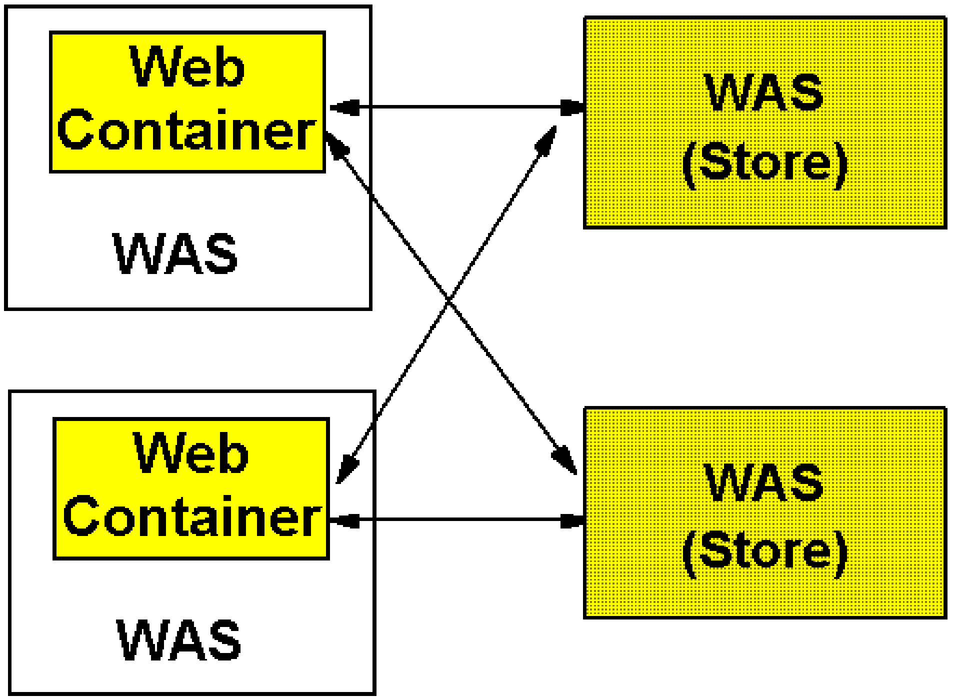


Figure 17-16 Example of Client Server topology

Replication domain and replicator entry

WebSphere Internal Messaging is built on small and fast publish/subscribe engine. It consists of two key elements, replication domain and replication entry. They are defined at the cell level.

Replicator domain defines the set of replicator processes that communicate with each other and shares a common publish/subscribe cluster. Multiple replication domain can reside in a cell.

Replicator entry, or simply replicator, runs inside an application server process and defines its basic settings, such as a communication port among replicators and a communication port with clients. A replication domain can contain multiple replicator entries.

The configuration of replication domain and replicator entry is written in multibroker.xml file. The example, shown in Example 17-7 on page 585, illustrates how a replication domain and replicator entries are defined. In this example, `MyClusterRepDomain` is a replication domain name, and `RepEntry1` and `RepEntry2` are replication entry names.

Example 17-7 an example of multibroker.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:multibroker="http://www.ibm.com/websphere/appserver/schemas/5.0/multibroker.xmi"
xmlns:ipc="http://www.ibm.com/websphere/appserver/schemas/5.0/ipc.xmi"
xmlns:multibroker.drsclient="http://www.ibm.com/websphere/appserver/schemas/5.0/multibroker.drs
client.xmi">
  <xmi:Documentation>
    <contact>WebSphere Application Server v5.0 Default Configuration Files v1.6
7/18/02</contact>
  </xmi:Documentation>
  <multibroker:MultibrokerDomain xmi:id="MultibrokerDomain_1" name="MyClusterRepDomain">
    <entries xmi:id="MultiBrokerRoutingEntry_1" brokerName="RepEntry1">
      <brokerEndPoint xmi:id="EndPoint_1" host="mka0klfr" port="1506"/>
      <clientEndPoint xmi:id="EndPoint_2" host="mka0klfr" port="1507"/>
    </entries>
    <entries xmi:id="MultiBrokerRoutingEntry_2" brokerName="RepEntry2">
      <brokerEndPoint xmi:id="EndPoint_3" host="mka0klfr" port="1508"/>
      <clientEndPoint xmi:id="EndPoint_4" host="mka0klfr" port="1509"/>
    </entries>
    <defaultDataReplicationSettings xmi:id="DataReplication_1" requestTimeout="5"
encryptionType="NONE" encryptionKeyValue="">
      <partition xmi:id="DRSPartition_1" size="10" partitionOnEntry="false"/>
      <serialization xmi:id="DRSSerialization_1" entrySerializationKind="BYTES"
propertySerializationKind="BYTES"/>
      <pooling xmi:id="DRSConnectionPool_1" size="10" poolConnections="false"/>
    </defaultDataReplicationSettings>
  </multibroker:MultibrokerDomain>
</xmi:XMI>
```

Enable WebSphere Internal Messaging

It is assumed in this section that the following tasks have already completed before enabling WebSphere Internal Messaging:

1. Create cluster

   In this example, it is assumed that `MyCluster` is a name of the cluster.
2. Add cluster members (application servers)

   In this example, it is assumed that `MyClusterServer1` and `MyClusterServer2` are names of the cluster members.
3. Install applications to the cluster
4. Update Web server plugin and copy the updated plugin file to Web server machine if necessary
5. Restart the web server

See for *WLM Red Book* how to create and configure cluster.

Author Comment: Need to put name and number of WLM Red Book above.

In order to enable WebSphere Internal Messaging:

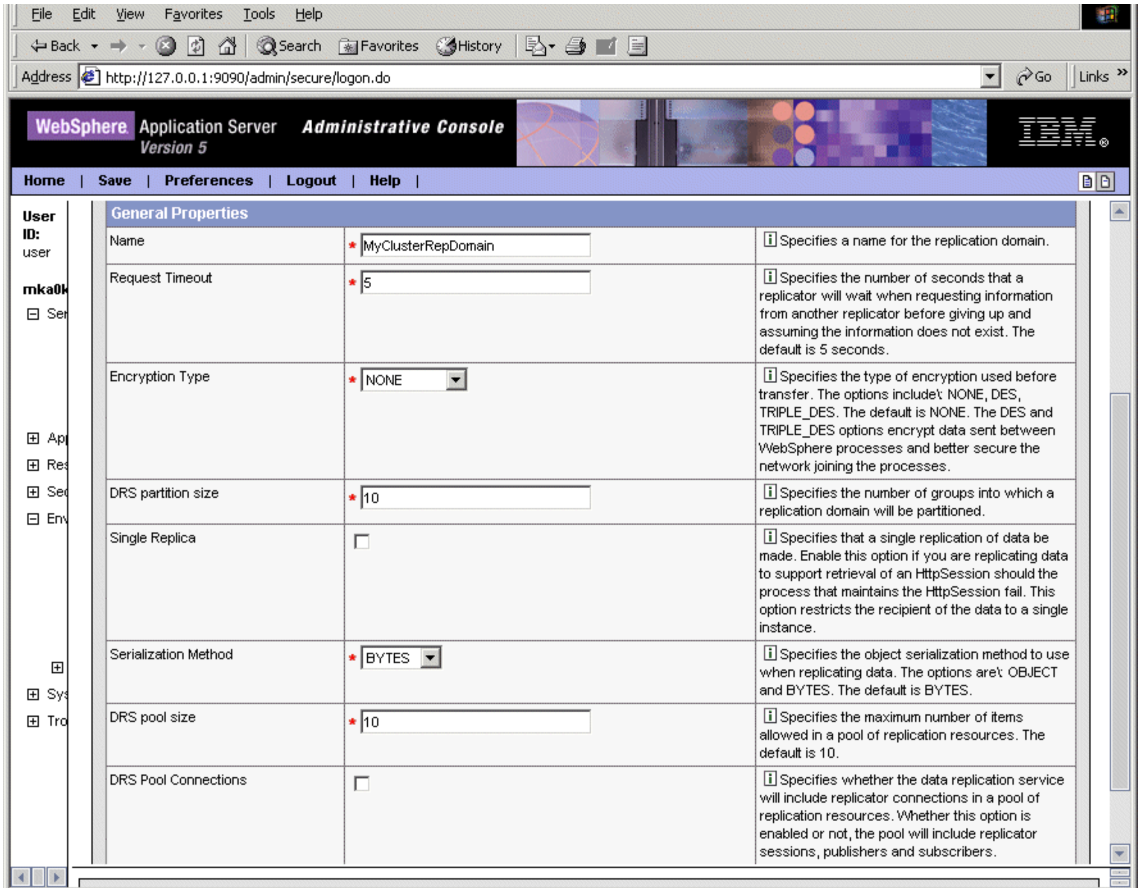
1. Create Replication Domain.

   Replicator domain defines the set of replicator processes that communicate with each other.

   a. Click **Environment -> Internal Replication Domains -> New**

   b. Enter each property. Enter at least a name for the replication domain. In this example, `MyClusterRepDomain` is used as name and defaults are used as other properties.

   c. Click **OK**


File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: user
mka0k

General Properties

Name	* MyClusterRepDomain	Specifies a name for the replication domain.
Request Timeout	* 5	Specifies the number of seconds that a replicator will wait when requesting information from another replicator before giving up and assuming the information does not exist. The default is 5 seconds.
Encryption Type	* NONE	Specifies the type of encryption used before transfer. The options include: NONE, DES, TRIPLE_DES. The default is NONE. The DES and TRIPLE_DES options encrypt data sent between WebSphere processes and better secure the network joining the processes.
DRS partition size	* 10	Specifies the number of groups into which a replication domain will be partitioned.
Single Replica	☐	Specifies that a single replication of data be made. Enable this option if you are replicating data to support retrieval of an HttpSession should the process that maintains the HttpSession fail. This option restricts the recipient of the data to a single instance.
Serialization Method	* BYTES	Specifies the object serialization method to use when replicating data. The options are: OBJECT and BYTES. The default is BYTES.
DRS pool size	* 10	Specifies the maximum number of items allowed in a pool of replication resources. The default is 10.
DRS Pool Connections	☐	Specifies whether the data replication service will include replicator connections in a pool of replication resources. Whether this option is enabled or not, the pool will include replicator sessions, publishers and subscribers.

Figure 17-17 Create Replication Domain

d. Click **Save**, **Save**

2. Add Replicator Entries.

   Replicator entry, or simply replicator, runs inside an application server process and defines its basic settings, such as a communication port among replicators and a communication port with clients.

   Repeat the following steps for each replicator entry.

   a. Click **Environment -> Internal Replication Domains -> <replicator_domain_name>**

   b. Click **Replicator Entries -> New**

      The following values are used for the first replication entry in this example.

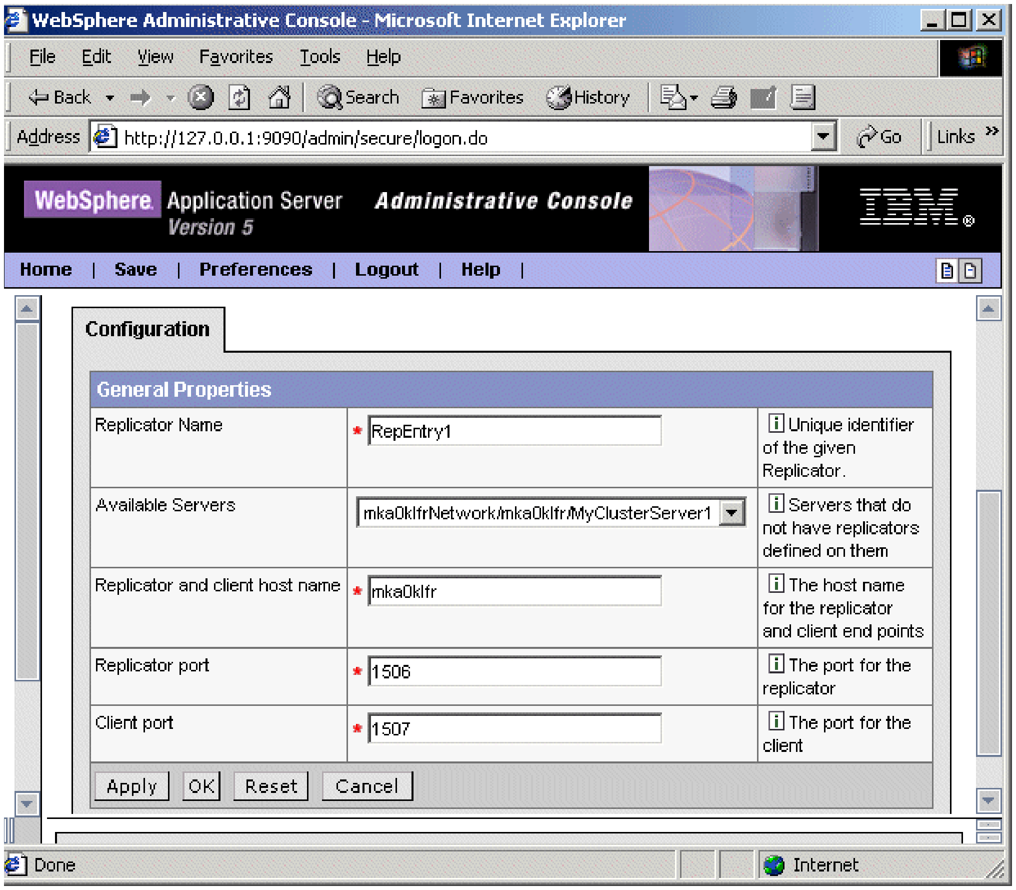
      • Name: `RepEntry1`

- Server: `m10df51fNetwork/m10df51f/MyClusterServer1`
- Replicator and client host name: `mka0klfr`
- Replicator port: `1506`
- Client port: `1507`

The following values are used for the second replication entry in this example.

- Name: `RepEntry2`
- Server: `m10df51fNetwork/m10df51f/MyClusterServer2`
- Replicator and client host name: `mka0klfr`
- Replicator port: `1508`
- Client port: `1509`

c. Click **OK**


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Configuration
Done
Internet

General Properties

Replicator Name	* RepEntry1	Unique identifier of the given Replicator.
Available Servers	mka0klfrNetwork/mka0klfr/MyClusterServer1	Servers that do not have replicators defined on them
Replicator and client host name	* mka0klfr	The host name for the replicator and client end points
Replicator port	* 1506	The port for the replicator
Client port	* 1507	The port for the client

Apply OK Reset Cancel

Figure 17-18 Add Replicator Entries

d. Click **Save**, **Save**

3. Configure cluster members.

   Repeat the following steps for each application server.

   a. Click **Servers -> Application Servers**

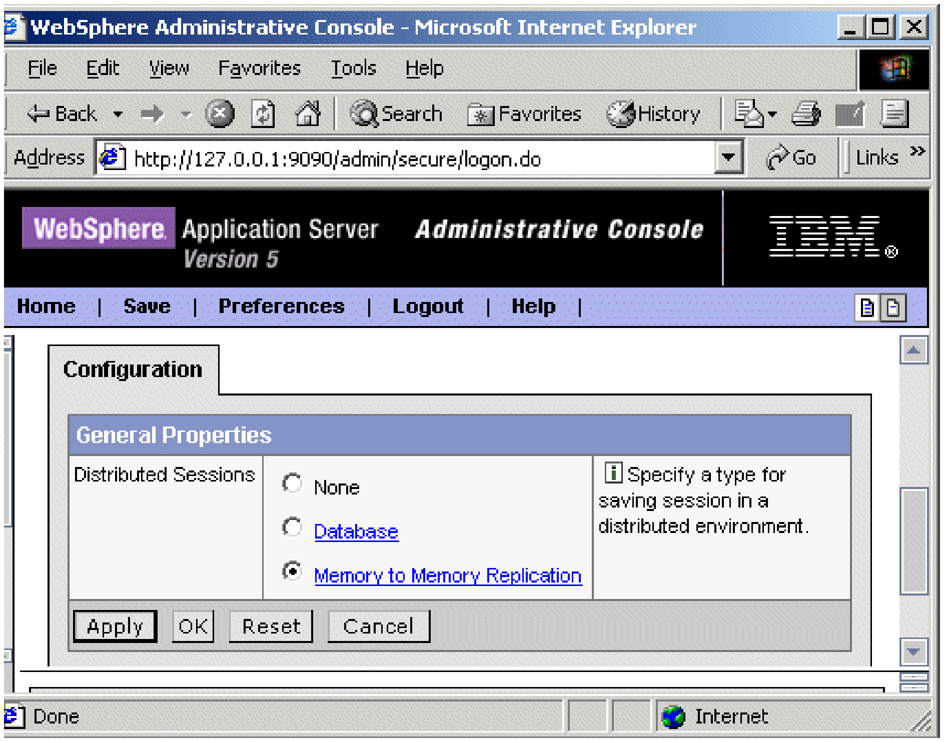
   b. Click **<application server>**. In this example, `MyClusterServer1` and `MyClusterServer2` are selected as application servers respectively.

   c. Click **Web Container**

   d. Click **Session Management**

   e. Click **Distributed Environment Settings**

   f. Select **Memory to Memory Replication** and click **Memory to Memory Replication**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
Configuration
General Properties
Apply OK Reset Cancel
Done
Internet

Distributed Sessions: ( ) None ( ) Database [✓ Memory to Memory Replication]

Specify a type for saving session in a distributed environment.

Figure 17-19 Distributed Environment Setting

g. Select **Select replicator from the following domain.** Choose a replicator domain and replicators from the lists. In this example, the following values are used:

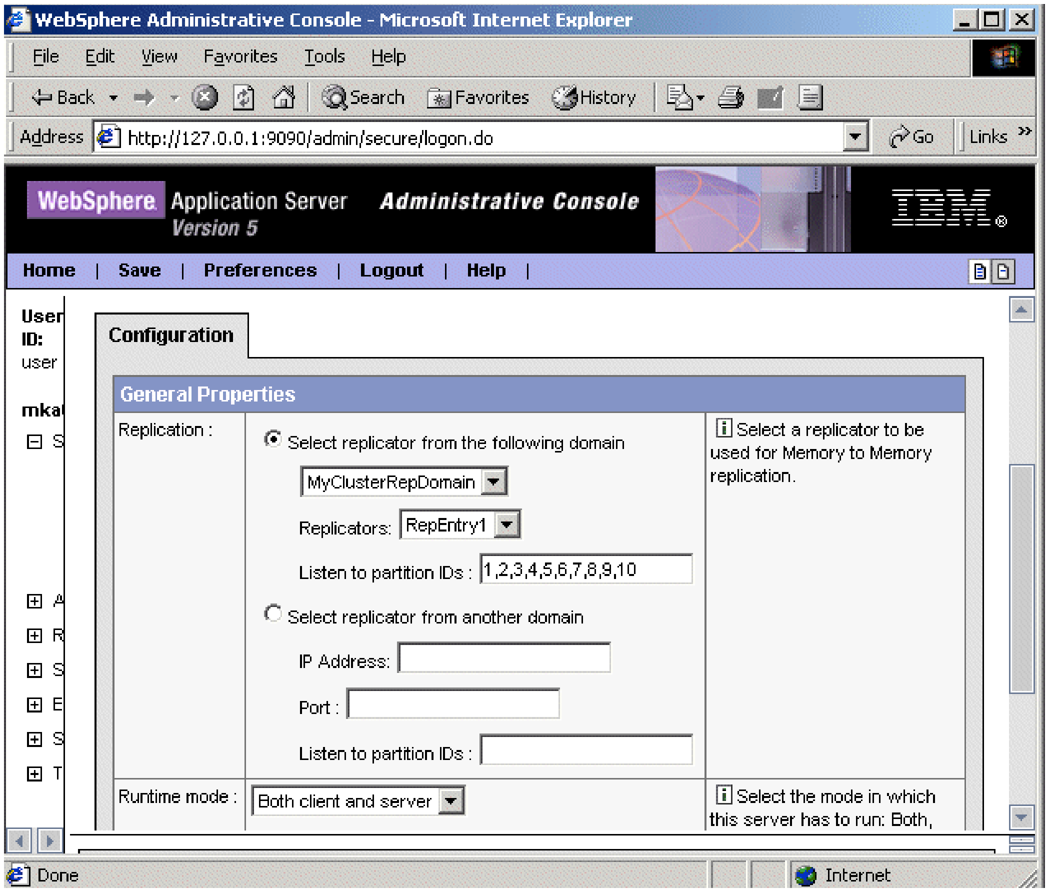
First application server:

- Replicator Domain: `MyClusterRepDomain`
- Replicator: `RepEntry1`

Second application server:

- Replicator Domain: `MyClusterRepDomain`
- Replicator: `RepEntry2`

h. Click **OK**


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: user
mka
Configuration
General Properties
Done
Internet

Replication : ● Select replicator from the following domain

MyClusterRepDomain

Replicators: RepEntry1

Listen to partition IDs : 1,2,3,4,5,6,7,8,9,10

○ Select replicator from another domain

IP Address:

Port :

Listen to partition IDs :

Select a replicator to be used for Memory to Memory replication.

Runtime mode : Both client and server

Select the mode in which this server has to run: Both,

Figure 17-20 Internal Messaging

i. Click **Save**, **Save**

4. Restart the cluster.

Replication domain settings

The followings are configuration properties for replication domain, which can be configured in the window shown in Figure 17-17 on page 587. DRS stands for Data Replication Service, which is service used in WebSphere Internal Messaging.

- Name

  Specifies a name of the replication domain. This name is unique throughout the cell.

- Request timeout

Specifies the number of seconds that a replicator waits when it requests information from another replicator.

- Encryption Type

  Specifies the encryption type used for transferring information between replicators. NONE, DES, or TRIPLE_DES can be selected. Encrypted transmission achieves better security. But it might impact the performance. If DES or TRIPLE_DES is specified, a key for data transmission is generated. It is recommend to generate a key by clicking **RegenerateKey** button periodically to enhance security.

- DRS Partition Size

  Specifies the number of partitions in a replication domain. See “Partioning” on page 594 for explanation of partition.

- Single Replica

  Specifies that a single replication of data be made. This setting provides filtering over partitioning. With this option, only one other replicatior entry receives a session.

- Serialization Method

  Specifies the method of object serialization for replicating data. Session can be either stored as bytes or an Object in persistent store. In order to store it as an Object, all class types of the objects in the session should be defined in the classpath of all application servers which use and/or store the session. Storing a session as bytes is convenient because there is no need to be careful about the classpath. This option is useful when some application servers are used for just storing sessions and do not perform servlet processing. Generally, storing a session as an Object is faster than storing a session as bytes. Of course, all Java objects stored in a session should be declared as Serializable.

- DRS pool size

  Specifies the maximum number of resources in the replication pool. Pooling resources can enhance the performance of replication.

- DRS pool connections

  Specifies whether replication connections are included in a replication resource pool or not.

See `Infocenter` for more information.

Replicator entry settings

The followings are configuration properties for replicator entry, which can be configured in the window shown in Figure 17-18 on page 589.

- Replicator name

  Specifies a name of the replicator entry. This name is unique throughout the cell.
- Available servers

  Specifies an application server for a replicator. A replicator runs as threads inside an application server. Only a single replicator can be defined for an application server.
- Replicator and client host name

  Specifies the IP address or host name on which a replicator runs.
- Replicator port

  Specifies a port number which enables communication among replicators.
- Client port

  Specifies a port number which enables communication between an application server process and a replicator.

See `Infocenter` for more information.

Internal messaging settings

The followings are configuration properties for memory to memory replication under session management menu. It can be configured in the window shown in Figure 17-20 on page 591.

- Replication

  Specifies a replicator for using WebSphere internal messaging. A replicator can be selected either from listed domains or from another domains.

  If **Select replicator from the following domain** is selected, a domain and a replicator should be selected from the lists. The partition IDs is also needed to be specified. See “Partioning” on page 594 about partitioning.

  If **Select replicator from another domain** is selected, IP address and port number of the replicator in another domain should be specified. The partition IDs is also needed to be specified. See “Partioning” on page 594 about partitioning.
- Runtime mode

  Specifies runtime mode from the list, which includes **Both**, **Client Only**, and **Server Only**. Data is sent in client mode, and is received in server mode.

See `Infocenter` for more information.

Author Comment: There is no help available for this window now. After it is available, content should be checked.

Partioning

The partitions can be set up to allow application servers to listen only on configured channels. This reduces overhead of data transfer among application servers. The partitions are available with both topologies, client-server or peer-to-peer. The nimbler of partiitions can be configurable.

Partitioning the replication domain is most often used to support retrieval of a session when the process maintaining the session fails. It is not supported for sharing cached data maintained by Web container dynamic caching.

To configure a partition, the number of partitions should be decided first and the partition assignment to each replicator entry should be done next. The number of partitions is configured by the **DRS partition size** property in the window shown in Figure 17-17 on page 587. The partition assignment to each replicator entry is configured by the **Listen to partition IDs** property in the window shown in Figure 17-20 on page 591.

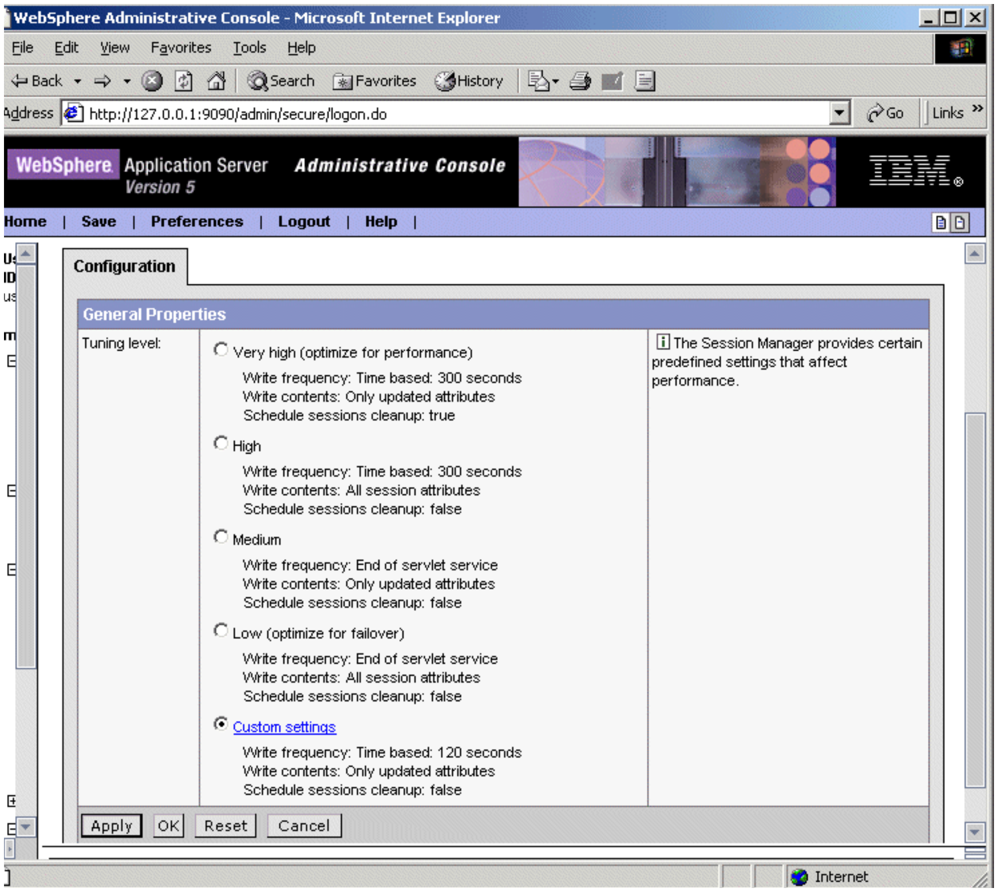
17.8.3 Manual tuning of persistent sessions

To manually tune persistent session management:

1. Click **Servers -> Application Servers**
2. Click **<application server>**. An application server which will be configured should be clicked here.
3. Click **Web Container**
4. Click **Session Management**
5. Click **Distributed Environment Settings**
6. Click **Custom Tuning Parameters**

In the window, shown in Figure 17-21 on page 595, which allows the administrator to manually tune:

- How often the session data is written.
- How much data is written.
- When the invalid sessions are cleaned up.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Configuration
General Properties

Tuning level:

( ) Very high (optimize for performance)
Write frequency: Time based: 300 seconds
Write contents: Only updated attributes
Schedule sessions cleanup: true

( ) High
Write frequency: Time based: 300 seconds
Write contents: All session attributes
Schedule sessions cleanup: false

( ) Medium
Write frequency: End of servlet service
Write contents: Only updated attributes
Schedule sessions cleanup: false

( ) Low (optimize for failover)
Write frequency: End of servlet service
Write contents: All session attributes
Schedule sessions cleanup: false

[✓ Custom settings]
Write frequency: Time based: 120 seconds
Write contents: Only updated attributes
Schedule sessions cleanup: false

The Session Manager provides certain predefined settings that affect performance.

Figure 17-21 Tuning

7. Click **Custom Tuning Parameters**

   Custom tuning window, shown in Figure 17-22 on page 597, appears.

The persistence tuning properties are:

- Write frequency

  This group of options defines how often the HTTP session data is written to the persistent session store.

  – End of servlet service

    If the session data has changed, it will be written to the persistent store after the servlet finishes processing an HTTP request. For more information, see 17.8.4, “Update persistent store at end of servlet service” on page 597.

  - Manual update

    The session data will be written to the persistent store when the sync() method is called on the IBMSession object. For more information, see 17.8.5, “Manual update of the persistent store” on page 598.

  - Time based

    If you select this option, session data will be written to the persistent store based on the specified write interval value. For more information, see 17.8.6, “Time based writes to the session database” on page 599.

  Note: WebSphere V4.0 does not validate session creations for duplicates and it depends on random session id generation. WebSphere V5.0 avoids this issue by writing a new session into a persistent store just after the creation of a new session to find out any duplicates.

- Write contents

  These options control what is written. Please refer to 17.8.10, “What is written to the persistent session database” on page 606 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 persistent store.

  - All session attributes

    All attribute are written to the persistent store.

- Schedule sessions cleanup

  WebSphere allows the administrator to defer the clearing of invalidated sessions (sessions that are no longer in use and timed out) from the persistent store 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)

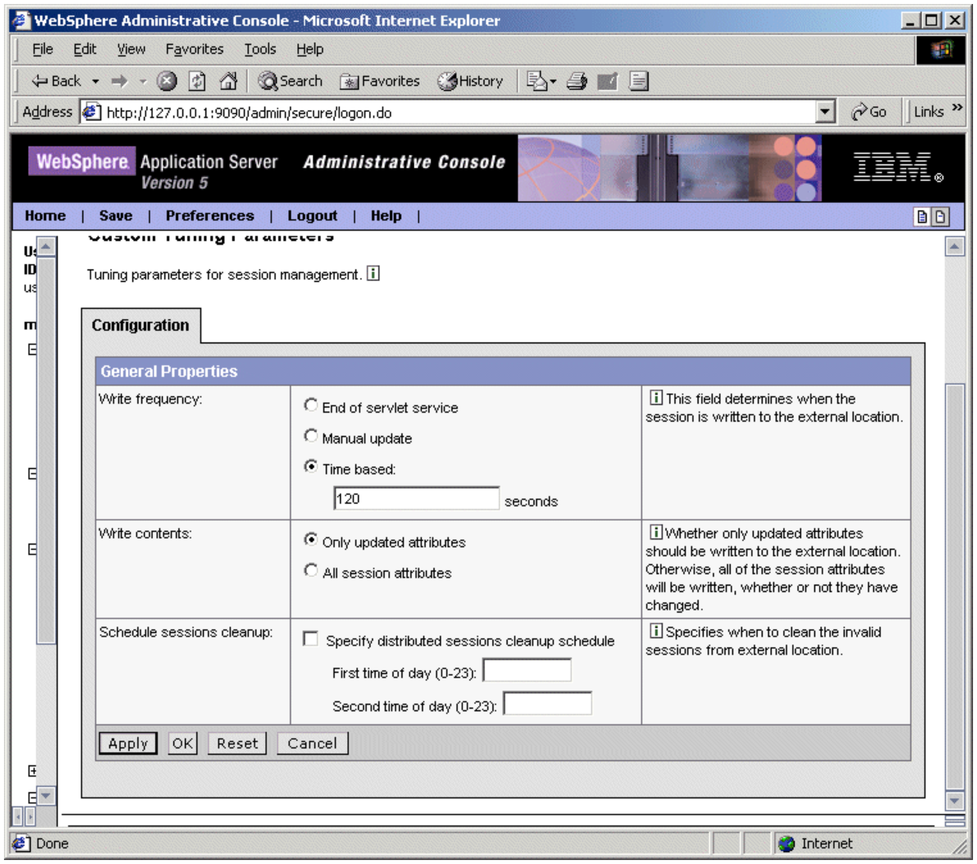
    The first hour during which the invalidated persistent sessions will be cleared from the persistent store. This value must be a positive integer between 0 and 23.

  - Second time of day (0-23)

    The second hour during which the invalidated persistent sessions will be cleared from the persistent store. This value must be a positive integer between 0 and 23.

The property, **Specify distributed sessions cleanup schedule**, is required to be selected to enable this option.

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.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://127.0.0.1:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
Configuration
Apply OK Reset Cancel
Done
Internet

Tuning parameters for session management.

General Properties

Write frequency:	○ End of servlet service ○ Manual update ● Time based: 120 seconds	This field determines when the session is written to the external location.
Write contents:	● Only updated attributes ○ All session attributes	Whether only updated attributes should be written to the external location. Otherwise, all of the session attributes will be written, whether or not they have changed.
Schedule sessions cleanup:	☐ Specify distributed sessions cleanup schedule First time of day (0-23): Second time of day (0-23):	Specifies when to clean the invalid sessions from external location.

Figure 17-22 Custom Tuning

17.8.4 Update persistent store 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 persistent store at the completion of the HttpServlet.service() method call. Exactly what is written depends on another setting. See 17.8.10, “What is written to the persistent session database” on page 606 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 persistent store update of the last access time. If you select the **Manual update** option, discussed in 17.8.5, “Manual update of the persistent store” on page 598, the update of the last access time in persistent store occurs on sync() call or at later time.

See “Reduce persistent store I/O” on page 617 for options available to change this database update behavior.

17.8.5 Manual update of the persistent store

WebSphere has a manual session writing mode. This 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 2.3 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 store. The session data that is written out to the persistent store 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 persistent store asynchronously, at later time.

Example 17-8 shows how the IBMSession class can be used to manually update the persistent store.

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 17-8 Using IBMSession for manual update of the persistent store

```
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 persistent store
   session.sync();
}
```

Note: In WebSphere V4.0, manual update can be performed when the manual update option is selected. In WebSphere V5.0, invoking sync() method always saves the session to the persistent store.

17.8.6 Time based writes to the session database

Time based write will write session data to the persistent store 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 persistent store can now be relaxed somewhat in these environments (as well as non-clustered environments) since the persistent store 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 persistent store writes.

Using this mode, WebSphere can significantly reduce the I/O to the persistent store 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 persistent store. If write at “End of servlet service” mode is enabled, then WebSphere could be writing to the persistent store every time it process an HTTP request. This can be a significant overhead. By using the time based mode, these updates to the persistent store 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 overhead of session transfer. See “Reduce persistent store I/O” on page 617 for more information.

Note: Time based writes requires session affinity for session data integrity.

Important: In WebSphere V5.0 IBM Extension, there is an option to have session scope per Enterprise application. See “Session scope” on page 556. There is also an option to enable serialized access to the session: see “Advanced settings for session management” on page 567.

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 persistent store 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 persistent store 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 persistent store writes since the receipt of a new session request may result in writing out the oldest cached session to the persistent store. 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 persistent store.
- A newly created session will always get written to the persistent store at the end of the service method.

17.8.7 Persistent sessions and non-serializable J2EE objects

In order for the WebSphere session manager to persist a session to the persistent store, 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.

17.8.8 Using larger DB2 page sizes

Important: This section only apply for the use of database persistence.

WebSphere 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 Database window. As seen in Figure 17-13 on page 579, page size is referred to as **DB2 row size** property.

Restart WebSphere. On startup, the session manager creates a new SESSIONS table based on the page size and table space name specified.

17.8.9 Support for single or multi-row schemas

Important: This section only apply for the use of database persistence.

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 618.

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 select the **Use Multi row schema** on the Database setting of the Session Manager window, shown in Figure 17-13 on page 579.
- 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 cluster.

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 17-2 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.
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.

17.8.10 What is written to the persistent session database

Important: This section only apply for the use of database persistence.

As we saw in Figure 17-22 on page 597, WebSphere supports two modes for writing persistent session contents to the database.

- **Only updated attributes**. It writes only the HttpSession properties that have been updated via setAttribute() and removeAttribute().
- **All session attributes**. 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 17-9 and Example 17-10, the initial session creation contains a setAttribute but subsequent requests for that session do not need to use setAttribute.

Example 17-9 Initial servlet

```
HttpSession sess = request.getSession(true);
myClass myObject = new myClass();
myObject.someInt = 1;
sess.setAttribute("myObject", myObject);   // Bind object to the session
```

Example 17-10 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 17-11 and Example 17-12 show the case where setAttribute is still required even though the write "All session attributes" option is enabled.

Example 17-11 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 17-12 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 17-3 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 17-3 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]
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

Row type	Write contents	Write frequency	Action for setAttribute	Action for remove-Attribute
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.

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.

17.9 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 Management window as seen in Figure 17-8 on page 567.
- For persistent sessions only, the administrator can specify times when the scan will be run. The session persistent store cleanup schedule for invalidated sessions is set in the Custom Tuning window, as seen in Figure 17-22 on page 597. This feature has the following benefits when used with persistent session:
  – Persistent store scans can be scheduled during periods that normally have low demand. This avoids slowing down online applications due to contention in the persistent store.
  – 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. Listeners, which are described in “Session listeners” on page 610, processing is potentially delayed by this this configuration. It is not recommended if listeners are used.

17.9.1 Session listeners

Some Listener classes are defined in the Servlet 2.3 specification in order to listen for state changes of a session and its attributes: for example, to create, destroy a session, and to add, delete, and modify its attributes. This allows greater control over interactions with sessions. This leads programmers to be able to monitor creation, deletion, and modification of sessions. Programmers can perform some initialization task when a session is created or some clean up task, such as deleting temporary file, when a session is removed. It is also possible to perform some specific tasks for the attribute when an attribute is added/deleted or modified.

The followings are the Listener interfaces to monitor the events associated with the HttpSession object:

- javax.servlet.http.HttpSessionListener

  This interface monitors creation and deletion, including session timeout, of a session.

- javax.servlet.http.HttpSessionAttributesListener

  This interface monitors changes of session attributes, such as add, delete, and replace.

- javax.servlet.http.HttpSessionActivationListener

  This interface monitors activation and passivation of sessions. This interface is useful to monitor the session exists whether on memory or not when persistent session is used.

The following table shown in Table 17-4 on page 611 is a summary of the interfaces and methods.

Table 17-4 Listener interfcaces and their methods

Target	Event	Interface	Method
session	create	HttpSessionListener	sessionCreated()
	destroy	HttpSessionListener	sessionDestroyed()
	activate	HttpSessionActivationListener	sessionDidActivate()
	passivate	HttpSessionActivationListener	sessionWillPassivate()
attribute	add	HttpSessionAttributesListener	attributeAdded()
	remove	HttpSessionAttributesListener	attributeRemoved()
	replace	HttpSessionAttributesListener	attributeReplaced()

See API documentation in http://java.sun.com/j2ee/sdk_1.3/techdocs/api/ for detail.

17.10 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 *WebSphere Security redbook* for more information on WebSphere security.

Author Comment: Name of the security redbook should be inserted into the previous sentence.

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 with SWAM. See *WebSphere Security Redbook* about SWAM.

Table 17-5 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 17-5 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.
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].

Note: com.ibm.websphere.servlet.session.UnauthorizedSessionRequestException is thrown to the servlet or JSP.

See “Advanced settings for session management” on page 567 for the steps to enable session security.

17.11 Session performance considerations

Author Comment: Currently, no performance tuning information is available for the use of memory to memory replication.
It is generally true that memory to memory replication is faster than persistent database. Since there is no supporting information about this, it is not written in this section. Performance tuning information might be available after the GA.

This section includes guidance for developing and administering scalable, high-performance Web applications using WebSphere Application Server session support.

17.11.1 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 persistent store. 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 17.8.6, “Time based writes to the session database” on page 599.

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 17.8.8, “Using larger DB2 page sizes” on page 603 on how WebSphere can provide faster persistence of larger session objects when using DB2.

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 instances

WebSphere also gives the administrator the option of creating additional application server instances. Creating additional instances spread 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 cluster members, the WebSphere HTTP plug-in provides affinity routing among cluster member 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.

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 store contains a large number of entries, frequent execution of the timeout scanner reduces overall performance. In cases where the persistent store contains many session entries, avoid setting the session timeout so low it triggers frequent, expensive scans of the persistent store 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.

17.11.2 Reduce persistent store 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 persistent store. If you are using persistent database, see 17.8.10, “What is written to the persistent session database” on page 606 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 persistent store 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 cluster member 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 17-13. 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 17-13 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 persistent store 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 distributed sessions cleanup schedule** 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.

17.11.3 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 17-6 shows a simplified representation of a multi-row database table.

Table 17-6 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”	

Session ID	Web application	Property	Small value	Large value
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. For information on session updates using single and multi-row session support see 17.8.9, “Support for single or multi-row schemas” on page 604. Also see 17.8.10, “What is written to the persistent session database” on page 606.

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.

17.11.4 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.

17.11.5 Session database tuning

While the session manager implementation in WebSphere 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.

17.12 Handle exception

Author Comment: In the class document, it is written that there is an option to propagate exceptions back to servlet/JSP or not, like when getting exceptions from database. Actually, there is no option now. If this option will not be available until GA, this section should be removed.

18

Configuring WebSphere resources

Resource providers are a class of objects that provide resources needed by running Java applications, and J2EE applications in particular. For example, if your application requires database access through a datasource, you can install a JDBC data source provider, and then configure a datasource to be used by your application.

This chapter discusses the following application server resource providers:

- JMS providers for obtaining JMS connection factories and destinations.
- 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.
- Environment providers for obtaining environment resource factories.

Note: In this chapter configuration of the different resources will be done via the administrative console. You can also configure these resources using the scripting interface *wsadmin*.

►

18.1 JMS and JMS providers

The Java Message Service (JMS) is a standard vendor-neutral API that is part of the J2EE platform and can be used to access enterprise messaging systems. Enterprise messaging systems facilitate the exchange of messages among software applications over a network.

JMS provides vendor-independent access to enterprise messaging systems. Many enterprise messaging products currently support JMS, including IBM's WebSphere MQ. Software applications that use the JMS API for sending or receiving messages are portable across brands of JMS vendors.

One of the principal advantages of JMS messaging is that it's asynchronous. In other words, a JMS client can send a message without having to wait for a reply. When a JMS client sends a message, it sends the message to a router, which is responsible for forwarding it to other clients. Clients sending messages are decoupled from the clients receiving them. Senders are not dependent on the availability of receivers.

Java applications that use JMS are called *JMS clients*, and the messaging system that handles routing and delivery of messages is called the *JMS provider*.

18.2 JMS Resources

JMS resources are the key to JMS portability. JMS resources are created by a JMS administrator and are specific to the underlying JMS-vendor implementation. These JMS resources however, implement standard JMS interfaces and are retrieved by JMS applications through JNDI. JMS developers must only know the JNDI name of the JMS resources, and do not have to write any vendor-specific code to use the administered objects.

In order to send a JMS message from an application we need the following JMS resources:

- A connection to the JMS provider. This is made possible by a JMS connection factory object.The JMS connection factory object is used by the application to connect to the JMS provider or message router.
- A destination address for the message. The destination address of the message is identified by a Destination object. This is used by the application to send and receive messages.

  The destination object represents a network-independent destination to which the message will be addressed. In JMS, messages are sent to destinations, instead of directly to other applications.

There are two types of destination objects:

- **Topic**

  A topic is analogous to an email list or newsgroup. Any application with the proper credentials can receive messages from and send messages to a topic. When a JMS client receives messages from a topic, the client is said to subscribe to that topic.

- **Queue**

  Queues are intended for point-to-point messaging. A queue may have multiple receivers, but only one receiver may receive each message.

The WebSphere administrative console can be used to configure and administer these resources for the integral JMS provider and the WebSphere MQ JMS provider.

Note: The functionality of the administrative console is twofold for the integral and WebSphere MQ JMS Provider. With the administrative console you do two things:

1. Configure JMS administrative objects in the WebSphere name space.
2. Configure and create messaging resources (queues) in the messaging system.

Author Comment: For the internal JMS Provider, when you create a queue, I cannot see a queue created under the MQSeries\Qmgrs\QueueManagerName\Queues directory? Is the purpose of the JMS integral provider to provide messaging only internally within a cell?

18.3 Installing a JMS provider

EJB 2.0, introduces message-driven beans for processing asynchronous messages delivered by JMS.

For an Application Server to support message-driven beans, a JMS provider that conforms to the JMS specification version 1.0.2 and supports the Application Server Facility (ASF) function defined within that specification needs to be installed.

In J2EE 1.3, JMS becomes an integral part of the J2EE platform. WebSphere Application Server Version 5 satisfies this requirement by including a JMS provider as part of the installation.

For a detailed description of WebSphere support for JMS and JMS providers, see Chapter 7, “Asynchronous messaging” on page 147.

This section describes the installation of the JMS providers that can be used with WebSphere Application Server:

1. WebSphere integral JMS provider. This is installed as part of WebSphere Application Server.
2. WebSphere MQ JMS provider.

   Note: As noted in Section 18.2, “JMS Resources” on page 625, if you install the integral JMS Provider as the messaging system, you can use the administrative console for both, creating and administering resource objects into the WebSphere namespace and creating those resources (queues) for the actual JMS Provider.

3. Generic JMS provider, which must conform to the JMS specification and support the ASF function.

   If you want to use a JMS provider other than the integral WebSphere JMS provider or an WebSphere MQ JMS provider, you should complete the following steps:

   a. Install and configure the JMS provider.
   b. Create and configure its resources by using the tools and information provided with the JMS provider.
   c. Define a generic JMS provider in WebSphere.
   d. Bind resources into the application server’s name space.

18.3.1 Installing the WebSphere integral JMS provider

The WebSphere integral JMS provider gets installed as part of the installation of WebSphere Application Server on a node.

Note: If you do a **custom** installation, make sure that **Embedded Messaging**, **Server and Client**, is selected among the features you would like to install.

The implementation of the integral JMS provider is based on the integration of IBM’s WebSphere WebSphere MQ product, the WebSphere MQ Event Broker ("Greyhound") product and the JMS Client ("MA88") support pack into WebSphere Application Server. The versions of these products that are integrated with WebSphere Application Server are reduced foot print and reduced function versions of the independently shipped MQ products.

Check the following to verify that Embedded Messaging has installed correctly on a node:

1. Check <WAS_ROOT_INSTALL>/mq_install.log

   This log should have a number of sections, including:

   – Prerequisites checking

   – WebSphere MQ install log

   – JMS install log

   – Publish/Subscribe install log

   All of the sections should end with something that looks like ERROR_SUCCESS (0).

2. Check the <WAS_ROOT_INSTALL>/createMQ.log file on the node.

   The install process should have created the Base Queue Manager and a number of MQ default queues. The queue manager will have a name of the form:

   ```
   WAS_<CellName>_<NodeName>
   ```

   For example, for a Base Application Server installation the Queue Manager will be located at:

   ```
   <MQ_ROOT>/Qmgrs/WAS_<cellname>_<nodename>
   ```

   When the node is added into a cell, the Queue Manager gets renamed to WAS_NetworkDeploymentCell_<nodename>, as shown in Figure 18-1:

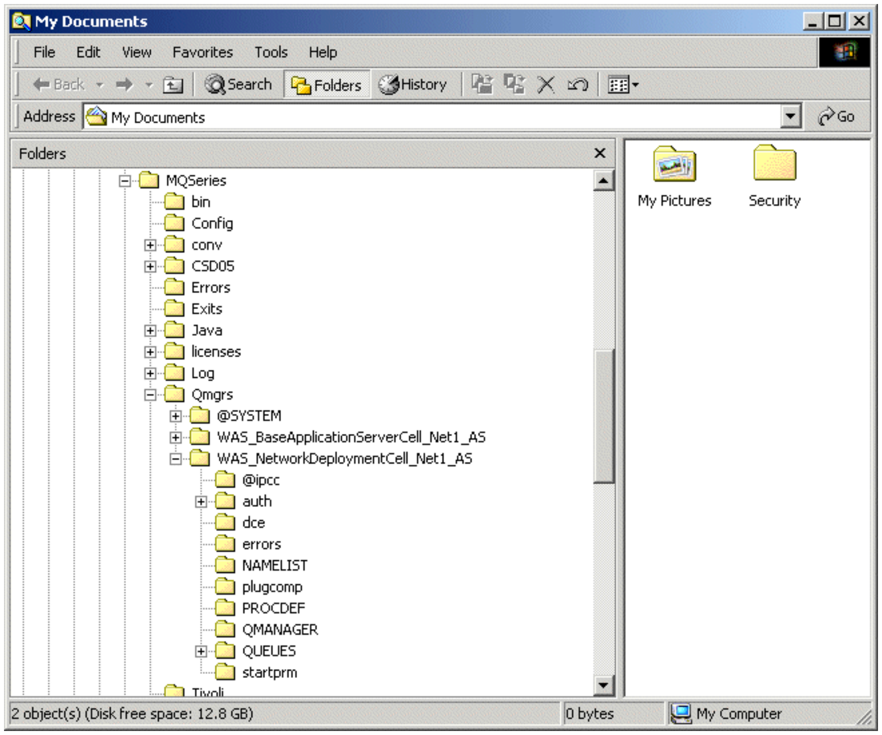


Figure 18-1 The queue manager gets created when you install the integral JMS Server Provider

Note:

- You can have zero or one Queue Managers per node.
- When a node is added to a cell, the base installation Queue manager is not deleted.

3. On Windows, you can also see the three components that make up Embedded Messaging by looking in Add/Remove Programs in the Control Panel:
   a. IBM WebSphere MQ classes for Java and WebSphere MQ classes for Java Message Service (the JMS client, ie, MA88).
   b. WebSphere MQ (point-to-point).
   c. WebSphere Embedded Messaging Publish And Subscribe Edition (publish/subscribe broker).

Creating the WebSphere integral JMS Server

When installing the integral JMS provider (WebSphere MQ, plus JMS client, plus Publish/Subscribe Broker), a JMS Server managed process, is automatically created for the node in which the JMS provider is installed.

On a base WebSphere installation, the JMS Server runs in the same JVM as the application server, ie. it is part of the application server process. However, when the node gets added into a cell, a WebSphere Application Server managed server *jmsserver* that runs in its own JVM, is created. This JVM is managed by the node agent.

The JMS Server process on a node provides the JMS functions of the JMS Server on that node. It is responsible for managing the Queue Manager (stop and start) and hosts the Broker.

Note: When using the integral JMS Provider there is no WebSphere Administration Console support for Queue Manager or channel administration.

Messaging queues for the integral JMS Provider (as opposed to administrative queue objects), get also *created* when we define them to a Node's JMS Server process.

Note: There is no WebSphere Administration Console support for administering the messaging queues, eg. sending test messages or clearing messages from a queue.

Make sure that the JMS server process has been started in order for the embedded messaging service to function properly. Look in the JMS Server's *SystemOut.log* file located on the node at:

```
<WAS_ROOT>/logs/jmsserver/SystemOut.log
```

The log will contain entries similar to those in Example 18-1:

Example 18-1 jmsserver's SystemOut.log file in the node

```
[8/14/02 17:45:36:453 EDT] 50d39cf0 JMSProvider   A MSGS0050I: Starting the
Queue Manager
[8/14/02 17:45:40:688 EDT] 50d39cf0 JMSProvider   A MSGS0051I: Queue Manager
open for business
[8/14/02 17:45:40:719 EDT] 50d39cf0 JMSProvider   A MSGS0052I: Starting the
Broker
[8/14/02 17:45:46:500 EDT] 50d39cf0 JMSProvider   A MSGS0053I: Broker open for
business
```

Notice in Figure 18-2, that when the JMS Server process is started, a number of additional processes (**amq***) are started. These are all part of the WebSphere MQ Queue Manager. In addition there are two **runmq*** processes. These processes are created when the JMS Server process starts the Queue Manager for the JMS provider.

For more information on these processes refer to the *MQSeries System Administration Guide*, SC33-1873-02.

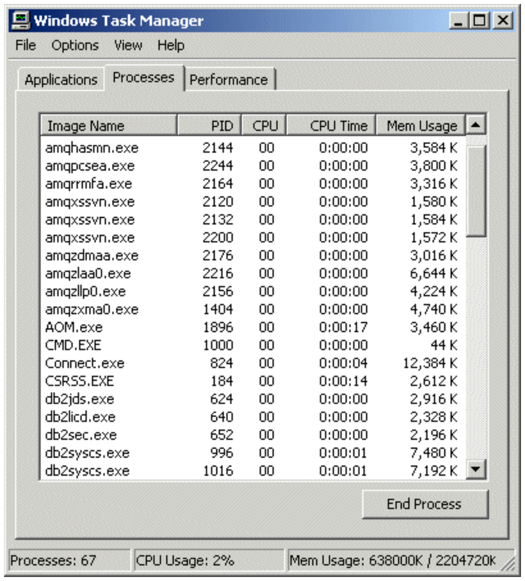


Image Name	PID	CPU	CPU Time	Mem Usage
amqhasmn.exe	2144	00	0:00:00	3,584 K
amqpcsea.exe	2244	00	0:00:00	3,800 K
amqrrmfa.exe	2164	00	0:00:00	3,316 K
amqxssvn.exe	2120	00	0:00:00	1,580 K
amqxssvn.exe	2132	00	0:00:00	1,584 K
amqxssvn.exe	2200	00	0:00:00	1,572 K
amqzdmaa.exe	2176	00	0:00:00	3,016 K
amqzlaa0.exe	2216	00	0:00:00	6,644 K
amqzllp0.exe	2156	00	0:00:00	4,224 K
amqzxma0.exe	1404	00	0:00:00	4,740 K
AOM.exe	1896	00	0:00:17	3,460 K
CMD.EXE	1000	00	0:00:00	44 K
Connect.exe	824	00	0:00:04	12,384 K
CSRSS.EXE	184	00	0:00:14	2,612 K
db2jds.exe	624	00	0:00:00	2,916 K
db2licd.exe	640	00	0:00:00	2,328 K
db2sec.exe	652	00	0:00:00	2,196 K
db2syscs.exe	996	00	0:00:01	7,480 K
db2syscs.exe	1016	00	0:00:01	7,192 K

Processes: 67   CPU Usage: 2%   Mem Usage: 638000K / 2204720K

Figure 18-2 WebSphere MQ processes started by the JMS Server process

Note: There can be at most one JMS server process on each node in the cell. Any of the application servers within the cell can access the resources (queues) associated with any of the JMS Server processes in the cell.

18.3.2 Installing WebSphere MQ as the JMS provider

In this section we describe how to install WebSphere MQ with support for the Java Message Service (JMS) for use with the WebSphere Application Server:

.To install WebSphere MQ with support for JMS, complete the following steps:

1. Install a recommended version of WebSphere MQ server, such as WebSphere MQ 5.2.1, as described in the installation instructions provided

with the WebSphere MQ package. Install also a recommended CSD on top of WebSphere MQ, CSD5 at present.

Do not install the WebSphere MQ classes for Java or JMS from the shipped CD.

Note: WebSphere Version 5 will support WebSphere MQ 5.3 + CSD also.

2. Download and install the WebSphere MQ 5.2 MA88 SupportPac (WebSphere MQ classes for Java and WebSphere MQ classes for Java Message Service) for your operating system

   `http://www-4.ibm.com/software/ts/mqseries/txppacs/ma88.html`

   For information about how to install the MA88 SupportPac, see the installation instructions for your operating system provided on the download Web page.

3. If you want to use WebSphere MQ Publish/Subscribe support, you need to provide a Publish/Subscribe broker. At present, you can do this by using either MQ Publish and Subscribe, WebSphere MQ Integrator, or the WebSphere MQ Publish/Subscribe MA0C SupportPac.

   For more information about WebSphere MQ Integrator, see the WebSphere MQ Integrator Web site at

   `http://www-4.ibm.com/software/ts/mqseries/platforms/#integrator`

   If you want to use the MA0C SupportPac, complete the following substeps:

   a. Download the WebSphere MQ 5.2 MA0C SupportPac (WebSphere MQ -Publish/Subscribe) for your operating system from the following url:

      `http://www-4.ibm.com/software/ts/mqseries/txppacs/ma0c.html`

   b. Install the MA0C SupportPac as described in the installation instructions for your operating system provided on the download Web page.

   Note: The use of MA0C as your publish/subscribe broker is now discouraged. However, it does work and is widely available. You should use WebSphere MQ Event Broker 2.1 (when available) or MQ Integrator instead.

4. Follow the WebSphere MQ 5.2 instructions for verifying your installation setup.

5. For UNIX platforms, add the following path to your LD_LIBRARY_PATH environment variable: <mq_install_path>/mqm/java/lib

7. For UNIX platforms, copy the following files from the <mq_install_path>/mqm/java/lib directory to the <WAS_ROOT>/lib/ext directory:

- fscontext.jar
- providerutil.jar
- com.ibm.mq.jar
- com.ibm.mqjms.jar

8. For UNIX platforms, copy the following files from the <mq_install_path>/mqm/java/lib directory to the <WAS_ROOT>/bin directory

- libmqjbnd02.so
- libMQXAi01.so

The full version of WebSphere MQ can be installed after installing the integral JMS provider. However, you will end up with only one copy of the message transport, ie MQ.

Existing JMS resource definitions for the integral WebSphere JMS provider will continue to work with WebSphere MQ as the JMS provider, so you do not need to redefine those JMS resources.

Make sure that WebSphere MQ and the JMS Client are at the required levels for embedded messaging. You won’t be able to install a version of WebSphere MQ that is lower than the one currently installed:

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

Note: Complete the following steps to install the full WebSphere MQ Provider after installing the integral JMS Provider:

1. Install a recommended version of WebSphere MQ server, such as WebSphere MQ 5.2.1. Run **`setup.exe`**.
2. Install a recommended CSD, such as CSD5 .
3. When asked if you want to remove or modify the version of WebSphere MQ currently installed, choose to remove the **Server**.
4. Select to keep existing queue managers.
5. Proceed with the installation of the full WebSphere MQ **Server** by running **`setup.exe`** again, after the old WebSphere MQ server has been removed.
6. Don't install the Client. Use the Client that was installed with the WebSphere integral JMS Server. This is because, beside the one that you get with your Base WebSphere installation, there is currently no MA88 available that meets the WebSphere prerequisites.

It is also recommended that you use WebSphere MQ Event Broker or MQSI as your publish/subscribe broker instead of MA0C.

18.3.3 Defining a generic JMS provider

To define a new JMS provider to WebSphere Application Server, for use instead of the integral WebSphere JMS provider or the WebSphere MQ JMS provider, you should have installed and configured the JMS provider and its resources by using the tools and information provided with the JMS provider.

To define a new JMS provider to WebSphere Application Server, use the administration web console to complete the following steps:

1. In the navigation tree, expand **`Resources->JMS-> Manage Generic JMS Providers`**.
2. Select the **`Scope`** and **`Apply`**.
3. Click **`New`** in the content page.
4. Define the JMS provider by specifying the appropriate values in the **`General Properties`** page.
5. Click **`OK`**.
6. To save your configuration, click **`Save`** on the task bar of the Administrative console window.

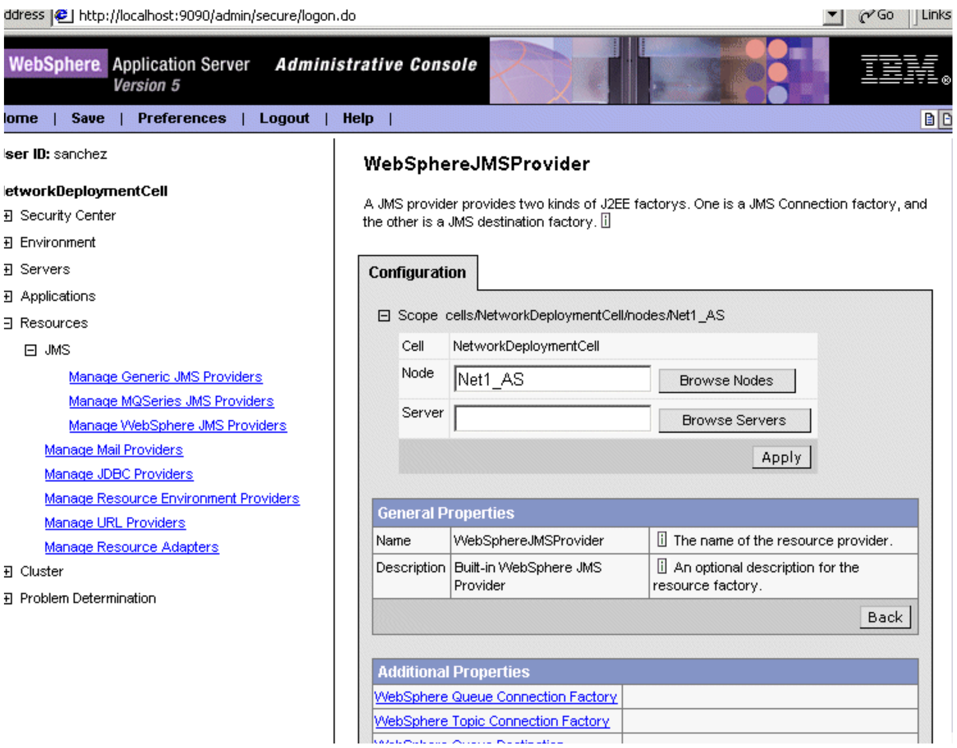
18.4 Managing JMS Providers

To select a JMS provider so you can view or change its configuration properties, use the administrative console.

Complete the following steps:

1. In the navigation tree, expand **Resources** -> **JMS**.
2. Select the type of JMS provider that you want to act on, that is **Generic**, **WebSphere MQ** or **WebSphere**.
3. Under **Scope** select the **Node** in which the JMS Provider is installed.
4. Click **Apply**.
5. Under **General Properties** you can view the properties for the JMS provider. Under **Additional Properties** you can view and change the resources for this JMS provider.

Figure 18-3 shows the console page for the WebSphere JMS providers installed in node Net1_AS.


ddress http://localhost:9090/admin/secure/logon.do
Go
Links
WebSphere Application Server Version 5
Administrative Console
IBM
lome | Save | Preferences | Logout | Help |
ser ID: sanchez
etworkDeploymentCell
Security Center
Environment
Servers
Applications
Resources
JMS
Manage Generic JMS Providers
Manage MQSeries JMS Providers
Manage WebSphere JMS Providers
Manage Mail Providers
Manage JDBC Providers
Manage Resource Environment Providers
Manage URL Providers
Manage Resource Adapters
Cluster
Problem Determination
Configuration
Browse Nodes
Browse Servers
Apply
Back
Additional Properties
WebSphere Queue Connection Factory
WebSphere Topic Connection Factory

WebSphereJMSProvider

A JMS provider provides two kinds of J2EE factorys. One is a JMS Connection factory, and the other is a JMS destination factory.

Scope cells/NetworkDeploymentCell/nodes/Net1_AS

Cell	NetworkDeploymentCell
Node	Net1_AS
Server	

General Properties

Name	WebSphereJMSProvider	The name of the resource provider.
Description	Built-in WebSphere JMS Provider	An optional description for the resource factory.

Figure 18-3 Console page for the WebSphere JMS provider installed in node Net1_AS

18.4.1 Managing the WebSphere JMS provider

To view the configuration properties of the integral WebSphere JMS provider that is installed with WebSphere Application Server complete the following steps:

1. In the navigation tree, expand **`Resources`** -> **`JMS`**.
2. Select **`Configure WebSphere JMS Providers`**.
3. Under **`Scope`** select the **`Node`** in which the integral JMS provider is installed.
4. Click **`Apply`**.
5. Under **`General Properties`** you can view the properties for the JMS provider:
   a. **Name**

      The name by which the JMS provider is known for administrative purposes. The default is *WebSphereJMSProvider*.

   b. **Description**

      A description of the JMS provider for administrative purposes. The default is *Built-in WebSphere JMS Provider*.

   Note: The values of these properties cannot be changed

6. Under **`Additional Properties`** you can view and change the resources for this JMS provider.

Note: When the integral JMS Provider is installed, there are a number of properties, WebSphere MQ properties, configured automatically for the integral WebSphere JMS Provider.

In general, the default values of WebSphere MQ properties are adequate. However, if you are running high messaging loads, you may need to change some WebSphere MQ properties; for example, properties for log file locations, file pages, and buffer pages. For more information about configuring WebSphere MQ properties, see the *MQSeries System Administration book*, SC33-1873.

Managing the WebSphere integral JMS Server process

You can use the WebSphere administrative console to display a list of all the JMS server processes in your cell, and to show and control their runtime status.You can also use the WebSphere administrative console to configure the general set of JMS server process properties, which add to the property values configured for the integral WebSphere JMS provider.

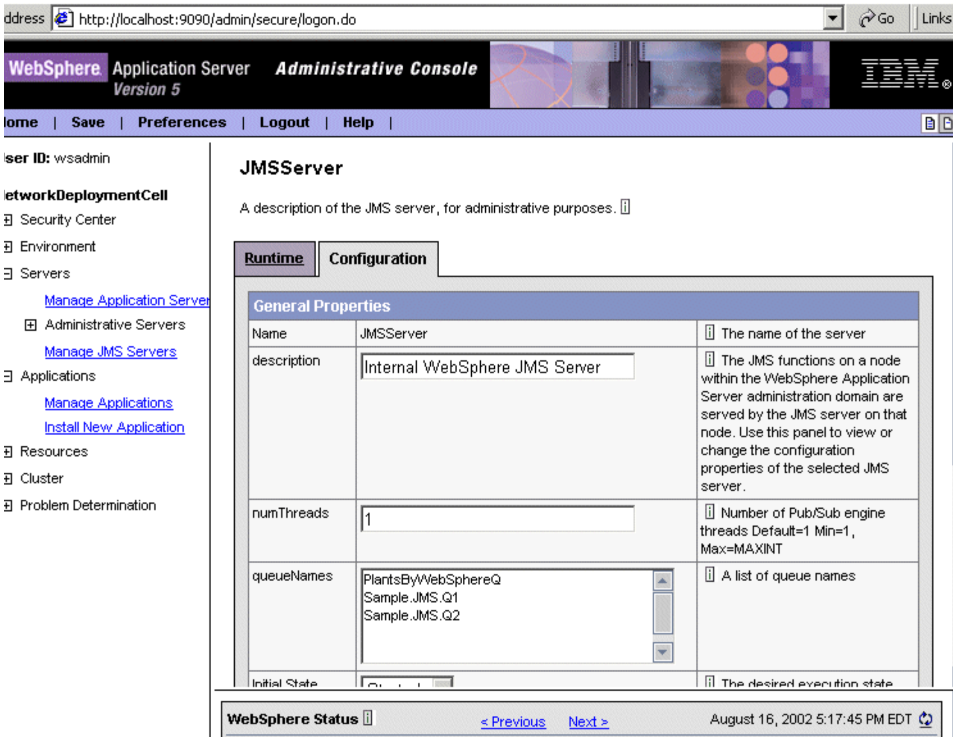
To configure the properties of a WebSphere integral JMS server process using the administrative console, complete the following steps:

1. In the navigation tree, select `Servers` -> `Manage JMS Servers`.

   This displays all the JMS Server processes configured in your cell (one per node in which embedded messaging has been installed), and their runtime status.

2. Click the name of the integral JMS Server process whose properties you want to edit.

3. This displays the properties of the JMS Server process in the content page, see Figure 18-4:



ddress http://localhost:9090/admin/secure/logon.do Go Links

WebSphere Application Server Version 5 Administrative Console

lome | Save | Preferences | Logout | Help |

ser ID: wsadmin

etworkDeploymentCell
- Security Center
- Environment
- Servers
  - Manage Application Server
  - Administrative Servers
  - Manage JMS Servers
- Applications
  - Manage Applications
  - Install New Application
- Resources
- Cluster
- Problem Determination

JMSServer

A description of the JMS server, for administrative purposes.

Runtime | Configuration

General Properties

Name	JMSServer	The name of the server
description	Internal WebSphere JMS Server	The JMS functions on a node within the WebSphere Application Server administration domain are served by the JMS server on that node. Use this panel to view or change the configuration properties of the selected JMS server.
numThreads	1	Number of Pub/Sub engine threads Default=1 Min=1, Max=MAXINT
queueNames	PlantsByWebSphereQ Sample.JMS.Q1 Sample.JMS.Q2	A list of queue names
Initial State		The desired execution state

WebSphere Status < Previous Next > August 16, 2002 5:17:45 PM EDT

Figure 18-4 Properties for the JMS Server process in one of the nodes

   a. **Name**

      The name by which the integral JMS server process is known for administrative purposes. The default name is *JMSServer*. This name cannot be changed.

   b. **Description**

A description of the JMS server process, for administrative purposes. The default is *Internal WebSphere JMS Server*. This string should not be changed.

c. **numThreads**

The number of concurrent threads to be used by the publish/subscribe matching engine.

The number of concurrent threads should only be set to a small number. The default value is 1.

d. **queueNames**

These are the actual messaging queues that get created for the integral WebSphere JMS provider. Each queue listed in this field must match the name of a WebSphere queue administrative object, (including the use of upper and lowercase). By adding the queue name to a specific JMS Server process we are specifying for which integral WebSphere JMS Provider (if there is more than one), the messaging queue gets created.

Note: There is no support for administering or managing these messaging queues, or the underlying Queue Manager or channels.

Complete the following steps to make a queue available to applications:

i. Define a queue administrative object under **`Resources`** -> **`JMS`** -> **`Configure WebSphere JMS Providers -> WebSphere Queue Destination`**

ii. For the JMS Server process on the host where the integral WebSphere JMS Provider is installed, go to the **General Properties** page, and add the administrative name of the WebSphere queue object to the **queueNames** field of the JMS Properties page.

To remove a queue from the JMS server process, remove its name from the **queueNames** field.

Note: When a queue is added to a JMS server process, the queue name is added to the list of queues hosted by that JMS server process. The list is defined in the jmsserver's *server.xml* configuration file.

e. **Initial State**

The initial state of the JMS Server process. It can be set to:

i. **Started**. The JMS server process is started automatically when you start the node.

Author Comment: This doesn't appear to be working at the moment

ii. **Stopped**. The JMS Server process is not started automatically.

For any deployed enterprise applications to use the JMS Server functions of a JMS Server process, the JMS Server process must be started.

4. On the JMS Server process **Configuration** page, under **Additional Properties,** view and change additional properties as necessary.

   For example, check that the ports used by the JMS Server process, are not being used in your system:

   a. **securityPort**

      The TCP/IP port used by the JMS Server process when security is enabled. The port number is configurable via **`Additional Properties`** -> **`securityPort`**. The default value is 5557.

      The security port number is listed in the *server.xml* configuration file for the jmsserver.

   b. **Queued Port**

      The TCP/IP port used by the JMS Server process to start the JMS Provider Queue Manager, for all point-to-point and full function publish/subscribe support. It is configurable via **`Additional Properties`** -> **`End Points`** -> **`JMSSERVER_QUEUED_ADDRESS`**. The default value is 5558.

      The JMS Server queued port number (JMSSERVER_QUEUED_ADDRESS) is listed in the *serverindex.xml* file for the node hosting the server, under the servername="jmsserver" stanza.

   c. **Direct Port**

      The TCP/IP port used by the JMS Server process for the default (high performance) publish/subscribe support. It is configurable via **`Additional Propertie`**s -> **`End Points`** -> **`JMSSERVER_DIRECT_ADDRESS`**. The default value is 5559.

      The JMS Server direct port number (JMSSERVER_DIRECT_ADDRESS) is listed in the *serverindex.xml* file for the node hosting the server, under the servername="jmsserver" stanza.

   d. **Soap Connector Address**

The SOAP JMX management port used by the JMS Server. The default value is 8876. The TCP/IP port number is configurable via **Additional Properties** -> **End Points** -> **SOAP_CONNECTOR_ADDRESS**.

The JMS Server SOAP connector address (SOAP_CONNECTOR_ADDRESS) is listed in the *serverindex.xml* file for the node hosting the server, under the servername=”jmsserver” stanza.

Note: See Chapter 7, “Asynchronous messaging” on page 147 for further information on the ports used by the JMS Server.

5. When you are happy with the changes, click **OK**.
6. To save the configuration, click **Save** on the task bar of the administrative console window.
7. To have the changed configuration take effect, stop then restart the JMS Server process.

To change the runtime status of a JMS Server process, complete the following steps:

1. In the navigation tree, select **Servers** -> **Manage JMS Servers**.
2. Select the integral JMS Server process that you want to act on.
3. Click **Start** to start the server, or **Stop** to stop it.

Note: The Queue Manager cannot be administered. It is started and stopped when you start and stop the JMS Server process for the Node.

18.4.2 Managing the WebSphere MQ JMS Provider

If you have installed WebSphere MQ as the JMS provider, over the integral JMS provider installed with WebSphere Application Server, complete the following steps to view the configuration properties of the WebSphere MQ JMS Provider:

1. In the navigation tree, expand **Resources** -> **JMS**.
2. Select **Configure WebSphere MQ JMS Providers**.
3. Under **Scope** select the **Node** in which the WebSphere MQ JMS Provider has been installed.
4. Click **Apply**.
5. Under **General Properties** you can view the properties for the WebSphere MQ JMS Provider:
   a. **Name**

The name by which the WebSphere MQ JMS provider is known, for administrative purposes. The default is *WebSphere MQ JMSProvider*.

b. **Description**

A description of the JMS provider, for administrative purposes. The default is *WebSphere MQ JMS Provider*.

c. **Classpath**

A list of paths or jar file names which together form the location for the resource provider classes. The default is ${MQJMS_LIB_ROOT} which is set to ${MQ_INSTALL_ROOT}/Java/lib.

Select **`Environment`** -> **`Manage WebSphere Variables`**, to view the values of these WebSphere variables.

d. **Native Path**

An optional path to any native libraries (.dll's, .so's). The default is also $(MQJMS_LIB_ROOT).

Note: You cannot change these properties from the JMS Provider Properties page.

6. Under **`Additional Properties`** you can view and change the resources for this JMS Provider.

18.4.3 Managing a Generic JMS Provider

If you are using a JMS provider other than the integral WebSphere JMS provider or the WebSphere MQ JMS provider, complete the following steps to configure the properties of the JMS Provider:

1. In the navigation tree, expand **`Resources`** -> **`JMS`**.
2. Select **`Manage Generic JMS Providers`**.
3. Under **`Scope`** select the **`Node`** in which the JMS Provider has been installed.
4. Click **`Apply`**.
5. Under **`General Properties`** configure the following properties:

a. **Name**

The name by which the JMS provider is known, for administrative purposes.

b. **Description**

A description of the JMS provider, for administrative purposes.

   c. **Classpath**

      A list of paths or jar file names which together form the location for the resource provider classes.

   d. **Native Path**

      An optional path to any native libraries (.dll's, .so's).

   e. **External initial context factory**

      The Java classname of the JMS Providers initial context factory.

      For example this would be `com.swiftmq.jndi.InitialContextFactoryImpl` for the SwiftMQ JMS Provider.

   f. **External provider URL**

      The JMS provider URL for external JNDI lookups.

      This specifies how JNDI lookups should be performed. Continuing with the example above, `smqp://localhost:4001`, would indicate that JNDI lookups should be performed by a JMS inbound listener on host localhost, port 4001. `smqp` is the SwiftMQ Protocol.

6. Click **`OK`** or **`Apply`**.
7. After clicking OK or Apply, use the **`Additional Properties`** panel to bind this JMS Provider resource into the WebSphere JNDI namespace.

Author Comment: What does this last point mean?

18.5 Configuring JMS Resources

This section describes how to configure the following JMS resources:

- Queue connection factory
- Topic connection factory
- Queue destination
- Topic destination

We will describe how these resources get configured into the WebSphere name space and how these resources get also **created** for the JMS Provider when using the following JMS providers:

- WebSphere integral JMS Provider.
- WebSphere MQ JMS Provider.
- Generic JMS Provider.

These administered objects are to be used by enterprise applications using JMS services.

18.5.1 Configuring resources for integral WebSphere JMS Provider

If you are using the integral WebSphere JMS provider in your environment, configure connection factories and destinations into the WebSphere name space:

- Configure queue connection factory.
- Configure topic connection factory.
- Configure queue destination.
- Configure topic destination.

Configure queue connection factory for integral WebSphere JMS Provider

A queue connection factory object is used to create JMS connections to the JMS Provider for point-to-point messaging with the JMS Provider queues (WebSphere MQ queues). We configure the queue connection factory objects into an application server’s name space via the administrative console.

Connection factory properties, control how connections are created to the associated JMS provider and the underlying queue destinations.

To configure properties for a queue connection factory object, which will be used to connect to an integral WebSphere JMS provider, complete these steps:

1. In the navigation tree, expand **Resources** -> **JMS** -> **Configure WebSphere JMS Providers**.
2. Select the **Scope** and click **Apply**.

   JMS resources are created at a scope level. When a JMS resource is created, the *resources.xml* file for that level gets updated with the new resource.

   Depending on the scope level selected, the resource will be available to one or more application servers. It is the application server’s JNDI name space that gets updated with the new resource.

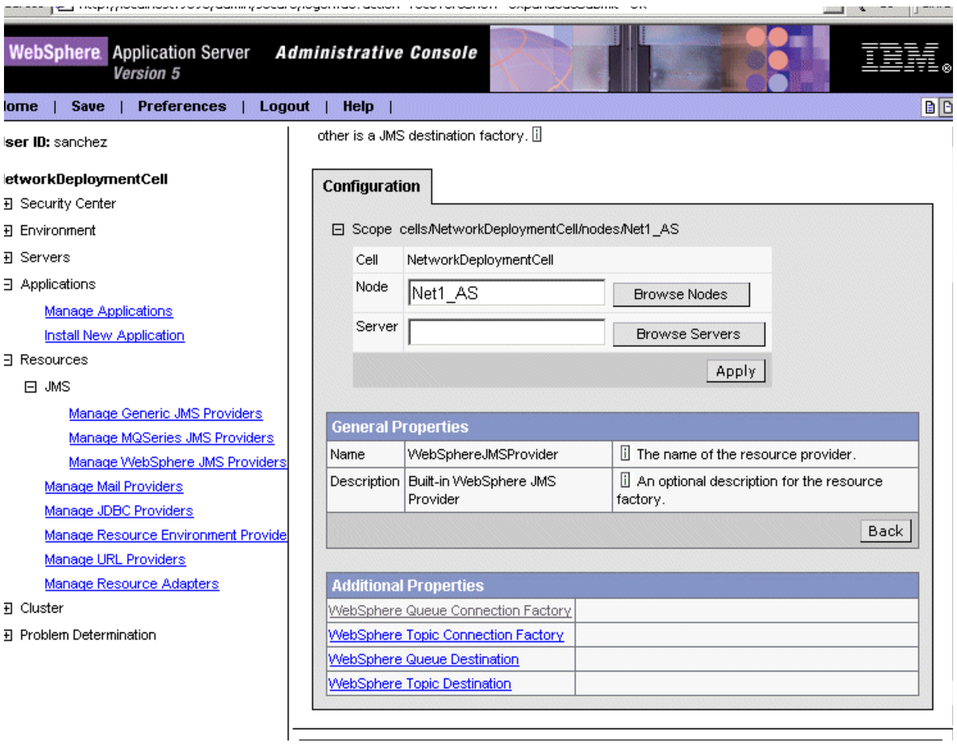
   For example, if we create a JMS resource such as a queue destination at the node level, this is what happens:

   a. The *resources.xml* file at the node level gets updated with the new queue.

   b. The local name space of each application server (but not the node agent) running under that node gets updated with the new resource.

c. Applications running under any of the application servers in that node can now use the resource, either from the local application server namespace, or from another server's namespace using the federated structure of the cell namespace.

Figure 18-5 shows the scope level entry set to node level:



other is a JMS destination factory.

Scope cells/NetworkDeploymentCell/nodes/Net1_AS

Cell	NetworkDeploymentCell
Node	Net1_AS
Server	

General Properties

Name	WebSphereJMSProvider	The name of the resource provider.
Description	Built-in WebSphere JMS Provider	An optional description for the resource factory.

Additional Properties

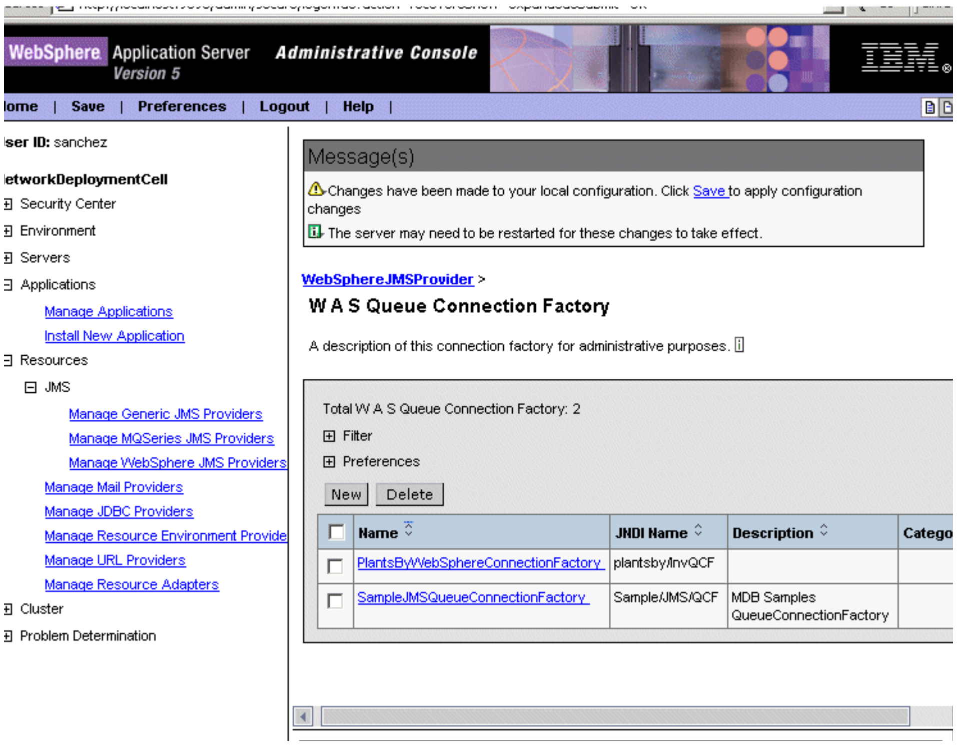
WebSphere Queue Connection Factory
WebSphere Topic Connection Factory
WebSphere Queue Destination
WebSphere Topic Destination

Figure 18-5 JMS Resource scoping determines which application servers can use the resource.

Note: The *resources.xml* file is of *merged* type scope, see Chapter 12, "System Management" on page 405, for more information on scope types.

3. Click on **WebSphere Queue Connection Factory** under **Additional Properties**.

   This displays WebSphere queue connection factory administered objects, available to application servers under the selected scope, and which are used to connect to an integral WebSphere JMS provider on the cell. This is shown in Figure 18-6:


WebSphere Application Server Version 5
Administrative Console
IBM
lome | Save | Preferences | Logout | Help |

ser ID: sanchez

etworkDeploymentCell
Security Center
Environment
Servers
Applications
Manage Applications
Install New Application
Resources
JMS
Manage Generic JMS Providers
Manage MQSeries JMS Providers
Manage WebSphere JMS Providers
Manage Mail Providers
Manage JDBC Providers
Manage Resource Environment Provide
Manage URL Providers
Manage Resource Adapters
Cluster
Problem Determination

Message(s)

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

The server may need to be restarted for these changes to take effect.

WebSphereJMSProvider >

W A S Queue Connection Factory

A description of this connection factory for administrative purposes.

Total W A S Queue Connection Factory: 2

Filter

Preferences

New Delete

	Name	JNDI Name	Description	Catego
☐	PlantsByWebSphereConnectionFactory	plantsby/InvQCF		
☐	SampleJMSQueueConnectionFactory	Sample/JMS/QCF	MDB Samples QueueConnectionFactory	

Figure 18-6 Queue connection factory administered objects at the selected scope

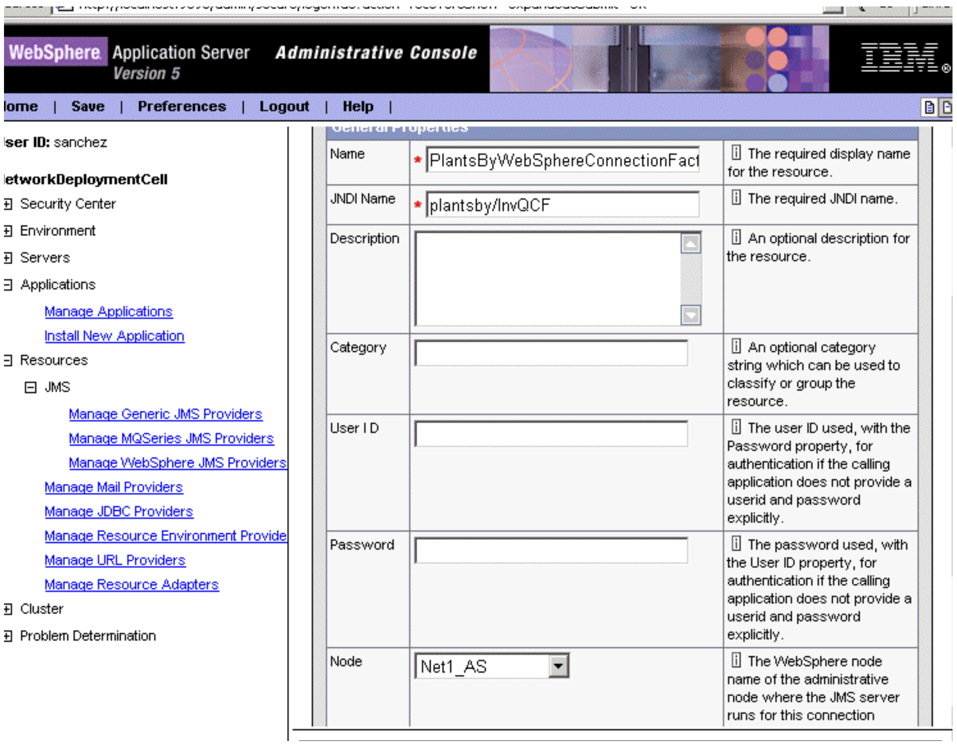
In this example we have two WebSphere queue connection factory objects, `PlantsByWebSphereConnectionFactory` and `SampleJMSQueueConnectionFactory`, available to all the application servers (and their applications) in the node.

These two queue connection factories have the properties to connect to an integral JMS provider in the cell.

4. To create a new queue connection factory object, click **`New`** in the content pane, see Figure 18-6.

   To change the properties of an existing queue connection factory, click one of the connection factories displayed.

   Figure 18-7, shows the properties for the `PlantsByWebSphereConnectionFactory` queue connection factory:



Name	* PlantsByWebSphereConnectionFact	The required display name for the resource.
JNDI Name	* plantsby/InvQCF	The required JNDI name.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
User ID		The user ID used, with the Password property, for authentication if the calling application does not provide a userid and password explicitly.
Password		The password used, with the User ID property, for authentication if the calling application does not provide a userid and password explicitly.
Node	Net1_AS	The WebSphere node name of the administrative node where the JMS server runs for this connection

Figure 18-7 Queue Connection factory properties

5. The configuration properties that need to be specified for a queue connection factory object that is to connect to an integral WebSphere JMS Provider (installed with WebSphere Application Server) are:

   a. **Name**

      The name by which this queue connection factory is known for administrative purposes.

   b. **JNDI name**

      The JNDI name used to bind the connection factory into the application server's name space. As a convention, use the form: `jms/Name`, where `Name` is the logical name of the resource.

      Figure 18-7, shows that the JNDI name for the `PlantsByWebSphereConnectionFactory` is `plantsby/InvQCF`. If we dump the namespace for one of the application servers in the node, we will see an entry for the queue connection factory object as follows:

      ```
      (top)/nodes/<node>/servers/<server>/plantsby/InvQCF
      ```

   c. **Description**

A description of this connection factory for administrative purposes.

d. **Category**

   A category used to classify or group this connection factory, for your administrative records.

e. **User ID**

   The user ID used for authentication if the calling application does not provide a userid and password explicitly. For example, if the calling application uses the method *createQueueConnection()* with no parameters.

   If a value is specified for the User ID property, a value must also be specified for the Password property.The JMS client passes the userid and password to the JMS server.

f. **Password**

   The password used, with the User ID property, for authentication if the calling application does not provide a userid and password explicitly.

g. **Node**

   The node name of the administrative node where the WebSphere integral JMS provider for this connection factory runs. Connections created by this factory connect to that integral JMS provider.

   Note: The node name is the WebSphere logical node name, not the hostname of the machine.

Other properties configurable under **Additional Properties** are **connection pool** settings and **session pool** settings.

Note: JMS Connection and session resources are managed by the J2C Connection Manager.

6. Click `OK`, when finished configuring the queue connection factory.
7. Click `Save`, to save the configuration.
8. To have the changed configuration take effect, stop then restart the application servers using the resource.

Configure topic connection factory for integral WebSphere JMS Provider

A topic connection factory object is used to create JMS connections to the JMS Publish/Subscribe Provider for publish/subscribe messaging to topic destinations. Topic connection factory objects are configured into the application servers name space using the administrative console.

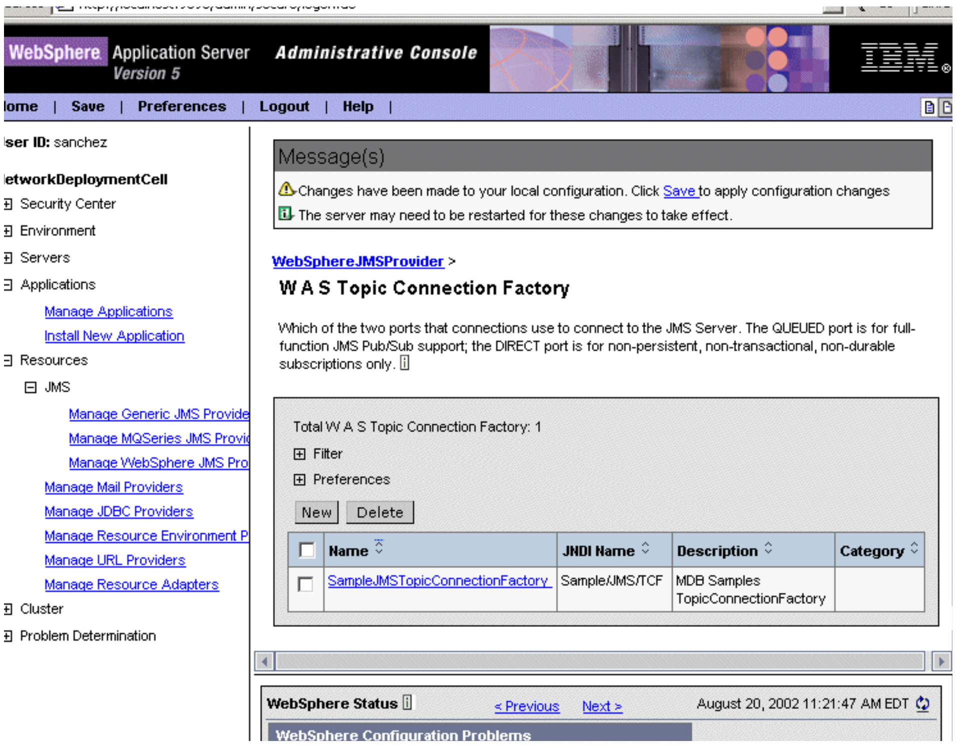
The topic connection factory properties control how connections are created to the associated JMS Publish/Subscribe Provider and the underlying topic destinations.

To configure the properties of a topic connection factory object, which will be used to connect to the Publish/Subscribe Provider installed with the integral WebSphere JMS provider, complete these steps:

1. In the navigation tree, expand **`Resources`** -> **`JMS`** -> **`Configure WebSphere JMS Providers`**.
2. Select the **`Scope`** and click **`Apply`**.

   For a discussion on the *Scope* setting, see, “Configure queue connection factory for integral WebSphere JMS Provider” on page 643.
3. Click on **`WebSphere Topic Connection Factory`** under **`Additional Properties`**.

   This displays WebSphere topic connection factory administered objects available to application servers under the selected scope, and which are used to connect to a Publish/Subscribe JMS Server installed with an integral WebSphere JMS provider. This is shown in Figure 18-8:


WebSphere Application Server Version 5
Administrative Console
IBM
lome | Save | Preferences | Logout | Help |

ser ID: sanchez

etworkDeploymentCell
- Security Center
- Environment
- Servers
- Applications
  - Manage Applications
  - Install New Application
- Resources
  - JMS
    - Manage Generic JMS Provide
    - Manage MQSeries JMS Provid
    - Manage WebSphere JMS Pro
  - Manage Mail Providers
  - Manage JDBC Providers
  - Manage Resource Environment P
  - Manage URL Providers
  - Manage Resource Adapters
- Cluster
- Problem Determination

Message(s)

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

The server may need to be restarted for these changes to take effect.

WebSphereJMSProvider >

W A S Topic Connection Factory

Which of the two ports that connections use to connect to the JMS Server. The QUEUED port is for full-function JMS Pub/Sub support; the DIRECT port is for non-persistent, non-transactional, non-durable subscriptions only.

Total W A S Topic Connection Factory: 1

Filter

Preferences

New Delete

	Name	JNDI Name	Description	Category
☐	SampleJMSTopicConnectionFactory	Sample/JMS/TCF	MDB Samples TopicConnectionFactory	

WebSphere Status < Previous Next > August 20, 2002 11:21:47 AM EDT

WebSphere Configuration Problems

Figure 18-8 Topic Connection Factory objects at the selected scope

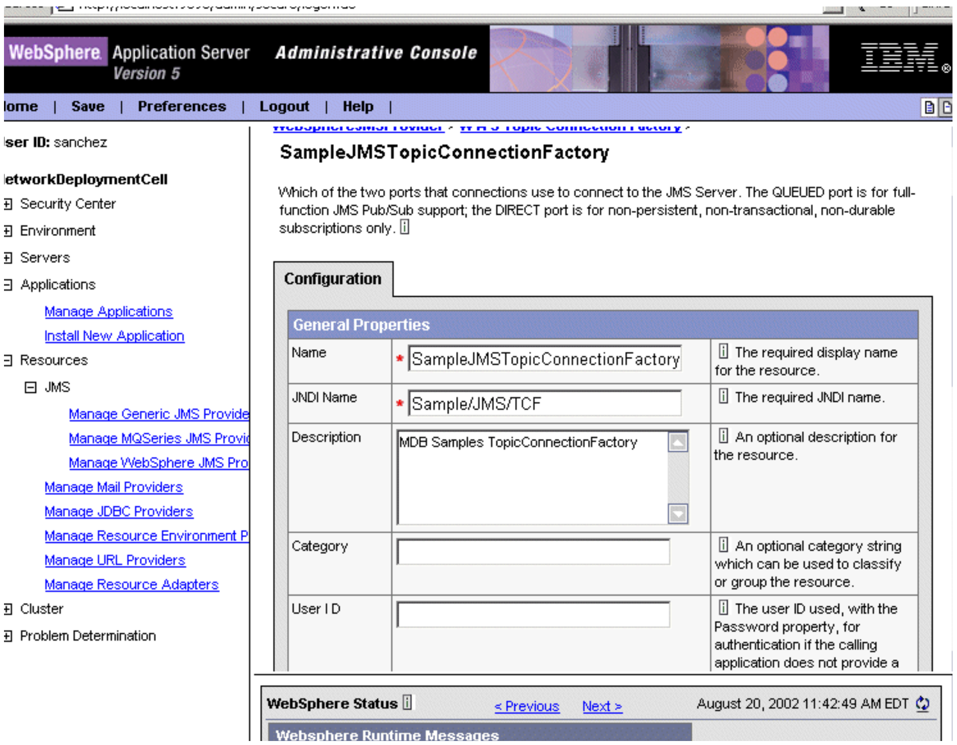
In this example we have one WebSphere topic connection factory object, `SampleJMSTopicConnectionFactory`, available to all the application servers (and their applications) in the node.

This topic connection factory object is used to connect to an integral JMS provider in the cell.

4. To create a new topic connection factory object, click **New** in the content pane.

   To change the properties of an existing topic connection factory, click one of the connection factories displayed.

   Figure 18-9, shows the properties for the `SampleJMSTopicConnectionFactory` topic connection factory object:


WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |

User ID: sanchez

NetworkDeploymentCell
- Security Center
- Environment
- Servers
- Applications
  - Manage Applications
  - Install New Application
- Resources
  - JMS
    - Manage Generic JMS Provide
    - Manage MQSeries JMS Provi
    - Manage WebSphere JMS Pro
  - Manage Mail Providers
  - Manage JDBC Providers
  - Manage Resource Environment P
  - Manage URL Providers
  - Manage Resource Adapters
- Cluster
- Problem Determination

SampleJMSTopicConnectionFactory

Which of the two ports that connections use to connect to the JMS Server. The QUEUED port is for full-function JMS Pub/Sub support; the DIRECT port is for non-persistent, non-transactional, non-durable subscriptions only.

Configuration

General Properties

Name	* SampleJMSTopicConnectionFactory	The required display name for the resource.
JNDI Name	* Sample/JMS/TCF	The required JNDI name.
Description	MDB Samples TopicConnectionFactory	An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
User I D		The user ID used, with the Password property, for authentication if the calling application does not provide a

WebSphere Status < Previous Next > August 20, 2002 11:42:49 AM EDT

Websphere Runtime Messages

Figure 18-9 Topic Connection factory properties

5. The configuration properties that need to be specified for a topic connection factory object that is to be used with an integral WebSphere JMS Provider (installed with WebSphere Application Server) are:

   a. **Name**

      The name by which this topic connection factory is known for administrative purposes.

   b. **JNDI name**

      The JNDI name used to bind the topic connection factory into the application server's name space. As a convention, use the form: `jms/Name`, where `Name` is the logical name of the resource.

      Figure 18-9, shows that the JNDI name for the `SampleJMSTopicConnectionFactory object` is `Sample/JMS/TCF`. If we dump the namespace for one of the application servers in the node (**`dumpnamespace`**), we will see an entry for the topic connection factory object as follows:

```
(top)/nodes/<node>/servers/<server>/Sample/JMS/TCF
```

c. **Description**

   A description of this topic connection factory for administrative purposes.

d. **Category**

   A category used to classify or group this topic connection factory, for your administrative records.

e. **User ID**

   The user ID used for authentication if the calling application does not provide a userid and password explicitly. For example, if the calling application uses the method *createTopicConnection()* with no parameters.

   If a value is specified for the User ID property, a value must also be specified for the Password property.The JMS client passes the userid and password to the Publish/Subscribe JMS Provider.

f. **Password**

   The password used, with the User ID property, for authentication if the calling application does not provide a userid and password explicitly.

g. **Node**

   The node name of the administrative node where the WebSphere integral JMS provider (Publish/Subscribe JMS Provider) for this topic connection factory runs. Connections created by this factory connect to that Publish/Subscribe JMS Provider.

   Note: The node name is the WebSphere logical node name, not the hostname of the machine.

h. **Port**

   The port used to connect to the Publish/Subscribe JMS Provider. The two possible values are:

   i. QUEUED

      This is the port used by the JMS Provider Queue Manager. If the topic connection factory object is configured with Port set to QUEUED, the JMS Publish/Subscribe Provider is accessed via the integral JMS Provider Queue Manager. The queued port is the port used for full-function JMS Publish/Subscribe support.

   ii. DIRECT

      The direct port is the Publish/Subscribe JMS Provider listener port used for non-persistent, non-transactional, non-durable subscriptions. If the topic connection factory object is configured with Port set to

DIRECT, the JMS Publish/Subscribe Provider is accessed direct on this port via TCP/IP.

Note: Message-driven beans cannot use the DIRECT listener port for Publish/Subscribe support. Therefore, any topic connection factory object configured with Port set to DIRECT cannot be used with message-driven beans.

The TCP/IP listener ports to access the Publish/Subscribe JMS Provider are defined in the *serverindex.xml* file for the node in which the WebSphere integral JMS Provider is installed, under the *servername=”jmsserver”* stanza.

i. **Client ID**

The JMS client identifier used for connections to the Publish/Subscribe JMS Provider.

j. **Clone Support**.

Author Comment: Clone support is not documented for the WebSphere integral Provider. What is it?

Other properties configurable under **Additional Properties** are:

- **Connection pool** settings.
- **Session pool** settings.

Note: JMS pooling is managed by the J2C Connection Manager.

6. Click `OK`, when you have finished configuring the topic connection factory object.
7. Click `Save`, to save the configuration.
8. To have the changed configuration take effect, stop then restart the application servers using the topic connection factory object.

Configure queue destination for integral WebSphere JMS Provider

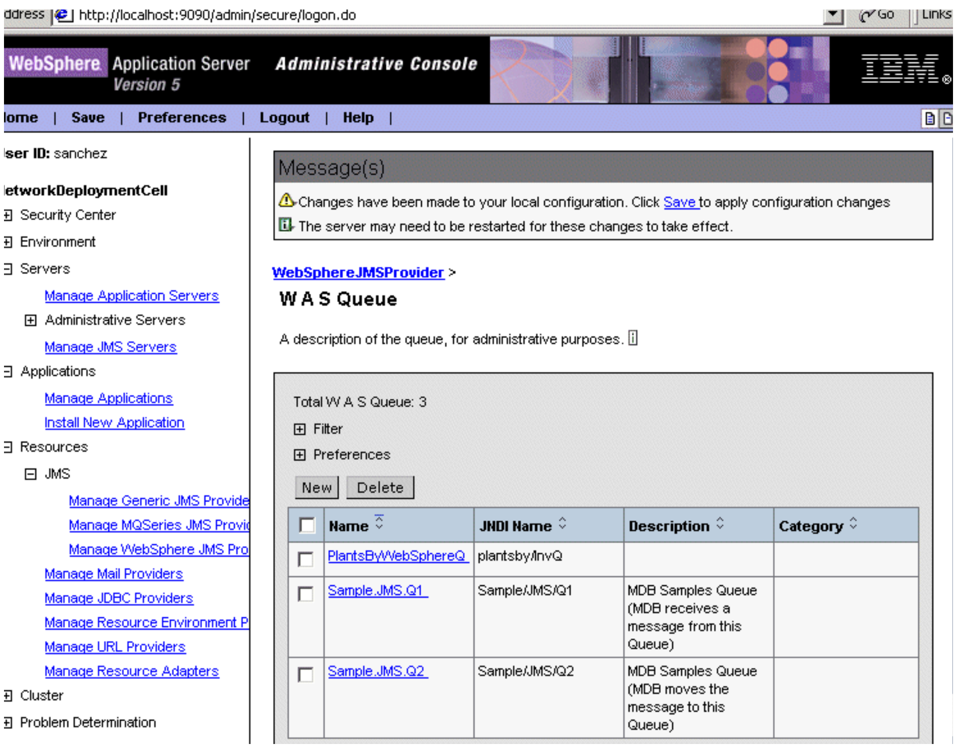
A queue destination object is a WebSphere administrative object used to represent a JMS messaging system queue.

Connections to the queue are made via a queue connection factory object that connects to the integral WebSphere JMS Provider.

To configure the properties of a queue destination object, that represents a queue hosted by the integral WebSphere JMS provider, complete these steps:

1. In the navigation tree, expand **Resources** -> **JMS** -> **Configure WebSphere JMS Providers**.
2. Select the **Scope** and click **Apply**. For a discussion on the *Scope* setting, see, “Configure queue connection factory for integral WebSphere JMS Provider” on page 643.
3. Click on **WebSphere Queue Destination** under **Additional Properties**.

   This displays WebSphere queue destination objects available to application servers under the selected scope. These objects represent messaging queues hosted by an integral WebSphere JMS provider in the cell. This is shown in Figure 18-10:



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

The server may need to be restarted for these changes to take effect.

WebSphereJMSProvider >

W A S Queue

A description of the queue, for administrative purposes.

Total W A S Queue: 3

	Name	JNDI Name	Description	Category
☐	PlantsByWebSphereQ	plantsby/InvQ		
☐	Sample.JMS.Q1	Sample/JMS/Q1	MDB Samples Queue (MDB receives a message from this Queue)	
☐	Sample.JMS.Q2	Sample/JMS/Q2	MDB Samples Queue (MDB moves the message to this Queue)	

Figure 18-10 Queue destination administered objects at the selected scope

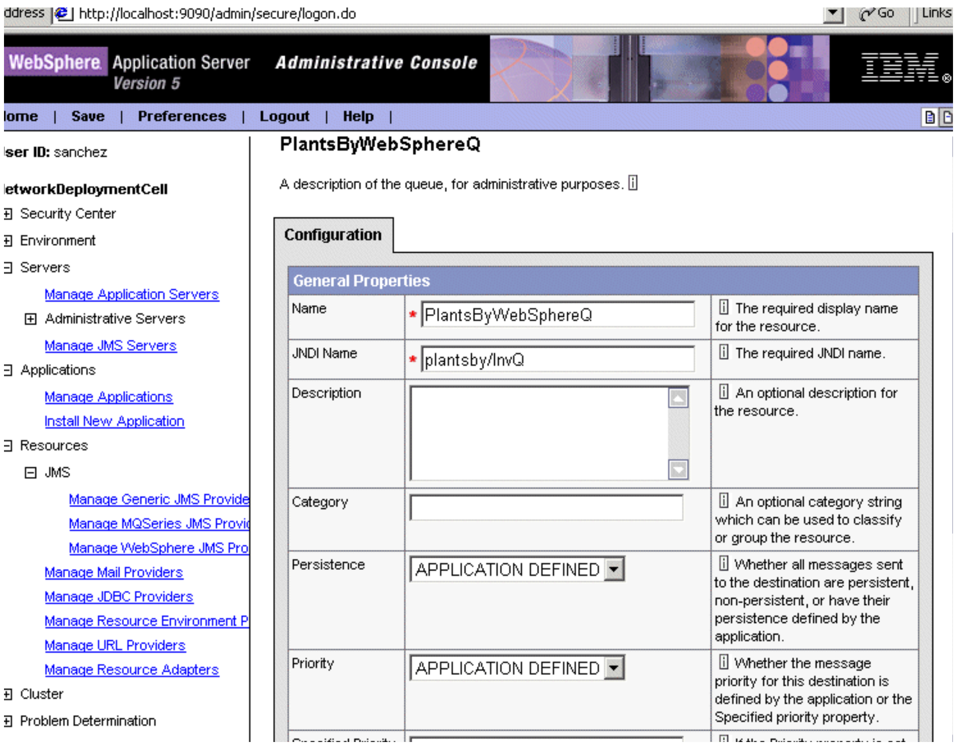
In this example we have three WebSphere queue destination objects, `PlantsByWebSphereQ`, `Sample.JMS.Q1 and Sample.JMS.Q2`, available to all the application servers (and their applications) in the node.

These three queue destination objects represent queues hosted by an integral JMS provider in the cell.

4. To create a new queue destination object, click **`New`** in the content pane, see Figure 18-10.

   To change the properties of an existing queue destination object, click one of the queue destination objects displayed.

   Figure 18-11, shows the properties for the `PlantsByWebSphereQ` queue destination object:



PlantsByWebSphereQ

A description of the queue, for administrative purposes.

General Properties

Name	* PlantsByWebSphereQ	The required display name for the resource.
JNDI Name	* plantsby/InvQ	The required JNDI name.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Persistence	APPLICATION DEFINED	Whether all messages sent to the destination are persistent, non-persistent, or have their persistence defined by the application.
Priority	APPLICATION DEFINED	Whether the message priority for this destination is defined by the application or the Specified priority property.

Figure 18-11 Queue Destination object properties

5. The configuration properties that need to be specified for a queue connection object, that represents a queue hosted by the integral WebSphere JMS Provider are:

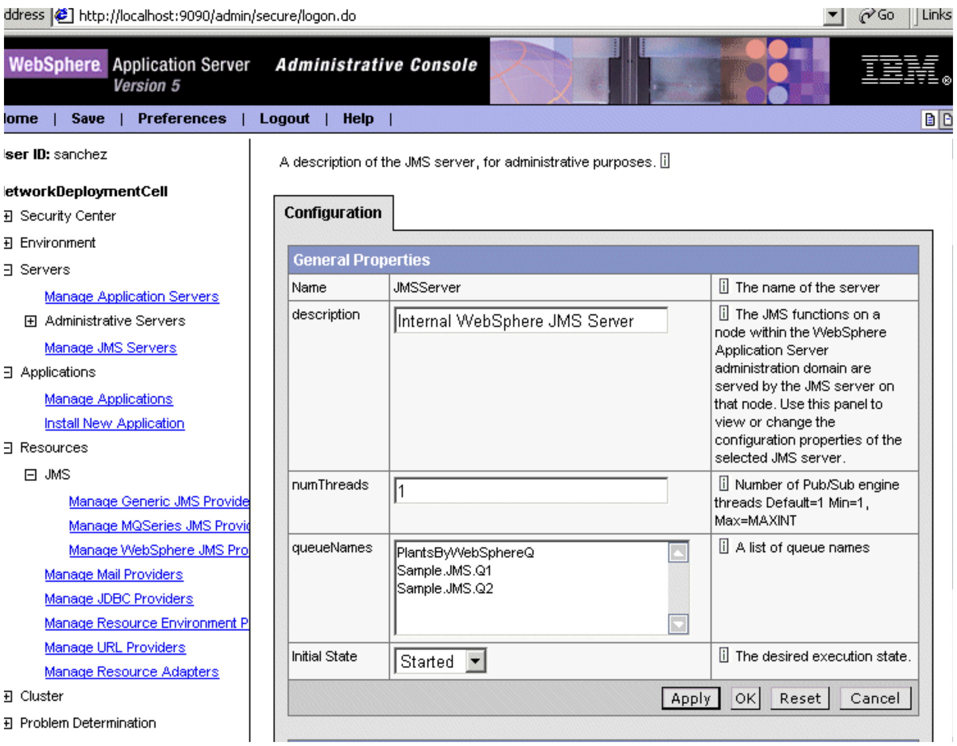
   a. **Name**

      The name by which the queue is known for administrative purposes.

Note: To make a queue destination object available to applications, you need to add the administrative name of the queue to the JMS Server process on the node that hosts the JMS Server. This was described in Section , “Managing the WebSphere integral JMS Server process” on page 636.

Adding the queue name to the JMS Server **creates** the messaging queue for the integral JMS Provider, though you won’t be able to see it nor administer it.

Figure 18-12 shows how the `PlantsByWebSphereQ` queue from the current example has been added to the JMS Server process on the node where the integral WebSphere JMS Server Provider is installed.



A description of the JMS server, for administrative purposes.

General Properties

Name	JMSServer	The name of the server
description	Internal WebSphere JMS Server	The JMS functions on a node within the WebSphere Application Server administration domain are served by the JMS server on that node. Use this panel to view or change the configuration properties of the selected JMS server.
numThreads	1	Number of Pub/Sub engine threads Default=1 Min=1, Max=MAXINT
queueNames	PlantsByWebSphereQ Sample.JMS.Q1 Sample.JMS.Q2	A list of queue names
Initial State	Started	The desired execution state.

Figure 18-12 Adding a queue to the JMS Server process creates a messaging queue for the JMS Provider.

b. JNDI name

The JNDI name used to bind the queue into the application server's name space. As a convention, use the form: `jms/Name`, where `Name` is the logical name of the resource.

Figure 18-11, shows that the JNDI name for the `PlantsByWebSphereQ` queue destination object is `plantsby/InvQ`. If we dump the namespace for one of the application servers in the node, we will see an entry for the queue destination object as follows:

```
(top)/nodes/<node>/servers/<server>/plantsby/InvQ
```

c. **Description**

   A description of the queue, for administrative purposes.

d. **Category**

   A category used to classify or group this queue, for your administrative records.

e. **Persistence**

   Specifies whether all messages sent to the destination are PERSISTENT, NON PERSISTENT, or have their persistence defined by the application, APPLICATION DEFINED.The default is APPLICATION DEFINED. It is the application that puts the message onto the queue that instructs the JMS Provider whether to store the message in case the Provider fails.

f. **Priority**

   Specifies whether the message priority for this destination is defined by the application, APPLICATION DEFINED, or whether we specify the priority, SPECIFIED. The default is APPLICATION DEFINED. It is the application that puts the message onto the queue that defines the message priority.

g. **Specified Priority**

   If the **Priority** is set to SPECIFIED, type here the message priority for this queue,. The ten levels of priority range from 0 (lowest) to 9 (highest). Messages sent to this queue have the priority value specified by this property.

h. **Expiry**

   Specifies whether the expiry timeout for this queue is defined by the application, APPLICATION DEFINED, whether we specify the expiration time, SPECIFIED, or whether the message on the queue never expires, UNLIMITED. The default is APPLICATION DEFINED. It is the application that puts the message onto the queue that defines whether the message is destroyed after a period of time.

i. **Specified Expiry**

If the Expiry property is set to SPECIFIED, type here the number of milliseconds (greater than 0) after which messages on this queue expire.0 indicates that messages never timeout.

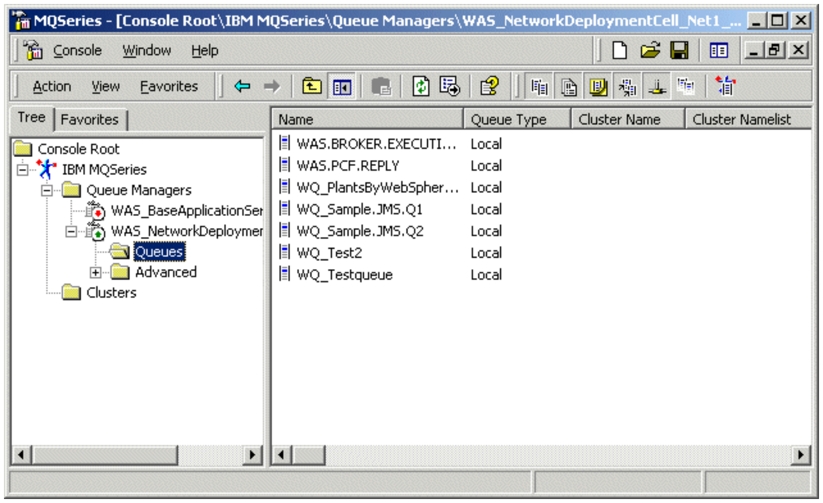
6. Click **OK**, when finished configuring the queue destination object.
7. Click **Save**, to save the configuration.
8. To have the changed configuration take effect, stop then restart the application servers using the resource.

Note: If you install the full version of WebSphere MQ after installing the integral JMS provider, existing JMS resource definitions for the integral WebSphere JMS provider will continue to work with WebSphere MQ as the JMS provider, so you do not need to redefine those JMS resources.

Figure 18-13 for example, shows a screenshot of WebSphere MQ Explorer after installing WebSphere MQ over the integral JMS Provider. We can see the queue managers previously created when installing the integral JMS Provider, such as `WAS_<cellname>_Net1_AS`, and the queues defined to the JMS Server process, such as `Sample.JMS.Q1`.

The queues actually created in the internal Queue Manager have the name WQ_<JMS Server queuename>.

With WebSphere MQ Explorer, we can now manage those queues. This was not possible with the integral JMS Provider.



Name	Queue Type	Cluster Name	Cluster Namelist
WAS.BROKER.EXECUTI...	Local		
WAS.PCF.REPLY	Local		
WQ_PlantsByWebSpher...	Local		
WQ_Sample.JMS.Q1	Local		
WQ_Sample.JMS.Q2	Local		
WQ_Test2	Local		
WQ_Testqueue	Local		

Figure 18-13 Installing WebSphere MQ over the integral WebSphere JMS Provider

Configure topic destination for integral WebSphere JMS Provider

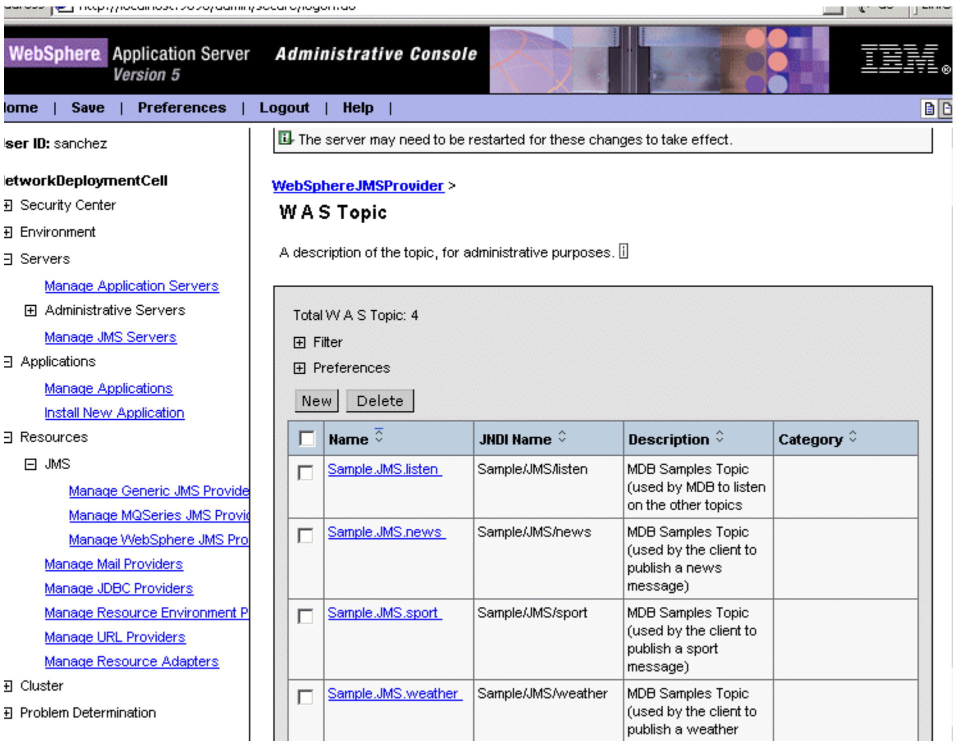
A topic destination is used to configure the properties of a WebSphere administrative object that represents a JMS topic in the messaging system.

Connections to the topic are created by a topic connection factory that connects to the Publish/Subscribe Provider installed with the integral WebSphere JMS Provider.

To configure the properties of a topic destination administrative object, which represents a topic served by the Publish/Subscribe JMS Provider that gets installed with the integral WebSphere JMS provider, complete these steps:

1. In the navigation tree, expand **`Resources`** -> **`JMS`** -> **`Configure WebSphere JMS Providers`**.
2. Select the **`Scope`** and click **`Apply.`**For a discussion on the *Scope* setting, see, “Configure queue connection factory for integral WebSphere JMS Provider” on page 643.
3. Click on **`WebSphere Topic Destination`** under **`Additional Properties`**.

   This displays WebSphere topic destination objects available to application servers under the selected scope. These objects represent actual topics served by an integral Publish/Subscribe JMS provider on the cell. This is shown in Figure 18-14:



The server may need to be restarted for these changes to take effect.

WebSphereJMSProvider >

W A S Topic

A description of the topic, for administrative purposes.

Total W A S Topic: 4

Filter

Preferences

New Delete

	Name	JNDI Name	Description	Category
☐	Sample.JMS.listen	Sample/JMS/listen	MDB Samples Topic (used by MDB to listen on the other topics	
☐	Sample.JMS.news	Sample/JMS/news	MDB Samples Topic (used by the client to publish a news message)	
☐	Sample.JMS.sport	Sample/JMS/sport	MDB Samples Topic (used by the client to publish a sport message)	
☐	Sample.JMS.weather	Sample/JMS/weather	MDB Samples Topic (used by the client to publish a weather	

Figure 18-14 Topic destination administered objects at the selected scope

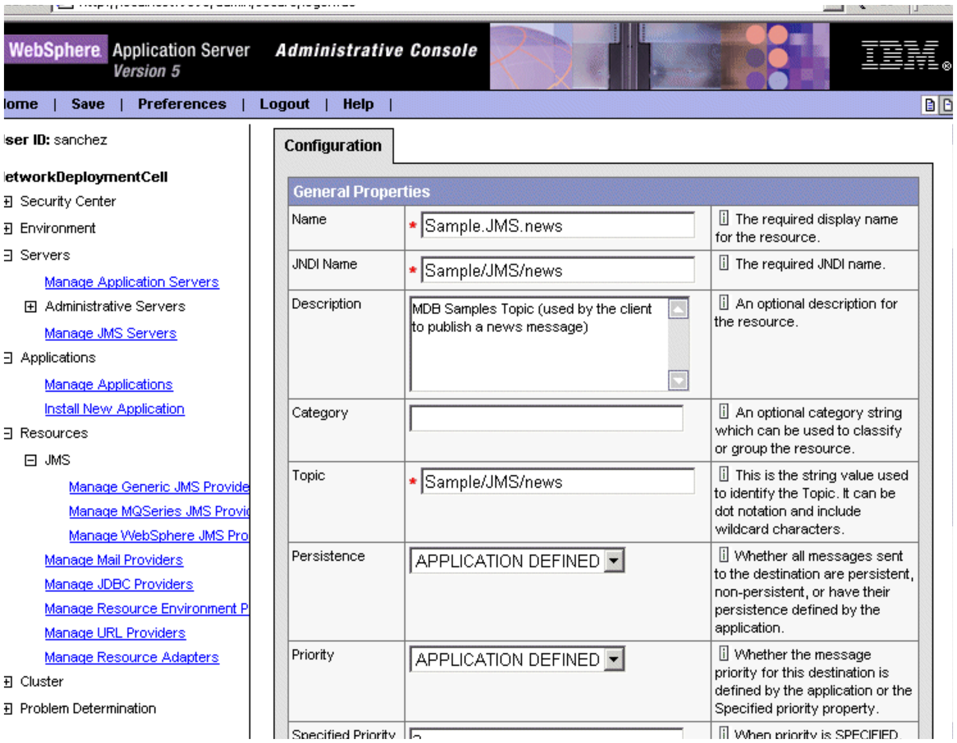
In this example we have four WebSphere topic destination objects, Sample.JMS.listen, Sample.JMS.news, Sample.JMS.sport and Sample.JMS.weather, available to all the application servers (and their applications) in my node.

These four topic destination objects represent topics served by the Publish/Subscribe Provider installed with the integral JMS provider.

4. To create a new topic destination object, click **`New`** in the content pane, as shown in Figure 18-14.

   To change the properties of an existing topic destination object, click one of the topic destination objects displayed.

   Figure 18-15, shows the properties for the `Sample.JMS.news topic` destination object:


WebSphere Application Server Version 5
Administrative Console
IBM
lome | Save | Preferences | Logout | Help |

ser ID: sanchez

letworkDeploymentCell
- Security Center
- Environment
- Servers
  - Manage Application Servers
  - Administrative Servers
  - Manage JMS Servers
- Applications
  - Manage Applications
  - Install New Application
- Resources
  - JMS
    - Manage Generic JMS Provide
    - Manage MQSeries JMS Provic
    - Manage WebSphere JMS Pro
  - Manage Mail Providers
  - Manage JDBC Providers
  - Manage Resource Environment P
  - Manage URL Providers
  - Manage Resource Adapters
- Cluster
- Problem Determination

Configuration

General Properties

Name	* Sample.JMS.news	The required display name for the resource.
JNDI Name	* Sample/JMS/news	The required JNDI name.
Description	MDB Samples Topic (used by the client to publish a news message)	An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Topic	* Sample/JMS/news	This is the string value used to identify the Topic. It can be dot notation and include wildcard characters.
Persistence	APPLICATION DEFINED	Whether all messages sent to the destination are persistent, non-persistent, or have their persistence defined by the application.
Priority	APPLICATION DEFINED	Whether the message priority for this destination is defined by the application or the Specified priority property.
Specified Priority		When priority is SPECIFIED

Figure 18-15 Topic Destination object properties

5. The configuration properties that need to be specified for a topic destination object, that represents a topic served by the Publish/Subscribe Provider installed with the integral WebSphere JMS Provider are:

   a. **Name**

      The name by which the topic is known for administrative purposes.

   b. **JNDI name**

      The JNDI name used to bind the topic destination into the application server's name space. As a convention, use the form: `jms/Name`, where `Name` is the logical name of the resource.

      Figure 18-15, shows that the JNDI name for the Sample.JMS.news topic destination object is `Sample/JMS/news`. If we dump the namespace for one of the application servers in the node, we will see an entry for the topic destination object as follows:

      ```
      (top)/nodes/<node>/servers/<server>/Sample/JMS/news
      ```

   c. **Description**

A description of the topic, for administrative purposes.

d. **Category**

A category used to classify or group this topic, for your administrative records.

e. **Topic**

This is the name of the topic defined to the Publish/Subscribe JMS Provider.

Note: The message topic for the Publish/Subscribe JMS Provider gets created dynamically when the application invokes the administered object.

f. **Persistence**

Specifies whether all messages sent to this topic destination are PERSISTENT, NON PERSISTENT, or have their persistence defined by the application, APPLICATION DEFINED.The default is APPLICATION DEFINED. It is the application that puts the message onto the topic destination that instructs the Publish/Subscribe JMS Provider whether to store the message in case the Provider fails.

g. **Priority**

Specifies whether the message priority for this topic destination is defined by the application, APPLICATION DEFINED, or whether we specify the priority, SPECIFIED. The default is APPLICATION DEFINED. It is the application that puts the message onto the topic destination that defines the message priority.

h. **Specified Priority**

If the **Priority** is set to SPECIFIED, type here the message priority for this topic. The ten levels of priority range from 0 (lowest) to 9 (highest). Messages sent to this topic have the priority value specified by this property.

i. **Expiry**

Specifies whether the expiry timeout for this topic destination is defined by the application, APPLICATION DEFINED, whether we specify the expiration time, SPECIFIED, or whether the message on the topic destination never expires, UNLIMITED. The default is APPLICATION DEFINED. It is the application that puts the message onto the topic destination that defines whether the message is destroyed after a period of time.

j. **Specified Expiry**

If the Expiry property is set to SPECIFIED, type here the number of milliseconds (greater than 0) after which messages on this topic destination expire.0 indicates that messages never timeout.

6. Click **OK**, when finished configuring the topic destination object.
7. Click **Save**, to save the configuration.
8. To have the changed configuration take effect, stop then restart the application servers using the resource.

18.5.2 Configuring resources for the WebSphere MQ JMS Provider

If you are using the WebSphere MQ JMS provider or a Publish/Subscribe Provider installed with WebSphere MQ, to support enterprise applications using JMS, you can still use the administrative console to configure connection factories and destination objects into the WebSphere name space.

The administrative objects that you configure in WebSphere must match the JMS messaging resources created in WebSphere MQ or the Publish/Subscribe Provider:

- Configure queue connection factory.
- Configure topic connection factory.
- Configure queue destination.
- Configure topic destination.

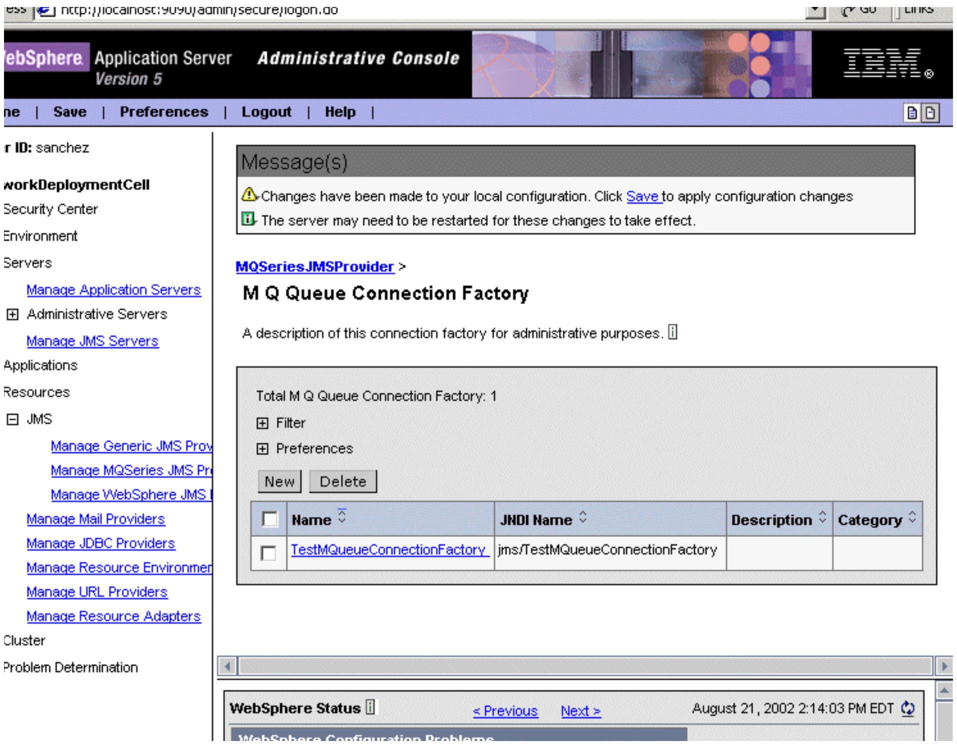
Note: WebSphere MQ resources do not get created automatically for you, as it was the case with the integral JMS Provider. You need to create those messaging resources in WebSphere MQ , using WebSphere MQ Explorer wizard for example.

Configure queue connection factory for WebSphere MQ JMS Provider

Complete the following steps, to configure properties for a queue connection factory object, that will be used to connect to an WebSphere MQ JMS provider:

1. In the navigation tree, expand **Resources** -> **JMS** -> **Configure WebSphere MQ JMS Providers**.
2. Select the **Scope** and click **Apply.** For a discussion on the *Scope* setting, see, “Configure queue connection factory for integral WebSphere JMS Provider” on page 643.
3. Click on **WebSphere MQ Queue Connection Factory** under **Additional Properties**.

This displays WebSphere MQ queue connection factory administered objects, available to application servers under the selected scope, and which are used to connect to an WebSphere MQ JMS provider. This is shown in Figure 18-16:


ebSphere Application Server Version 5
Administrative Console
IBM
ne | Save | Preferences | Logout | Help |

r ID: sanchez

workDeploymentCell
Security Center
Environment
Servers
Manage Application Servers
Administrative Servers
Manage JMS Servers
Applications
Resources
JMS
Manage Generic JMS Prov
Manage MQSeries JMS Pr
Manage WebSphere JMS
Manage Mail Providers
Manage JDBC Providers
Manage Resource Environmer
Manage URL Providers
Manage Resource Adapters
Cluster
Problem Determination

Message(s)

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

The server may need to be restarted for these changes to take effect.

MQSeriesJMSProvider >

M Q Queue Connection Factory

A description of this connection factory for administrative purposes.

Total M Q Queue Connection Factory: 1

Filter

Preferences

New Delete

	Name	JNDI Name	Description	Category
☐	TestMQueueConnectionFactory	jms/TestMQueueConnectionFactory		

WebSphere Status < Previous Next > August 21, 2002 2:14:03 PM EDT

Figure 18-16 MQQueue connection factory administered objects at the selected scope

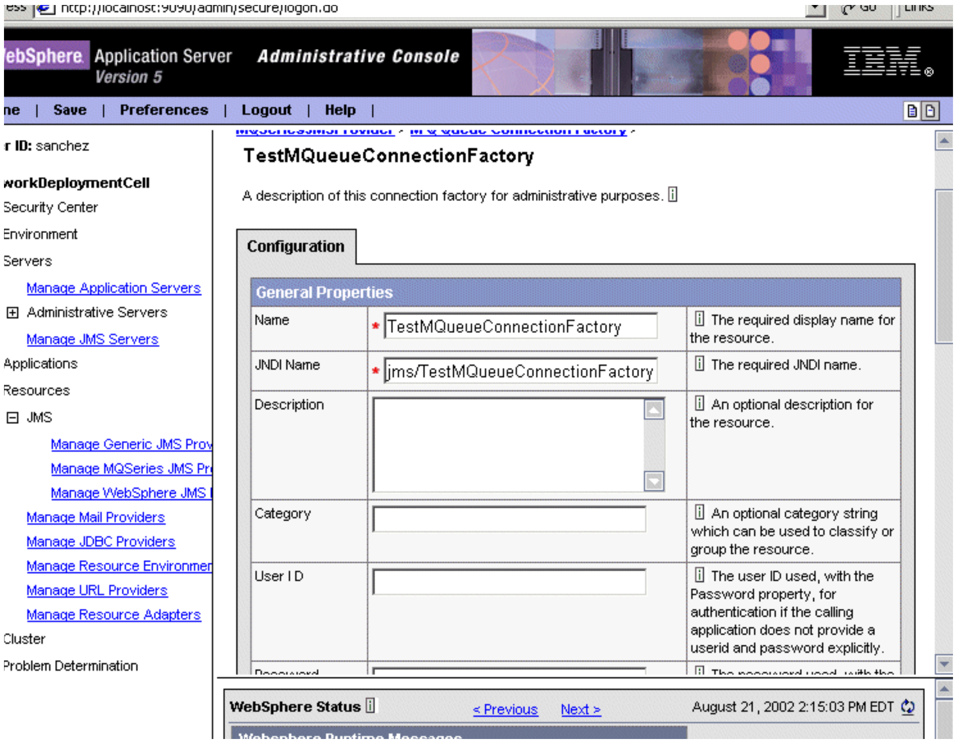
In this example we have one WebSphere MQ queue connection factory object, `TestMQueueConnectionFactory`, available to all the application servers (and their applications) in the node.

This connection factory object has all the necessary properties to connect to a full WebSphere MQ JMS provider.

4. To create a new queue connection factory object, click **`New`** in the content pane, see Figure 18-16.

   To change the properties of an existing queue connection factory, click one of the connection factories displayed.

   Figure 18-17, partially shows the properties for the `TestMQueueConnectionFactory` WebSphere MQ queue connection factory object:


ebSphere Application Server Version 5 Administrative Console IBM.

ne | Save | Preferences | Logout | Help |

r ID: sanchez

workDeploymentCell
Security Center
Environment
Servers
Manage Application Servers
⊞ Administrative Servers
Manage JMS Servers
Applications
Resources
⊟ JMS
Manage Generic JMS Prov
Manage MQSeries JMS Pr
Manage WebSphere JMS
Manage Mail Providers
Manage JDBC Providers
Manage Resource Environmen
Manage URL Providers
Manage Resource Adapters
Cluster
Problem Determination

TestMQueueConnectionFactory

A description of this connection factory for administrative purposes.

Configuration

General Properties

Name	* TestMQueueConnectionFactory	The required display name for the resource.
JNDI Name	* jms/TestMQueueConnectionFactory	The required JNDI name.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
User ID		The user ID used, with the Password property, for authentication if the calling application does not provide a userid and password explicitly.

WebSphere Status < Previous Next > August 21, 2002 2:15:03 PM EDT

Figure 18-17 WebSphere MQ queue connection factory properties

5. The configuration properties that need to be specified for a queue connection factory object that is to connect to an WebSphere MQ JMS Provider are:

   a. **Name**

      The name by which this queue connection factory is known for administrative purposes.

Author Comment: The uniqueness of the resource name needs to be checked. This applies to all resources for all providers.

   b. **JNDI name**

      The JNDI name used to bind the connection factory into the application server's name space. As a convention, use the form: `jms/Name`, where `Name` is the logical name of the resource.

      Figure 18-17, shows that the JNDI name for the `TestMQueueConnectionFactory` queue connection factory object is `jms/TestMQueueConnectionFactory`. If we dump the namespace for one of the application servers in the node, we will see an entry for the queue connection factory object as follows:

```
(top)/nodes/<node>/servers/<server>/jms/TestMQueueConnectionFactory
```

c. **Description**

A description of this connection factory for administrative purposes.

d. **Category**

A category used to classify or group this connection factory, for your administrative records.

e. **User ID**

The user ID used for authentication if the calling application does not provide a userid and password explicitly. For example, if the calling application uses the method *createQueueConnection()* with no parameters.

If a value is specified for the User ID property, a value must also be specified for the Password property.The JMS client passes the userid and password to the JMS server.

f. **Password**

The password used, with the User ID property, for authentication if the calling application does not provide a userid and password explicitly.

g. **Queue manager**

The name of the WebSphere MQ Provider queue manager that we will be connecting to. Connections created by this factory connect to that queue manager.

Note: The queue manager is created and managed from WebSphere MQ, (with the WebSphere MQ Explorer wizard for example). For a full WebSphere MQ provider, you can have any number of queue managers for that provider, and with any name you want. For the integral JMS provider this was not possible. You only have one queue manager for the integral provider, which is automatically defined with name of the form WAS_<CellName>_<NodeName>.

h. **Host**

The name of the host on which the WebSphere MQ queue manager runs, for client connection only.

i. **Port**

The TCP/IP port number used for connection to the WebSphere MQ queue manager, for client connection only. This is the port used by the queue manager listener.

j. **Channel**

The name of the channel used for connection to the WebSphere MQ queue manager, for client connection only.

k. **Transport Type**

The transport type used for communication between the JMS application and the WebSphere MQ Provider queue manager. The possible values are **BINDINGS** (WebSphere MQ bindings) and **CLIENT** (TCP/IP). The default is BINDINGS.

BINDINGS requires the JMS application and the WebSphere MQ server be located on the same machine. CLIENT permits the queue manager to be on a different machine to the application.

If we select Transport Type: BINDINGS, the connection factory specifies connection properties to communicate with a local WebSphere MQ Provider queue manager.

Note: For detailed information on WebSphere MQ client and bind modes, see the *MQSeries System Administration* document, SC33-1873-02.

l. **Temp Model**

The name of the model queue definition that can be used by the queue manager to create temporary queues if a queue requested does not already exist.

m. **Client ID**

The JMS client identifier used for connections to the WebSphere MQ queue manager.

n. **CCSID**

The coded character set identifier used to connect to the WebSphere MQ provider queue manager. This code character set identifier must be one of the CCSIDs supported by WebSphere MQ.

Note: For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *MQSeries System Administration* document, SC33-1873-02.

o. **Msg Retention**

Select this tick box if you want unwanted messages to be left on the queue. Otherwise, unwanted messages will be dealt with according to their disposition options.

Other properties configurable under **Additional Properties** are **connection pool** settings and **session pool** settings.

Note: JMS Connection and session resources are managed by the J2C Connection Manager.

6. Click **OK**, when finished configuring the queue connection factory.
7. Click **Save**, to save the configuration.
8. To have the changed configuration take effect, stop then restart the application servers using the resource.

Configure topic connection factory for WebSphere MQ JMS Provider

A topic connection factory object is used to create JMS connections to the JMS Publish/Subscribe Provider for publish/subscribe messaging to topic destinations.

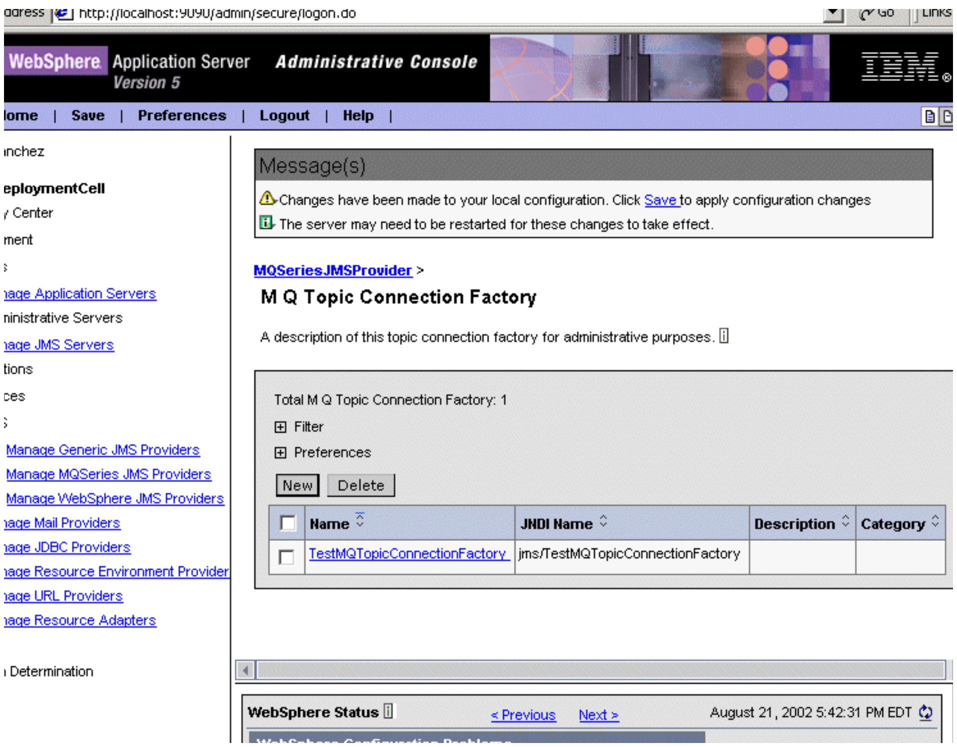
In Section 18.3.2, “Installing WebSphere MQ as the JMS provider” on page 631, we listed the Publish/Subscribe providers that you can use with WebSphere MQ, for Publish/Subscribe support. These were MQ Publish and Subscribe, WebSphere MQ Integrator, the WebSphere MQ Publish/Subscribe MA0C SupportPac or the WebSphere MQ Event Broker.

In this section we assume that one of these packages has been installed for Publish/Subscribe support.

To configure the properties of a topic connection factory object, which will be used to connect to the Publish/Subscribe Provider installed with WebSphere MQ, complete these steps:

1. In the navigation tree, expand **Resources** -> **JMS** -> **Configure WebSphere MQ JMS Providers**.
2. Select the **Scope** and click **Apply**. For a discussion on the *Scope* setting, see, “Configure queue connection factory for integral WebSphere JMS Provider” on page 643.
3. Click on **WebSphere MQ Topic Connection Factory** under **Additional Properties**.

This displays WebSphere MQ topic connection factory administered objects available to application servers under the selected scope, and which are used to connect to a Publish/Subscribe Provider installed with WebSphere MQ. This is shown in Figure 18-18:


ddress http://localhost:9090/admin/secure/logon.do
Go
Links
WebSphere Application Server Version 5
Administrative Console
IBM
lome | Save | Preferences | Logout | Help |
nchez
eploymentCell
y Center
ment
;
age Application Servers
ninistrative Servers
age JMS Servers
tions
ces
;
Manage Generic JMS Providers
Manage MQSeries JMS Providers
Manage WebSphere JMS Providers
age Mail Providers
age JDBC Providers
age Resource Environment Provider
age URL Providers
age Resource Adapters
Determination
Message(s)

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

The server may need to be restarted for these changes to take effect.

MQSeriesJMSProvider >

M Q Topic Connection Factory

A description of this topic connection factory for administrative purposes.

Total M Q Topic Connection Factory: 1

	Name	JNDI Name	Description	Category
☐	TestMQTopicConnectionFactory	jms/TestMQTopicConnectionFactory		

Figure 18-18 WebSphere MQ topic connection factory objects at the selected scope

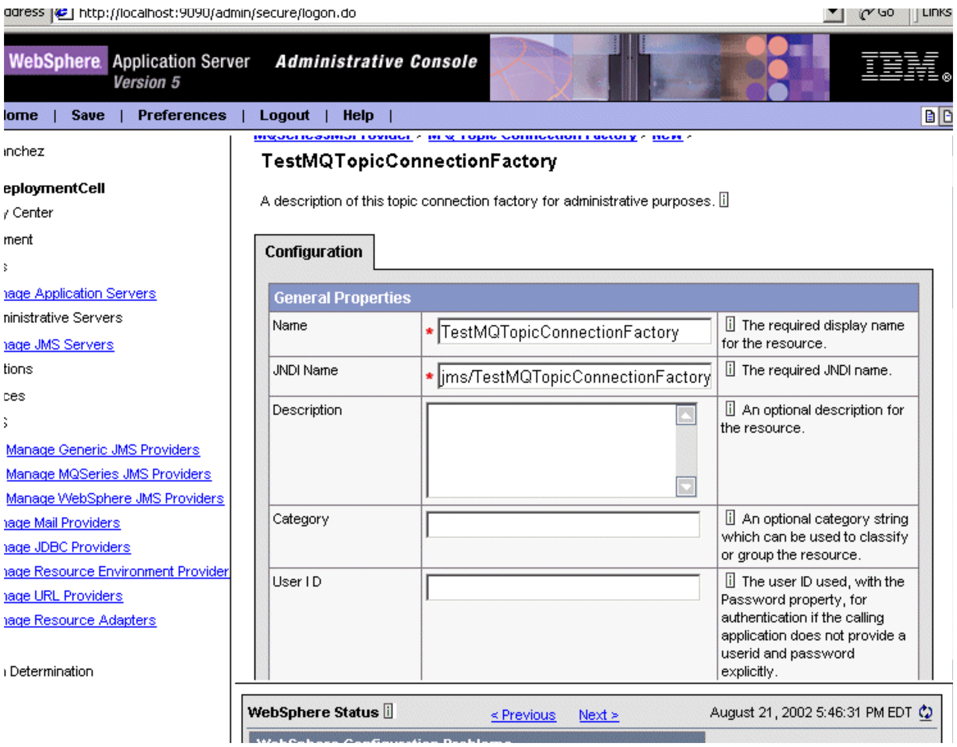
In this example we have one WebSphere MQ topic connection factory object, `TestMQTopicConnectionFactory`, available to all the application servers (and their applications) in my node.

This topic connection factory object is used to connect to a Publish/Subscribe provider installed with WebSphere MQ.

4. To create a new topic connection factory object, click **New** in the content pane, see Figure 18-18.

   To change the properties of an existing topic connection factory, click one of the connection factories displayed.

   Figure 18-19, shows the properties for the `TestMQTopicConnectionFactory` topic connection factory object:


WebSphere Application Server Version 5
Administrative Console
Home | Save | Preferences | Logout | Help |
Manage Application Servers
Manage JMS Servers
Manage Generic JMS Providers
Manage MQSeries JMS Providers
Manage WebSphere JMS Providers
Manage Mail Providers
Manage JDBC Providers
Manage Resource Environment Provider
Manage URL Providers
Manage Resource Adapters
Configuration
WebSphere Status
< Previous
Next >

TestMQTopicConnectionFactory

A description of this topic connection factory for administrative purposes.

General Properties

Name	* TestMQTopicConnectionFactory	The required display name for the resource.
JNDI Name	* jms/TestMQTopicConnectionFactory	The required JNDI name.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
User ID		The user ID used, with the Password property, for authentication if the calling application does not provide a userid and password explicitly.

August 21, 2002 5:46:31 PM EDT

Figure 18-19 WebSphere MQ topic connection factory properties

5. The configuration properties that need to be specified for a topic connection factory object that is to be used with a Publish/Subscribe Provider installed with WebSphere MQ are:

   a. **Name**

      The name by which this topic connection factory is known for administrative purposes.

   b. **JNDI name**

      The JNDI name used to bind the connection factory into the application server's name space. As a convention, use the form: `jms/Name`, where `Name` is the logical name of the resource.

      Figure 18-19, shows that the JNDI name for the `TestMQTopicConnectionFactory topic` connection factory object is `jms/TestMQTopicConnectionFactory`. If we dump the namespace for one of the application servers in the node, we will see an entry for the queue connection factory object as follows:

```
(top)/nodes/<node>/servers/<server>/jms/TestMQTopicConnectionFactory
```

c. **Description**

A description of this connection factory for administrative purposes.

d. **Category**

A category used to classify or group this connection factory, for your administrative records.

e. **User ID**

The user ID used for authentication if the calling application does not provide a userid and password explicitly. For example, if the calling application uses the method *createTopicConnection()* with no parameters.

If a value is specified for the User ID property, a value must also be specified for the Password property.

f. **Password**

The password used, with the User ID property, for authentication if the calling application does not provide a userid and password explicitly.

g. **Host**

The name of the host on which the WebSphere MQ queue manager that hosts the broker (Publish/Subscribe Provider) runs, for client connection only.

h. **Port**

The TCP/IP port number used for connection to the WebSphere MQ queue manager that hosts the broker, for client connection only. This is the port used by the queue manager listener.

i. **Transport Type**

The transport type used for communication between the JMS application and the WebSphere MQ Provider queue manager that hosts the broker. The possible values are **BINDINGS** (WebSphere MQ bindings) and **CLIENT** (TCP/IP). The default is BINDINGS.

BINDINGS requires the JMS application and the WebSphere MQ server to be located on the same machine. CLIENT permits the queue manager to be on a different machine to the application.

Note: For detailed information on WebSphere MQ client and bind modes, see the *MQSeries System Administration* document, SC33-1873-02.

j. **Channel**

The name of the channel used for connection to the WebSphere MQ queue manager that hosts the broker, for client connection only.

k. **Queue manager**

   The name of the WebSphere MQ Provider queue manager that hosts the broker. Connections created by this factory connect to that queue manager.

l. **Broker Control Queue**

   The name of the broker's control queue, to which all command messages (except **`Publish`** and **`Delete Publication`** command messages) are sent. Publisher and subscriber applications, and other brokers, send command messages to this queue.

   For example, a publisher that wishes to register with the broker, would send a **`Register Publisher`** command message to the broker's control queue.

m. **Broker Queue Manager**

   The name of the WebSphere MQ queue manager that provides the Publish/Subscribe message broker.

Author Comment: Is this a duplicate of the Queue Manager setting above?

n. **Broker Publication Queue**

   The name of the broker's input queue used to submit publications, for example SYSTEM.BROKER.DEFAULT.STREAM. The Broker can have multiple publication queues. You can set up a Topic Connection Factory for each one of them.

o. **Broker Subscription Queue**

   The name of the broker's queue from which non-durable subscription messages are retrieved. A subscriber can have an exclusive queue assigned to it from which it retrieves all its messages, or it can use a shared queue, such as SYSTEM.JMS.ND.SUBSCRIPTION.QUEUE, from which it, and other subscribers, retrieve their messages.

p. **Broker CC Subscription Queue**. The name of the broker's queue from which non-durable subscription messages are retrieved for a ConnectionConsumer.

Note: The above queues are WebSphere MQ queues managed by the broker queue manager.

q. **Broker Version**

   Whether the message broker is provided by the WebSphere MQ MA0C Supportpac or newer versions (MQ Integrator and MQ Publish and Subscribe).

r. **Temp Model**

   The name of the model queue definition that the broker can use to create dynamic queues to receive publications for streams other than the default stream. This is only used if the stream queue does not already exist. If this model queue definition does not exist, all stream queues must be defined by the administrator.

s. **Client ID**

   The JMS client identifier used for connections to the WebSphere MQ queue manager hosting the broker.

t. **CCSID**

   The coded character set identifier used to connect to the WebSphere MQ provider queue manager hosting the broker. This code character set identifier must be one of the CCSIDs supported by WebSphere MQ.

   Note: For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *MQSeries System Administration* document, SC33-1873-02.

u. **Clone support**

   Select this checkbox to enable WebSphere MQ clone support to allow the same durable subscription across topic clones.

Other properties configurable under **Additional Properties** are **connection pool** settings and **session pool** settings.

Note: JMS Connection and session resources are managed by the J2C Connection Manager.

6. Click `OK`, when you have finished configuring the topic connection factory object.
7. Click `Save`, to save the configuration.
8. To have the changed configuration take effect, stop then restart the application servers using the topic connection factory object.

Configure queue destination for WebSphere MQ JMS Provider

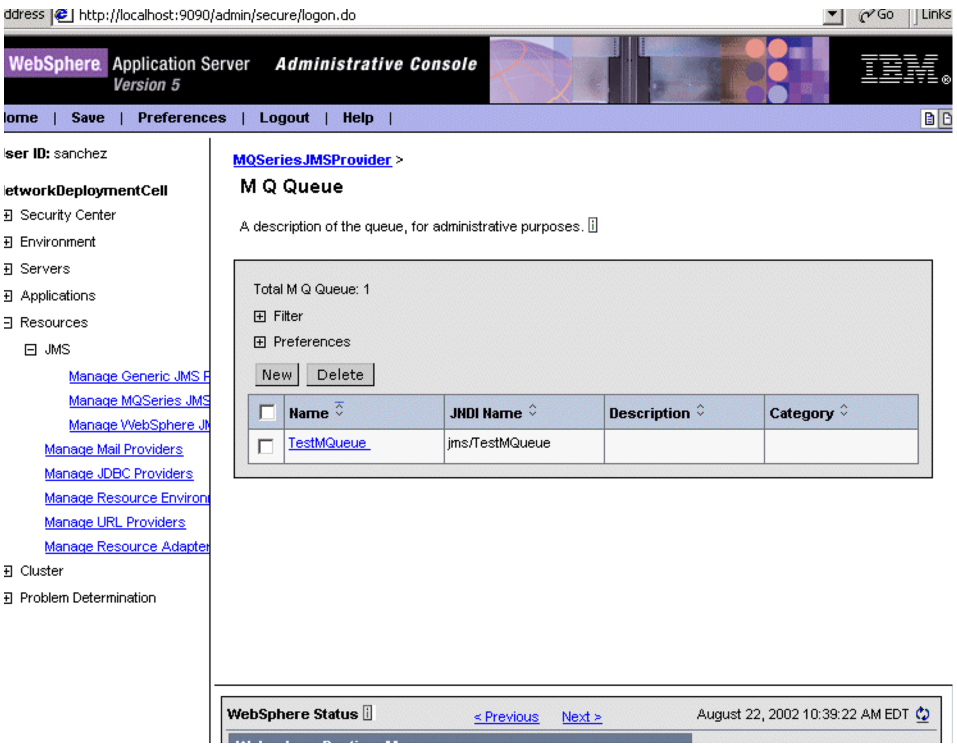
Complete the following steps to configure the properties of a queue destination object, that represents a queue hosted by the WebSphere MQ JMS provider:

1. In the navigation tree, expand **`Resources`** -> **`JMS`** -> **`Configure WebSphere MQ JMS Providers`**.
2. Select the **`Scope`** and click **`Apply`**.

   For a discussion on the *Scope* setting, see, "Configure queue connection factory for integral WebSphere JMS Provider" on page 643.

3. Click on **`WebSphere MQ Queue Destination`** under **`Additional Properties`**.

   This displays WebSphere MQ queue destination objects available to application servers under the selected scope. These objects represent messaging queues hosted by a WebSphere MQ JMS provider. This is shown in Figure 18-20:



A description of the queue, for administrative purposes.

Total M Q Queue: 1

	Name	JNDI Name	Description	Category
☐	TestMQueue	jms/TestMQueue		

Figure 18-20 WebSphere MQ queue destination administered objects at the selected scope

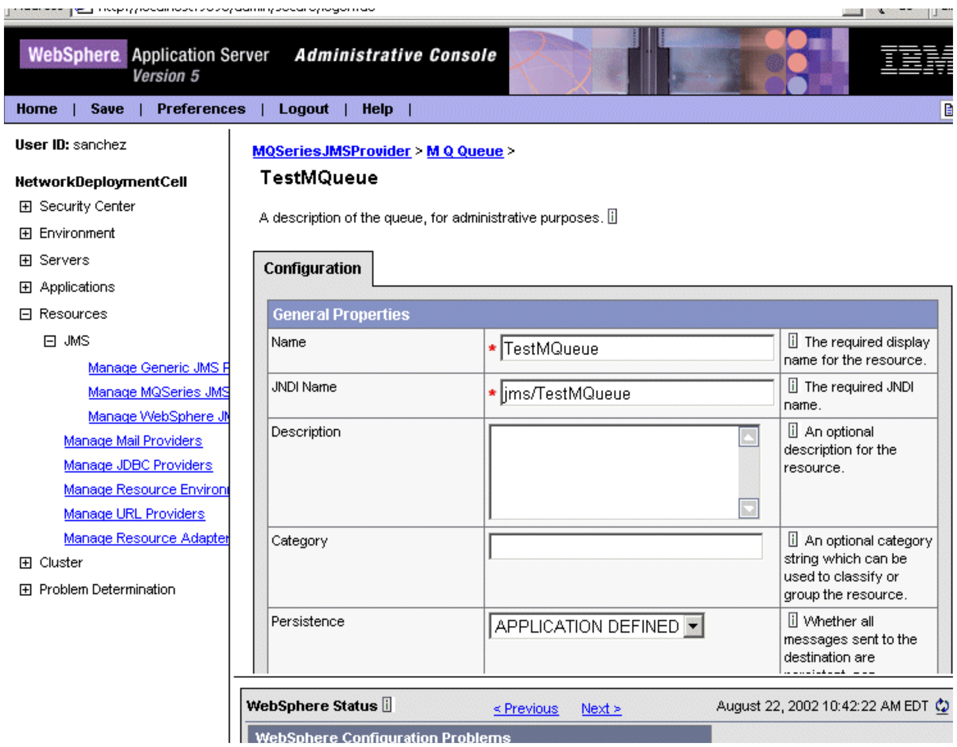
In this example we have one WebSphere MQ queue destination object, `TestMQueue`, available to all the application servers (and their applications) in the node.

This queue destination object represents a queue hosted by an WebSphere MQ JMS Provider.

4. To create a new queue destination object, click **`New`** in the content pane, see Figure 18-20.

   To change the properties of an existing queue destination object, click one of the queue destination objects displayed.

   Figure 18-21, shows the properties for the `TestMQueue` queue destination object:



WebSphere Application Server Version 5 Administrative Console

Home | Save | Preferences | Logout | Help |

User ID: sanchez

NetworkDeploymentCell
⊞ Security Center
⊞ Environment
⊞ Servers
⊞ Applications
⊟ Resources
⊟ JMS
Manage Generic JMS P
Manage MQSeries JMS
Manage WebSphere JM
Manage Mail Providers
Manage JDBC Providers
Manage Resource Environ
Manage URL Providers
Manage Resource Adapter
⊞ Cluster
⊞ Problem Determination

MQSeriesJMSProvider > M Q Queue >

TestMQueue

A description of the queue, for administrative purposes.

Configuration

General Properties		
Name	* TestMQueue	The required display name for the resource.
JNDI Name	* jms/TestMQueue	The required JNDI name.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Persistence	APPLICATION DEFINED	Whether all messages sent to the destination are

WebSphere Status < Previous Next > August 22, 2002 10:42:22 AM EDT

WebSphere Configuration Problems

Figure 18-21 WebSphere MQ queue destination object properties

5. The configuration properties that need to be specified for a queue connection object, that represents a queue hosted by a full WebSphere MQ JMS Provider are:

   a. Name

The name by which the queue is known for administrative purposes.

b. **JNDI name**

The JNDI name used to bind the queue into the application server's name space. As a convention, use the form: `jms/Name`, where `Name` is the logical name of the resource.

Figure 18-21, shows that the JNDI name for the `TestMQueue` queue destination object is `jms/TestMQueue`. If we dump the namespace for one of the application servers in the node, we will see an entry for the queue destination object as follows:

```
(top)/nodes/<node>/servers/<server>/jms/TestMQueue
```

c. **Description**

A description of the queue, for administrative purposes.

d. **Category**

A category used to classify or group this queue, for your administrative records.

e. **Persistence**

Specifies the persistence of messages sent to this destination. The possible values are:

- PERSISTENT
- NON PERSISTENT
- APPLICATION DEFINED
- QUEUE DEFINED

The default is APPLICATION DEFINED. It is the application that puts the message onto the queue that instructs the WebSphere MQ JMS Provider whether to store the message, in case the provider fails.

For QUEUE DEFINED persistence, messages on the destination queue have their persistence defined by the WebSphere MQ queue definition properties.

Note: For the full WebSphere MQ JMS Provider, messaging queues get created in WebSphere MQ under a queue manager. WebSphere MQ allows you to browse messages on a queue, send a test message, clear messages on a queue, etc.

For the integral JMS Provider, messaging queues were created from the definition to a JMS Server process. There is no provided mechanism for sending test messages or clearing existing queues.

f. **Priority**

Specifies the message priority for this destination queue. The possible values are:

- APPLICATION DEFINED
- QUEUE DEFINED
- SPECIFIED

The default is APPLICATION DEFINED. It is the application that puts the message onto the queue that defines the message priority.

For QUEUE DEFINED priority, messages on the destination queue have their priority defined by the WebSphere MQ queue definition properties.

g. **Specified Priority**

If the **Priority** is set to SPECIFIED, this field determines the message priority for this queue. The ten levels of priority range from 0 (lowest) to 9 (highest). Messages sent to this queue have the priority value specified by this property.

h. **Expiry**

Specifies the expiry timeout for this queue. Possible values are:

- APPLICATION DEFINED
- UNLIMITED
- SPECIFIED

The default is APPLICATION DEFINED. It is the application that puts the message onto the queue that defines whether the message is destroyed after a period of time.

i. **Specified Expiry**

If the Expiry property is set to SPECIFIED, type here the number of milliseconds (greater than 0) after which messages on this queue expire. Zero indicates that messages never timeout.

j. **Base Queue Name**

The name of the actual WebSphere MQ messaging queue to which messages are sent.

Note: The Base Queue Name is the actual messaging queue created in WebSphere MQ under a queue manager. For the integral JMS Server, messaging queues got automatically created for you when you define them to the JMS Server . With full WebSphere MQ you need to create these messaging queues yourself.

WebSphere MQ allows you to manage queues using tools such as the WebSphere MQ Explorer wizard. This is not possible with the integral JMS Provider.

k. **Base Queue Manager Name**

   The name of the WebSphere MQ queue manager hosting the queue specified by the **Base Queue Name** property.

l. **CCSID**

   The coded character set identifier used to connect to the WebSphere MQ provider queue manager. This code character set identifier must be one of the CCSIDs supported by WebSphere MQ.

   Note: For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *MQSeries System Administration* document, SC33-1873-02.

m. **Use Native Encoding**

   Select this checkbox to indicate that the queue destination is to use native encoding to represent numeric values in the message data. If native encoding is not enabled (the default), you need to specify the properties for **Integer**, **Decimal** and **Floating Point** encoding.

   Note: For information about encoding properties, see the *MQSeries Using Java* document, SC34-5456-08.

n. **Integer Encoding**

   If native encoding is not enabled, select whether the encoding defined for binary integers is Normal or Reversed. The default value is Normal.

o. **Decimal Encoding**

   If native encoding is not enabled, select whether the encoding defined for decimal integers is Normal or Reversed. The default value is Normal.

p. **Floating Point Encoding**. If native encoding is not enabled, select whether the encoding defined for floating-point numbers is IEEENormal, IEEEReversed or S390. The default value is IEEE_Normal.

> **Note:** For information about encoding properties, see the *MQSeries Using Java* document, SC34-5456-08.

q. **Target Client**

Whether the receiving application is a JMS compliant application (JMS) or not (WebSphere MQ).

r. **Queue Manager Host**

The hostname of the machine where the queue manager for this queue destination resides.

s. **Queue Manager Port**

The port number used by the queue manager hosting this queue.

t. **Server Connection Channel Name**

The name of the channel used for connection to the WebSphere MQ queue manager hosting this queue.

u. **User Name**

The user ID, used for authentication when connecting to the queue manager hosting the queue destination.

If a value is specified for the **User Name** property, a value must also be specified for the **Password** property.

v. **Password**

The password, which together with the User Name property, is used for authentication when connecting to the queue manager hosting the queue destination.

> **Note:** Properties such as Queue Manager Host, Queue Manager Port, Server Connection Channel, User Name and Password are not mandatory. They are included for identification purposes, so it is easy for the administrator to identify where the messaging queue is hosted.

Author Comment: Suggestion - need to indicate which fields are mandatory and which aren't.

6. Click **OK**, when finished configuring the queue destination object.
7. Click **Save**, to save the configuration.
8. To have the changed configuration take effect, stop then restart the application servers using the resource.

Configur e topic destination for WebSphere MQ JMS Provider

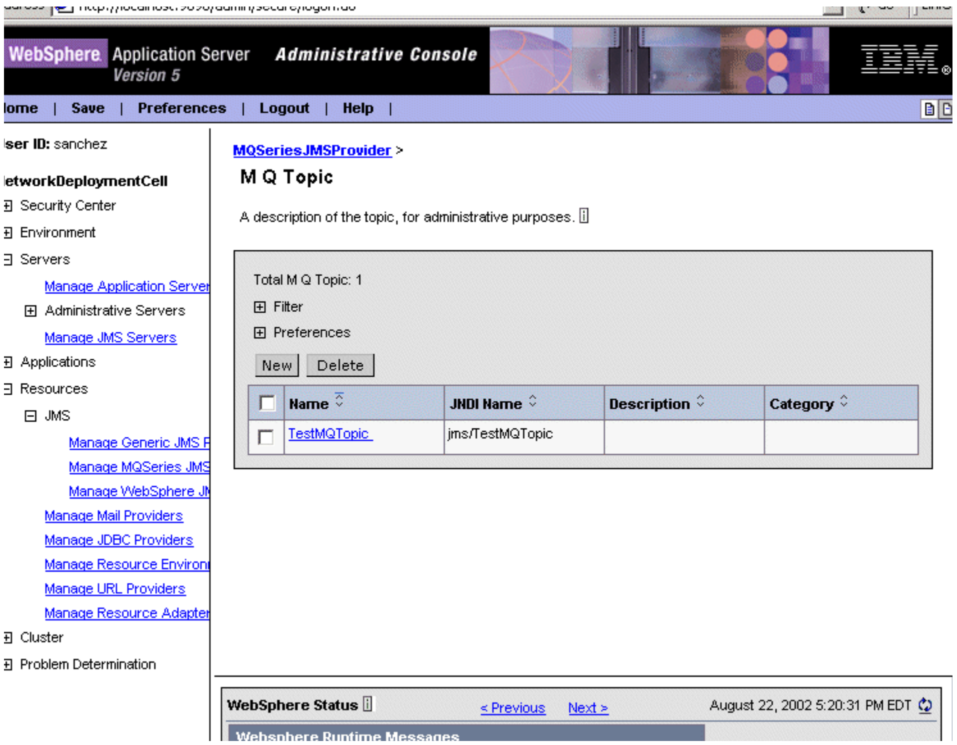
The Publish/Subscribe providers supported by WebSphere MQ are:

- MQ Publish and Subscribe
- WebSphere MQ Integrator
- WebSphere MQ Publish/Subscribe MA0C SupportPac
- WebSphere MQ Event Broker

Complete the following steps to configure the properties of a topic destination administrative object, which represents a topic served by the Publish/Subscribe JMS Provider installed with WebSphere MQ.

1. In the navigation tree, expand **Resources** -> **JMS** -> **Configure WebSphere MQ JMS Providers**.
2. Select the **Scope** and click **Apply**. For a discussion on the *Scope* setting, see, "Configure queue connection factory for integral WebSphere JMS Provider" on page 643.
3. Click on **WebSphere MQ Topic Destination** under **Additional Properties**.

   This displays WebSphere MQ topic destination objects available to application servers under the selected scope. These objects represent topics served by a Publish/Subscribe JMS provider installed with WebSphere MQ. This is shown in Figure 18-22:



A description of the topic, for administrative purposes.

Total M Q Topic: 1

	Name	JNDI Name	Description	Category
☐	TestMQTopic	jms/TestMQTopic		

Figure 18-22 WebSphere MQ topic destination administered objects at the selected scope

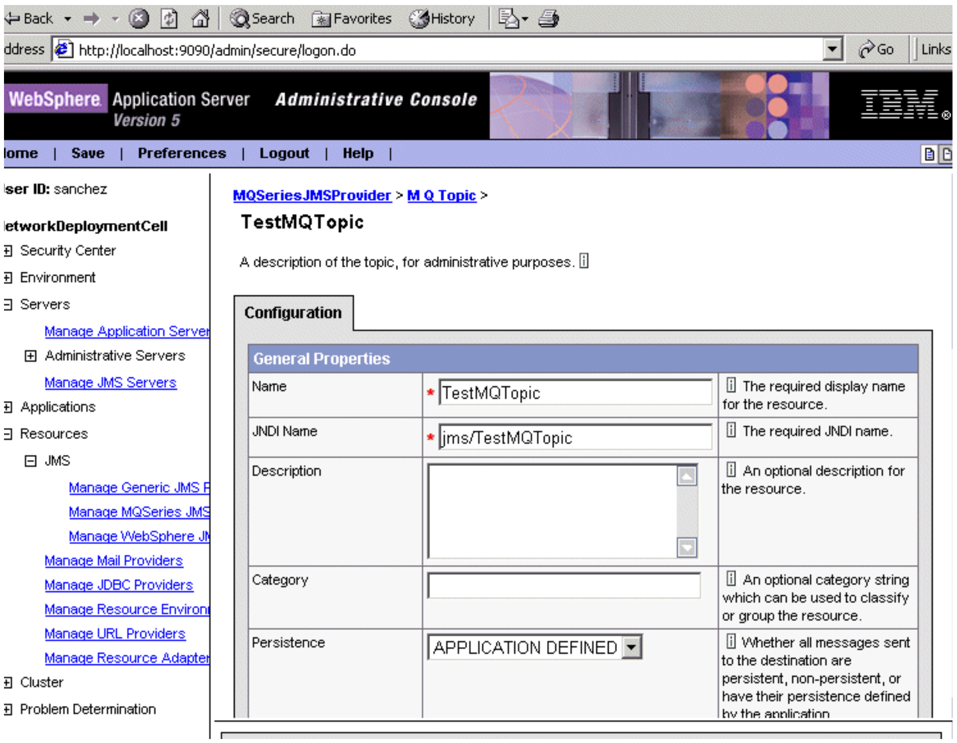
In this example we have one WebSphere MQ topic destination object, `TestMQTopic`, available to all the application servers (and their applications) in the node.

This topic destination object represent a topic served by the Publish/Subscribe Provider installed with WebSphere MQ.

4. To create a new topic destination object, click **`New`** in the content pane, see Figure 18-22.

   To change the properties of an existing topic destination object, click one of the topic destination objects displayed.

   Figure 18-23, shows the properties for the `TestMQTopic` topic destination object:



TestMQTopic

A description of the topic, for administrative purposes.

General Properties

Name	* TestMQTopic	The required display name for the resource.
JNDI Name	* jms/TestMQTopic	The required JNDI name.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Persistence	APPLICATION DEFINED	Whether all messages sent to the destination are persistent, non-persistent, or have their persistence defined by the application

Figure 18-23 WebSphere MQ topic destination object properties

5. The configuration properties that need to be specified for a topic destination object, that represents a topic served by the Publish/Subscribe provider installed with WebSphere MQ are:

   a. **Name**

      The name by which the topic is known for administrative purposes.

   b. **JNDI name**

      The JNDI name used to bind the topic destination into the application server’s name space. As a convention, use the form: `jms/Name`, where `Name` is the logical name of the resource.

      Figure 18-23, shows that the JNDI name for the `TestMQTopic` topic destination object is `jms/TestMQTopic`. If we dump the namespace for one of the application servers in the node, we will see an entry for the topic destination object as follows:

      ```
      (top)/nodes/<node>/servers/<server>/jms/TestMQTopic
      ```

   c. **Description**

A description of the topic, for administrative purposes.

d. **Category**

A category used to classify or group this topic, for your administrative records.

e. **Persistence**

Specifies the persistence of messages sent to this topic destination. The possible values are:

- PERSISTENT
- NON PERSISTENT
- APPLICATION DEFINED
- QUEUE DEFINED

The default is APPLICATION DEFINED. It is the application that puts the message onto the topic destination that instructs the Publish/Subscribe Provider installed with WebSphere MQ whether to store the message, in case the provider or the queue manager fails.

For QUEUE DEFINED persistence, messages on the topic destination queue have their persistence defined by the WebSphere MQ queue definition properties.

f. **Priority**

Specifies the message priority for this topic destination. The possible values are:

- APPLICATION DEFINED
- QUEUE DEFINED
- SPECIFIED

The default is APPLICATION DEFINED. It is the application that puts the message onto the topic destination that defines the message priority.

For QUEUE DEFINED priority, messages on the topic destination queue have their priority defined by the WebSphere MQ queue definition properties.

g. **Specified Priority**

If the **Priority** is set to SPECIFIED, type here the message priority for this topic. The ten levels of priority range from 0 (lowest) to 9 (highest). Messages sent to this topic have the priority value specified by this property.

h. **Expiry**

Specifies whether the expiry timeout for this topic destination is defined by the application, APPLICATION DEFINED, whether we specify the expiration time, SPECIFIED, or whether the message on the topic destination never expires, UNLIMITED. The default is APPLICATION DEFINED. It is the application that puts the message onto the topic destination that defines whether the message is destroyed after a period of time.

i. **Specified Expiry**

If the Expiry property is set to SPECIFIED, this field determines the number of milliseconds (greater than 0) after which messages on this topic destination expire. Zero indicates that messages never timeout.

j. **Base Topic Name**

The name of the Publish/Subscribe Provider topic to which messages are sent.

k. **CCSID**

The coded character set identifier used to connect to the queue manager hosting the Publish/Subscribe provider. This code character set identifier must be one of the CCSIDs supported by WebSphere MQ.

Note: For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *MQSeries System Administration* document, SC33-1873-02.

l. **Use Native Encoding**

Select this checkbox to indicate that the topic destination should use native encoding to represent numeric values in the message data. If native encoding is not enabled, (the default), you need to specify the properties for **Integer**, **Decimal** and **Floating Point** encoding.

Note: For information about encoding properties, see the *MQSeries Using Java* document, SC34-5456-08.

m. **Integer Encoding**

If native encoding is not enabled, select whether the encoding defined for binary integers is Normal or Reversed. The default value is Normal.

n. **Decimal Encoding**

If native encoding is not enabled, select whether the encoding defined for decimal integers is Normal or Reversed. The default value is Normal.

o. **Floating Point Encoding**

   If native encoding is not enabled, select whether the encoding defined for floating-point numbers is IEEENormal, IEEEReversed or S390. The default value is IEEE_Normal.

   Note: For information about encoding properties, see the *MQSeries Using Java* document, SC34-5456-08.

p. **Target Client**

   Whether the receiving application is a JMS compliant application (JMS) or not (WebSphere MQ).

q. **Broker Durable Subscription Queue**

   The name of the broker’s queue from which durable subscription messages are retrieved.

r. **Broker CC Durable Subscription Queue**

   The name of the broker’s queue from which durable subscription messages are retrieved for a ConnectionConsumer.

Note: The above queues are WebSphere MQ queues managed by the broker queue manager.

6. Click **OK**, when finished configuring the topic destination object.
7. Click **Save**, to save the configuration.
8. To have the changed configuration take effect, stop then restart the application servers using the resource.

18.5.3 Configuring resources for a Generic JMS Provider

If you use a JMS provider other than the integral WebSphere JMS provider or an WebSphere MQ JMS provider, you can still use the administrative console to register JMS destinations and JMS connection factories into the WebSphere namespace.

- Configure queue connection factory.
- Configure topic connection factory.
- Configure queue destination.
- Configure topic destination.

Note: For a detailed description of how WebSphere 5.0 supports generic JMS providers, see Chapter 7, “Asynchronous messaging” on page 147.

Configure JMS connection factory for a Generic JMS Provider

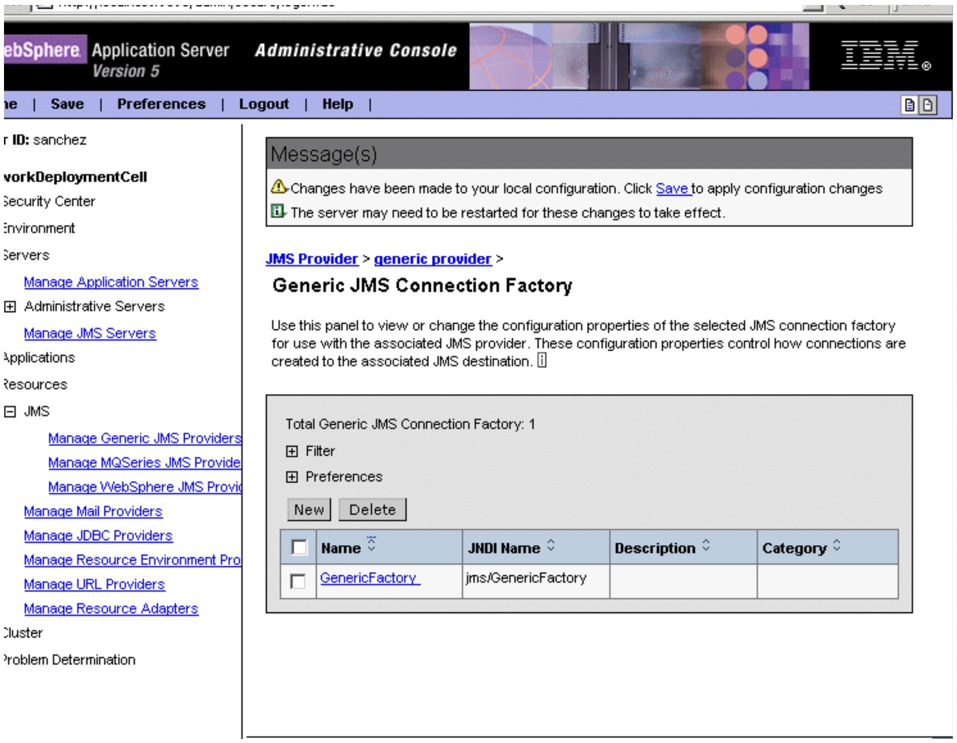
To configure properties for a JMS connection factory object, that will be used to connect to a JMS Provider other than the integral WebSphere JMS provider or an WebSphere MQ JMS provider, complete these steps:

1. In the navigation tree, expand **`Resources`** -> **`JMS`** -> **`Manage Generic JMS Providers`**.
2. Select the **`Scope`** and click **`Apply`**.

   For a discussion on the *Scope* setting, see, “Configure queue connection factory for integral WebSphere JMS Provider” on page 643.
3. Select the Generic JMS provider for which you want to configure the connection factory.

   This displays a table of properties for the JMS provider and links to the types of JMS resources supported.
4. Click on **`JMS Connection Factories`** under **`Additional Properties`**.

   This displays the Generic provider connection factory administered objects, available to application servers under the selected scope, and which are used to connect to the Generic JMS provider. This is shown in Figure 18-24:



Message(s)

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

The server may need to be restarted for these changes to take effect.

JMS Provider > generic provider >

Generic JMS Connection Factory

Use this panel to view or change the configuration properties of the selected JMS connection factory for use with the associated JMS provider. These configuration properties control how connections are created to the associated JMS destination.

Total Generic JMS Connection Factory: 1

Filter

Preferences

New Delete

	Name	JNDI Name	Description	Category
☐	GenericFactory	jms/GenericFactory		

Figure 18-24 Generic connection factory administered objects at the selected scope

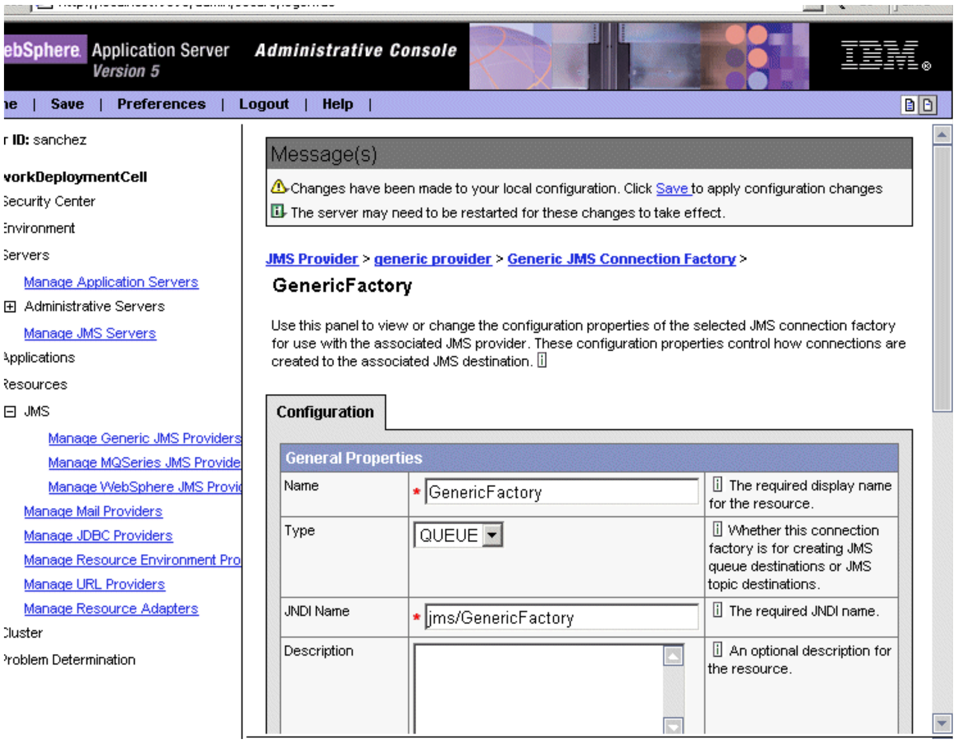
In this example we have one Generic JMS connection factory object, `GenericFactory` available to all the application servers (and their applications) in my node.

This connection factory object has all the necessary properties to connect to the Generic JMS provider.

5. To create a new connection factory object, click **`New`** in the content pane, see Figure 18-22.

   To change the properties of an existing connection factory, click one of the connection factories displayed.

   Figure 18-25, partially shows the properties for the `GenericFactory` connection factory object used to connect to the selected Generic JMS provider:



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

The server may need to be restarted for these changes to take effect.

JMS Provider > generic provider > Generic JMS Connection Factory >

GenericFactory

Use this panel to view or change the configuration properties of the selected JMS connection factory for use with the associated JMS provider. These configuration properties control how connections are created to the associated JMS destination.

Name	* GenericFactory	The required display name for the resource.
Type	QUEUE	Whether this connection factory is for creating JMS queue destinations or JMS topic destinations.
JNDI Name	* jms/GenericFactory	The required JNDI name.
Description		An optional description for the resource.

Figure 18-25 Generic connection factory properties

6. The configuration properties that need to be specified for a connection factory object that is to connect to a Generic JMS Provider are:

   a. **Name**

      The name by which this connection factory is known for administrative purposes.

   b. **Type**

      Whether the connection factory is used for point-to-point messaging, **QUEUE**, or for publish/subscribe messaging, **TOPIC**.

   c. **JNDI name**

      The JNDI name used to bind the connection factory into the application server's name space. As a convention, use the form: `jms/Name`, where `Name` is the logical name of the resource.

      Figure 18-25, shows that the JNDI name for the `GenericFactory` connection factory object is `jms/GenericFactory`. If we dump the namespace for one of the application servers in the node, we will see an entry for the connection factory object as follows:

```
(top)/nodes/<node>/servers/<server>/jms/GenericFactory
```

d. **Description**

   A description of this connection factory for administrative purposes.

e. **Category**

   A category used to classify or group this connection factory, for your administrative records.

f. **External JNDI Name**

   The connection factory JNDI name, as registered in the Generic Provider's name space.

Note: The connection factory object that holds connection properties with the Generic JMS Provider is registered with the Generic Provider namespace.

With the administrative console, what we are creating is a local representation of it, i.e a connection factory object that is registered in the application server's namespace and which is bound to the remote object.

The External Initial Context Factory and External Provider URL were provided as part of the configuration of the provider, see Section 18.4.3, "Managing a Generic JMS Provider" on page 641.

For the resource lookup to work, the Generic Provider's InitialContextFactory and JMS client implementation classes, will need to be included in the application server's classpath.

For further details, see Chapter 7, "Asynchronous messaging" on page 147.

g. **User ID**. The user ID used for authentication if the calling application does not provide a userid and password explicitly. For example, if the calling application uses the method *createQueueConnection()* with no parameters.

   If a value is specified for the User ID property, a value must also be specified for the Password property.The JMS client passes the userid and password to the JMS server.

h. **Password**

   The password used, with the User ID property, for authentication if the calling application does not provide a userid and password explicitly.

Other properties configurable under **Additional Properties** are **connection pool** settings and **session pool** settings.

Note: JMS Connection and session resources are managed by the J2C Connection Manager.

7. Click **OK**, when finished configuring the queue connection factory.
8. Click **Save**, to save the configuration.
9. To have the changed configuration take effect, stop then restart the application servers using the resource.

Configuring a JMS destination for a Generic JMS Provider

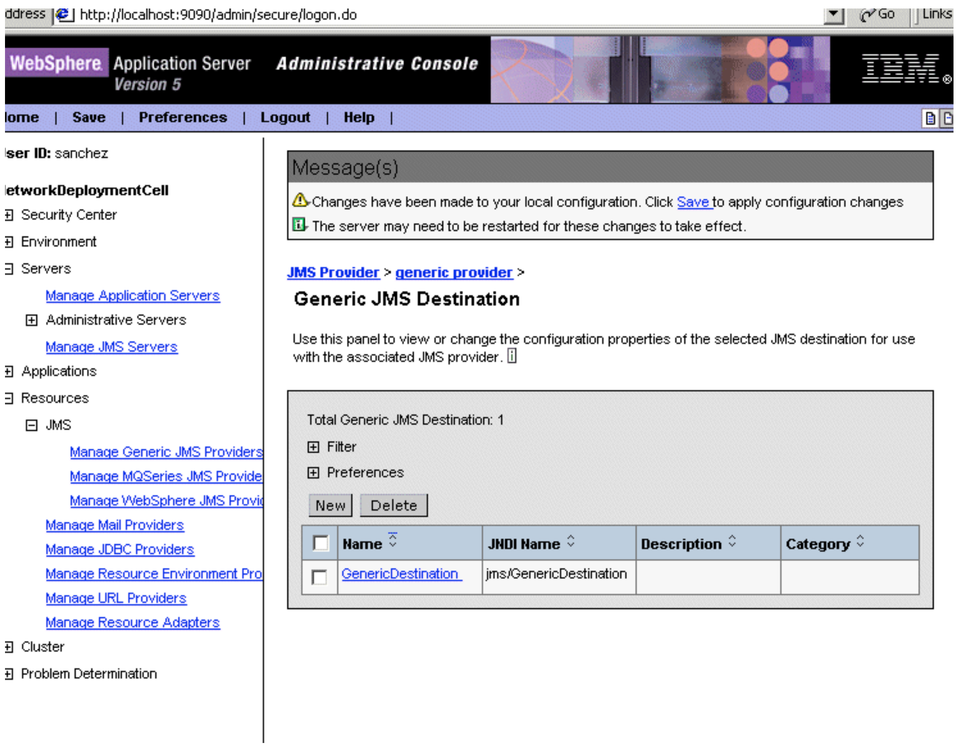
Complete the following steps to configure the properties of a JMS destination object, that represents a resource hosted by Generic JMS provider:

1. In the navigation tree, expand **Resources** -> **JMS** -> **Manage Generic JMS Providers**.
2. Select the **Scope** and click **Apply.** For a discussion on the *Scope* setting see, “Configure queue connection factory for integral WebSphere JMS Provider” on page 643.
3. Select the Generic JMS provider for which you want to configure the destination object.

   This displays a table of properties for the JMS provider and links to the types of JMS resources supported.

4. Click on **JMS Destinations** under **Additional Properties**.

   This displays JMS destinations available to application servers under the selected scope. These objects represent destinations hosted by the Generic JMS provider. This is shown in Figure 18-26:



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

The server may need to be restarted for these changes to take effect.

JMS Provider > generic provider >

Generic JMS Destination

Use this panel to view or change the configuration properties of the selected JMS destination for use with the associated JMS provider.

Total Generic JMS Destination: 1

	Name	JNDI Name	Description	Category
☐	GenericDestination	jms/GenericDestination		

Figure 18-26 Generic destination administered objects at the selected scope

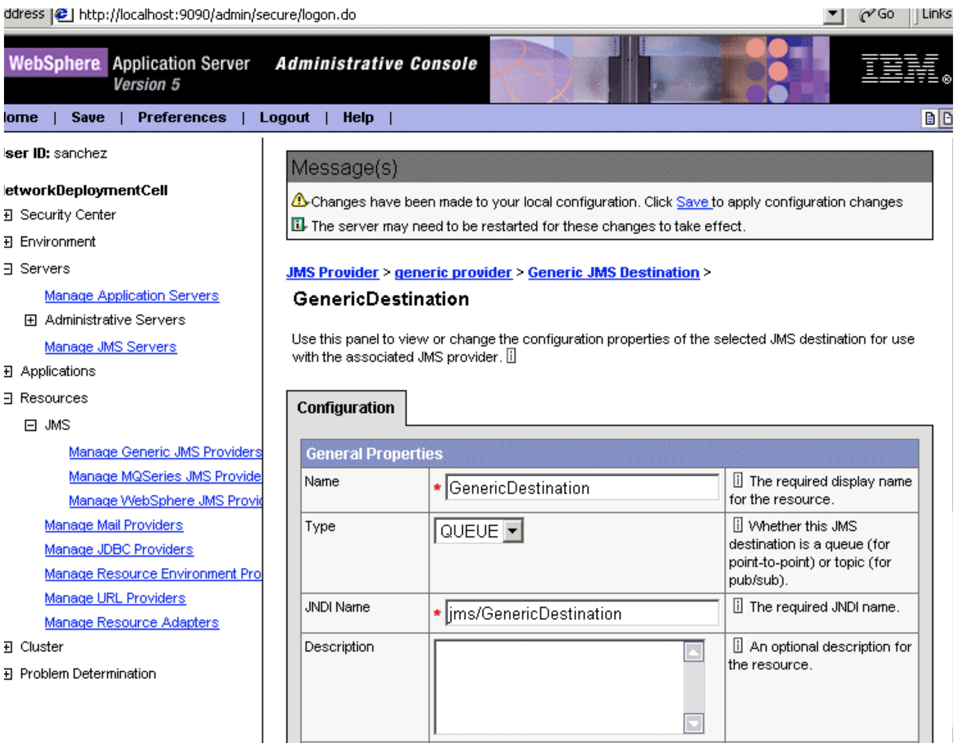
In this example we have one Generic destination object, `GenericDestination`, available to all the application servers (and their applications) in the node.

This destination object represents a destination resource hosted by the Generic JMS Provider.

5. To create a new destination object, click **`New`** in the content pane, see Figure 18-26.

   To change the properties of an existing destination object, click one of the destination objects displayed.

   Figure 18-27, shows the properties for the `GenericDestination` destination object:


ddress http://localhost:9090/admin/secure/logon.do Go Links
WebSphere Application Server Version 5
Administrative Console
IBM
lome | Save | Preferences | Logout | Help |
ser ID: sanchez
etworkDeploymentCell
Security Center
Environment
Servers
Manage Application Servers
Administrative Servers
Manage JMS Servers
Applications
Resources
JMS
Manage Generic JMS Providers
Manage MQSeries JMS Provide
Manage WebSphere JMS Provid
Manage Mail Providers
Manage JDBC Providers
Manage Resource Environment Pro
Manage URL Providers
Manage Resource Adapters
Cluster
Problem Determination

Message(s)

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

The server may need to be restarted for these changes to take effect.

JMS Provider > generic provider > Generic JMS Destination >

GenericDestination

Use this panel to view or change the configuration properties of the selected JMS destination for use with the associated JMS provider.

Configuration

General Properties

Name	* GenericDestination	The required display name for the resource.
Type	QUEUE	Whether this JMS destination is a queue (for point-to-point) or topic (for pub/sub).
JNDI Name	* jms/GenericDestination	The required JNDI name.
Description		An optional description for the resource.

Figure 18-27 Generic destination object properties

6. The configuration properties that need to be specified for a destination object, that represents a resource hosted by a Generic JMS Provider are:

   a. **Name**

      The name by which the destination object is known for administrative purposes.

   b. **Type**

      Whether this JMS destination object represents a messaging **QUEUE** for point-to-point messaging or a **TOPIC** for Publish/Subscribe messaging.

   c. **JNDI name**

      The JNDI name used to bind the destination into the application server's name space. As a convention, use the form: `jms/Name`, where `Name` is the logical name of the resource.

      Figure 18-27, shows that the JNDI name for the `GenericDestination` destination object is `jms/GenericDestination`. If we dump the namespace for one of the application servers in the node, we will see an entry for the queue destination object as follows:

```
(top)/nodes/<node>/servers/<server>/jms/GenericDestination
```

d. **Description**

A description of the destination, for administrative purposes.

e. **Category**

A category used to classify or group this destination, for your administrative records.

f. **External JNDI Name**

The destination JNDI name, as registered in the Generic Provider's name space.

Note: The destination object (queue or topic) that holds properties for the JMS Provider queue or topic messaging resource, is registered with the Generic Provider namespace.

With the administrative console, what we are creating is a local representation of it, i.e a destination that is registered in the application server's namespace and which is bound to the remote object.

For further details, see Chapter 7, "Asynchronous messaging" on page 147.

7. Click **OK**, when finished configuring the queue destination object.
8. Click **Save**, to save the configuration.

To have the changed configuration take effect, stop then restart the application servers using the resource.

18.5.4 JMS provider sample

See Chapter 7, "Asynchronous messaging" on page 147 for an example of how to access and use a JMS provider from Java code.

18.5.5 More information

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

- WebSphere InfoCenter, "Implementing WebSphere J2EE applications that use JMS" and "Administering WebSphere JMS support.

`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`

18.6 JDBC and JDBC Resources

The JDBC API provides a Java application a programming interface for data access of relational databases from the Java programming language.

The JDBC 2.0 API is comprised of two packages:

- The java.sql package, which is the JDBC 2.0 core API.
- The javax.sql package, which is the JDBC 2.0 Standard Extension API. This package provides *DataSource* and *Connection pooling* functionality.

In the next sections we will explain how to create and configure DataSource objects for use by JDBC applications. This is the recommended way of getting a connection to a database, and the only way if you are looking to use connection pooling and or distributed transactions.

18.6.1 What are JDBC providers and datasources?

A *DataSource* object represents a real world data source, such as a relational database. When a DataSource object has been registered with a JNDI naming service, an application can retrieve it from the naming service and use it to make a connection to the data source it represents.

Information about the data source and how to locate it, such as its name, the server on which it resides, its port number, and so on, is stored in the form of properties on the DataSource object. This makes an application more portable because it does not need to hard code a driver name, which often includes the name of a particular vendor. It also makes maintaining the code easier because if, for example, the data source is moved to a different server, all that needs to be done is to update the relevant property in the DataSource object. None of the code using that data source needs to be touched.

Once a DataSource object has been registered with an application server's JNDI name space, application programmers can use it to make a connection to the data source it represents.

The connection will usually be a pooled connection, i.e once the application closes the connection, the connection is returned to a connection pool, rather than being destroyed.

DataSource classes and JDBC drivers are implemented by the data source vendor. By configuring a JDBC Provider, what we are doing is providing information on the set of classes used to implement the DataSource object and the database driver, among others, i.e it provides the environment settings for the DataSource object.

Note: A driver may be written purely in the Java programming language or in a mixture of the Java programming language and the Java Native Interface (JNI) native methods.

In the next sections we will describe how to create and configure DataSource objects, as well as how to configure the connection pools used to serve connections from the DataSource.

18.6.2 WebSphere support for datasources

The programming model is as follows:

1. An application retrieves a *DataSource object* from the JNDI naming space.
2. After the DataSource object is obtained, the application code calls *getConnection()* on the datasource to get a *Connection* object. The connection is a pooled connection, ie. it is obtained from a pool of connections.
3. Once the connection is acquired, the application then sends SQL queries or updates to the database.

WebSphere Version 5 provides two types of datasources differentiated by how the connections are handled:

- WAS4 datasource
- WAS5 datasource

The two types of datasource are shown in the following sections:

WAS4 datasource

WebSphere Version 4.0, provides its own *JDBC connection manager* to handle connection pooling and JDBC access. This is shown in Figure 18-28. If an application chooses to use a WAS4 datasource, the application will have the same connection behavior as in WebSphere Version 4.

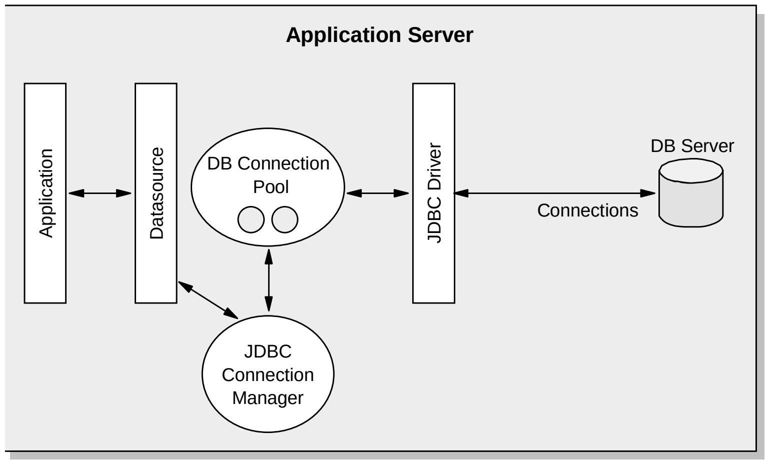


Figure 18-28 Connection pooling in WebSphere Version 4

WebSphere 5 datasource

In WebSphere Version 5, connection pooling is provided by two parts, a *JCA connection manager*, and a *relational resource adapter*, see Figure 18-29.

Author Comment: Picture may need to be changed. Too specific to CMP 2.0 EJBs -use of connection factory.

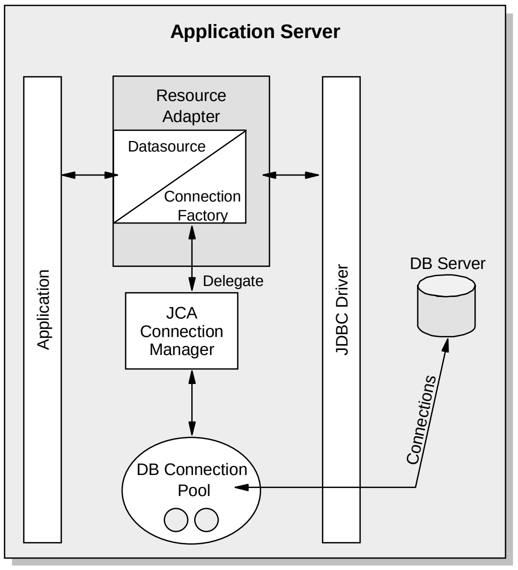


Figure 18-29 Resource adapter in J2EE Connector Architecture

The JCA connection manager provides the connection pooling, the local transaction and security supports. The relational resource adapter provides both JDBC wrappers and JCA CCI implementation that allows BMP, JDBC applications and CMP beans to access the database.Figure 18-30 shows the relational resource adapter model.

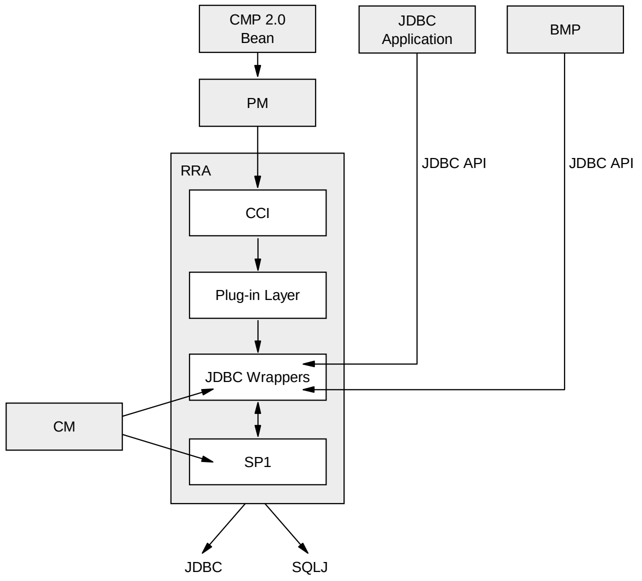


Figure 18-30 Persistence Resource adapter model (PM: Persistence Manager, CM: Connection Manager)

Note: WebSphere provides a Persistence Resource Adapter to provide relational persistence services to the EJB beans that are deployed in the J2EE 1.3 application as well as providing database access to BMP and JDBC applications. The Persistence Resource Adapter is made of two components: the persistence manager, which is present to support the new EJB 2.0 CMP persistence model, and the relational resource adapter.

The persistence resource adapter code is packaged into the following Java packages:

- com.ibm.ws.rsadapter.cci - contains CCI implementation and JDBC wrappers.
- com.ibm.ws.rsadapter.spi - contains SPI implementation.
- com.ibm.ws.rsadapter.jdbc - contains all the JDBC wrappers.
- com.ibm.websphere.rsadapter - DataStoreHelper, WSCallerHelper and DataAccessFunctionSet.

The Relational Resource adapter is the EJB 2.0 Persistence Manager's persistent vehicle to handle the data access to/from the backend store. It provides relational persistence services to the EJB beans that are deployed in the J2EE 1.3 applications. The Relational Resource Adapter implementation is based on the J2EE Connector (JCA) specification and implements the JCA CCI and SPI interfaces.

The relational resource adapter that is available with Network Deployment only provides JDBC data access to a relational database, but potentially the relational resource adapter can be used for other forms of data access. For example, SQLJ access is provided with the WebSphere 5.0 Extended Deployment edition for example.

The following database platforms are supported for JDBC: DB2 family, Oracle, Sybase, Informix, SQLServer, Cloudscape and third party vendor JDBC data source using SQL99 standards.

If an application chooses to use a WAS5 datasource, the datasource will be using the JCA connector architecture to get to the relational database. Although there is no difference between the existing WebSphere JDBC support versus the new support in terms of application development, there will be some connection behavior changes because of different architectures.

For an EJB the sequence is as follows:

1. An EJB performs a JNDI lookup of a *datasource* connection factory and issues a *getConnection()* request.
2. The connection factory delegates the request to a connection manager.
3. The connection manager looks for an instance of a connection pool in the application server. If no connection pool is available, then the manager uses the *ManagedConnectionFactory* to create a physical (nonpooled) connection.

So from a JDBC application point of view there is no difference between using a *WAS 4 datasource* or a *WAS 5 datasource*. It is the implementation of the datasource that changes.

For more details, refer to the J2EE Connector Architecture specification, `http://java.sun.com/j2ee/connector/`

18.6.3 Datasource usage guidelines

The datasource that can be used is *dictated* by whether the application is a J2EE 1.2, or a J2EE 1.3 application, and whether it uses EJB 1.1 modules or EJB 2.0 Modules. If you try to use a WAS 5 datasource with an EJB 1.1 module, for example, the application will fail to start or will start with errors.

The guidelines for datasource use are:

- **J2EE 1.2 applications**

  All EJB beans, JDBC applications, or Servlets 2.2 must use the **4.0** datasource.

- **J2EE 1.3 applications**

  a. EJB 1.1 Modules (1.1 deployment descriptor)

     All EJB 1.x beans must use the **4.0** datasource.

  b. EJB 2.0 Modules (2.0 deployment descriptor)

     All EJB beans including CMP version 2.0 and 1.x must use the **5.0** data source.

  c. JDBC applications and Servlet 2.3

     Must use the **5.0** data source.

Table 18-1 outlines the possible combinations of EJB modules and datasources supported by WebSphere 5.0.

Table 18-1 WAS4 vs WAS5 datasources

	EJB 1.1	**EJB 2.0**
J2EE 1.2 Deployment Descriptor	4.0 Datasources	N/A
J2EE 1.3 Deployment Descriptor	5.0 Datasources (recommended) or 4.0 Datasources	5.0 Datasources

Author Comment: There is discrepancy between whether ejb 1.1 with j2ee 1.3 deployment descriptor can use 4.0 datasources. The infocenter, text above the table, says no. The table, taken from the was 5.0 persistence manager presentation says yes. They need to be made consistent.

18.7 Creating a datasource

The following steps are involved in creating a datasource:

1. Create a JDBC provider resource

   The JDBC provider gives the classpath of the datasource implementation class and the supporting classes for database connectivity. This is vendor specific.

2. Create a datasource resource

   The JDBC datasource encapsulates the database specific connection settings.

Note: The JDBC provider needs to be created before we can create the datasource.

18.7.1 Creating a JDBC provider

To create a JDBC provider, complete the following steps from the administrative console:

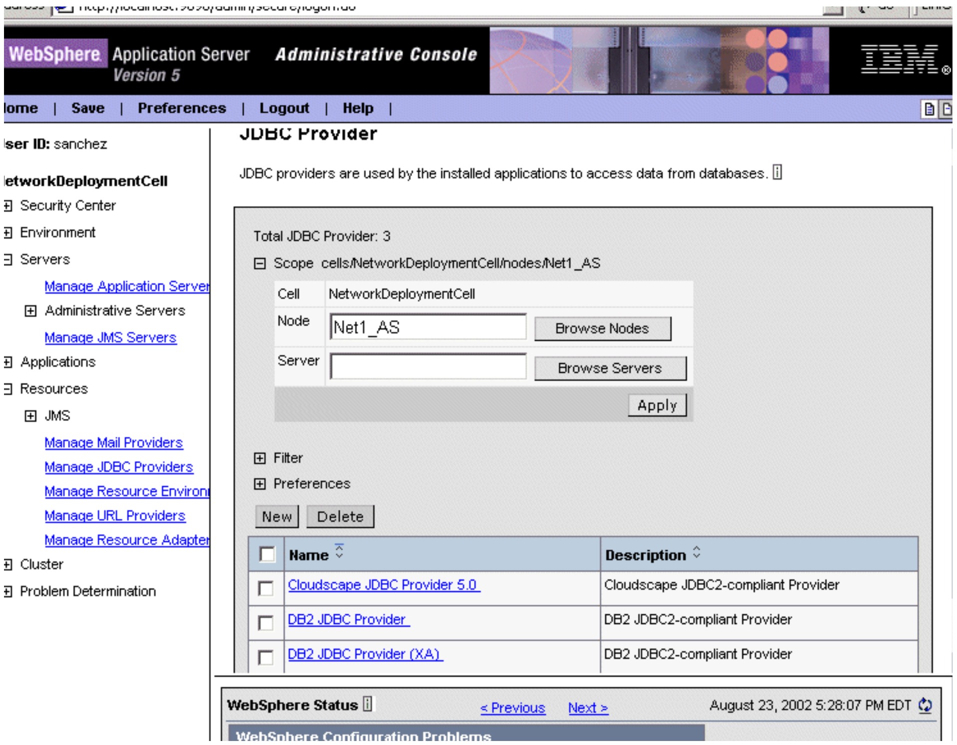
1. Expand **`Resources`** from the navigation tree.
2. Click on **`Manage JDBC Providers`**.
3. Select the **`Scope`** and click **`Apply`**.

   Note: JDBC resources, i.e datasources, are created at a specific scope level. The datasource scope level is inherited from the JDBC Provider. For example, if we create a JDBC provider at the node level, and then create a datasource using that JDBC provider, the datasource will inherit:

   - The JDBC provider settings, such as classpath, implementation class, etcetera.
   - The JDBC provider scope level. In this example, if the scope was set to node level, all application servers running on that node will register the datasource in their namespace.

   The *resources.xml* file, will also get updated at the node and application server level.

   The console will now show all the JDBC Providers created at that scope level, see Figure 18-31. In this example we have created three JDBC Providers, `DB2 JDBC Provider`, `DB2 JDBC Provider (XA)` and `Cloudscape JDBC Provider` 5.0 at the node level.



JDBC Provider

JDBC providers are used by the installed applications to access data from databases.

Total JDBC Provider: 3

Scope cells/NetworkDeploymentCell/nodes/Net1_AS

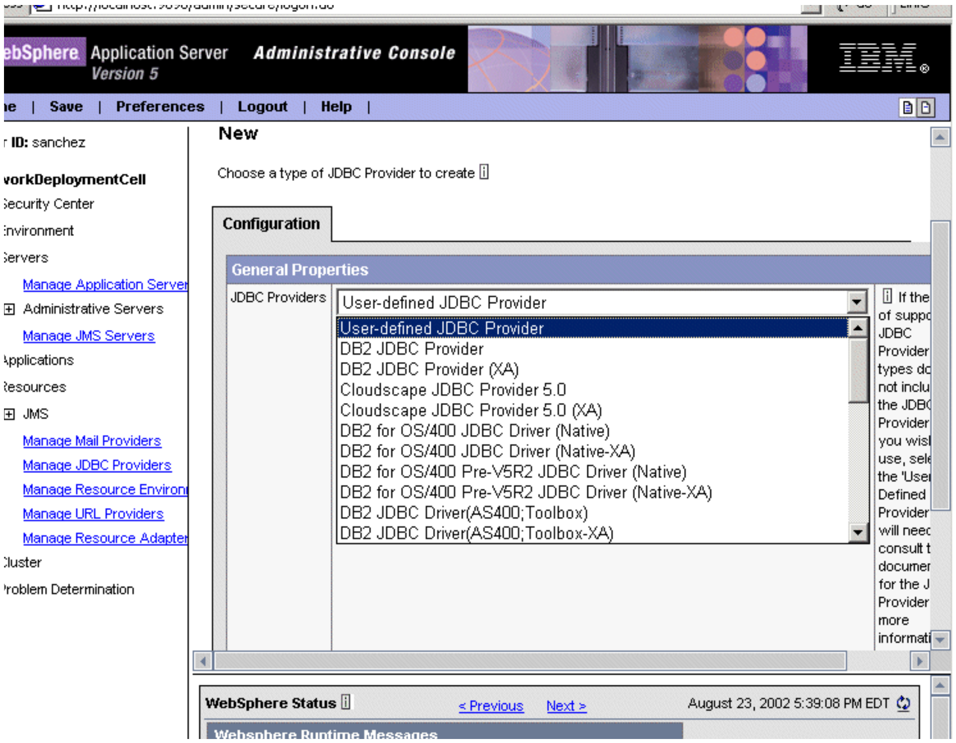
Cell	NetworkDeploymentCell
Node	Net1_AS
Server	

	Name	Description
☐	Cloudscape JDBC Provider 5.0	Cloudscape JDBC2-compliant Provider
☐	DB2 JDBC Provider	DB2 JDBC2-compliant Provider
☐	DB2 JDBC Provider (XA)	DB2 JDBC2-compliant Provider

August 23, 2002 5:28:07 PM EDT

Figure 18-31 JDBC Providers created at the selected scope

4. Select **New** to create a new JDBC Provider.
5. Use the drop-down list to select the type of JDBC Provider you want to create. This is shown in Figure 18-32.



New

Choose a type of JDBC Provider to create

Configuration

General Properties

JDBC Providers: User-defined JDBC Provider

- User-defined JDBC Provider
- DB2 JDBC Provider
- DB2 JDBC Provider (XA)
- Cloudscape JDBC Provider 5.0
- Cloudscape JDBC Provider 5.0 (XA)
- DB2 for OS/400 JDBC Driver (Native)
- DB2 for OS/400 JDBC Driver (Native-XA)
- DB2 for OS/400 Pre-V5R2 JDBC Driver (Native)
- DB2 for OS/400 Pre-V5R2 JDBC Driver (Native-XA)
- DB2 JDBC Driver(AS400;Toolbox)
- DB2 JDBC Driver(AS400;Toolbox-XA)

If the of suppo JDBC Provider types do not inclu the JDBC Provider you wis use, sele the 'User Defined Provider will need consult t documen for the J Provider more informati

Figure 18-32 List of supported JDBC Providers

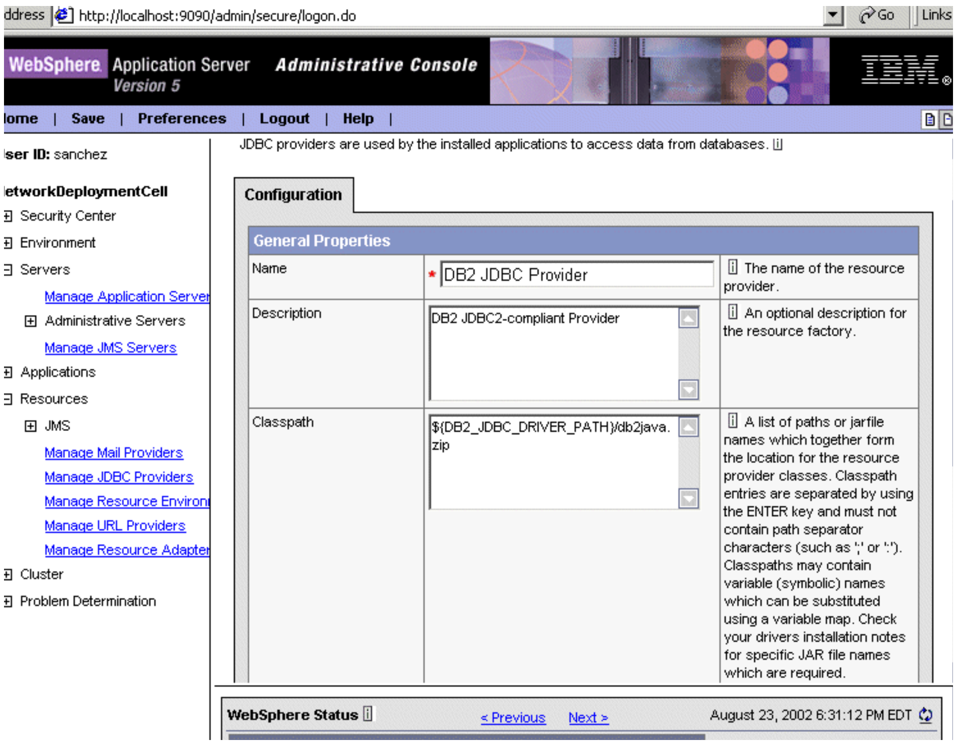
For example, use the `DB2 JDBC Provider` if you want your datasource to connect to a DB2 database using connection pooling.

Connection pooling is a mechanism whereby when an application closes a connection, that connection is recycled rather than being destroyed. Because establishing a connection is an expensive operation, reusing connections can improve performance dramatically by cutting down on the number of new connections that need to be created.

Use the `DB2 JDBC Provider (XA)` if you want your datasource to connect to a DB2 database using connection pooling, and the connections need to be used in distributed transactions (two-phase commit transactions). A connection capable of being used in a distributed transaction can also be used for a non- distributed transaction.

If the list of supported JDBC Provider types does not include the JDBC Provider that you wish to use, select the User-Defined JDBC Provider. You will need to consult the vendor’s documentation for the JDBC Provider for information on specific properties that might be required.

6. Click **`Apply`**. The settings page for your JDBC Provider appears. Figure 18-33 shows the configuration page for a `DB2 JDBC Provider`.



JDBC providers are used by the installed applications to access data from databases.

General Properties		
Name	* DB2 JDBC Provider	The name of the resource provider.
Description	DB2 JDBC2-compliant Provider	An optional description for the resource factory.
Classpath	${DB2_JDBC_DRIVER_PATH}/db2java.zip	A list of paths or jarfile names which together form the location for the resource provider classes. Classpath entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Classpaths may contain variable (symbolic) names which can be substituted using a variable map. Check your drivers installation notes for specific JAR file names which are required.

Figure 18-33 JDBC Provider properties

7. Enter the JDBC provider properties.

   As shown in Figure 18-33, the JDBC provider general properties are:

   a. **Name**

      A name for the provider. It is recommended you enter a name that is suggestive of the database product you are using. The default name for a DB2 JDBC Provider is DB2 JDBC Provider.

   b. **Description**

      A description of the provider, for your administrative records.

   c. **Classpath**

      A list of paths or JAR file names which together form the location for the resource provider classes. For example `c:\ibm\sqllib\java\db2java.zip` if the datasource connects to DB2.

Note:

- The use of WebSphere environment variables gives you more flexibility. For example, if you selected *Cell* as the scope level, and you specified the *Classpath* as per above, this provider would only work on those nodes in which the db2java.zip file is installed to the same directory structure. Applications running on UNIX nodes for example, wouldn't be able to use this JDBC Provider's datasource.

  Refer to Chapter 14, "Configuring the environment" on page 509 for more information on WebSphere environment variables.
- Classpath entries need to be separated by the <Enter> key.

d. **Native Path**

   An optional path to any native libraries. Entries are required if the JBDC provider chosen uses non-Java (native) libraries.

e. **Implementation class**

   The name of the Java datasource class used to connect to the database, as provided by the database vendor. This is provided for you when you select the type of JDBC Provider.

   For example, the one-phase commit protocol classes for DB2 and Oracle are:

   - DB2: `COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource`
   - Oracle: `oracle.jdbc.pool.OracleConnectionPoolDataSource`

   This class must be available in the driver file specified by **Classpath** property above.

8. Click **`OK`**, when finished configuring the JDBC Provider.
9. Click **`Save`**, to save the configuration.

Tip: To make a datasource available on multiple nodes using different directory structures, complete the following steps:

1. Define the JDBC Provider at the Cell scope.

   Use WebSphere environment variables for the classpath and native path.

2. Create the datasource that uses this JDBC Provider.

   This will define the datasource resource in the *resources.xml* file at the cell directory level (config/cells/<cellname>/resources.xml). All files defined at the cell scope are replicated to every node in the cell.

3. For the paths to the driver on each node to be unique, use a variable to specify the driver location and have that variable be defined differently on each node.

   For example, ${DRIVER_PATH} can be used for the classpath in the provider definition. You can then have a definition of ${DRIVER_PATH} in the cell level *variables.xml* file to act as a default driver location. Then you can override that variable on any node by defining ${DRIVER_PATH} in the *variables.xml* file for that node (which takes precedence over the cell level definition)

18.7.2 Creating the JDBC datasource

After creating and configuring the JDBC Provider, you can now create a datasource under it. The datasource will use the JDBC Provider classes to connect to the database.

Creating a WAS 4 datasource

To create a WAS 4 datasource, complete these steps:

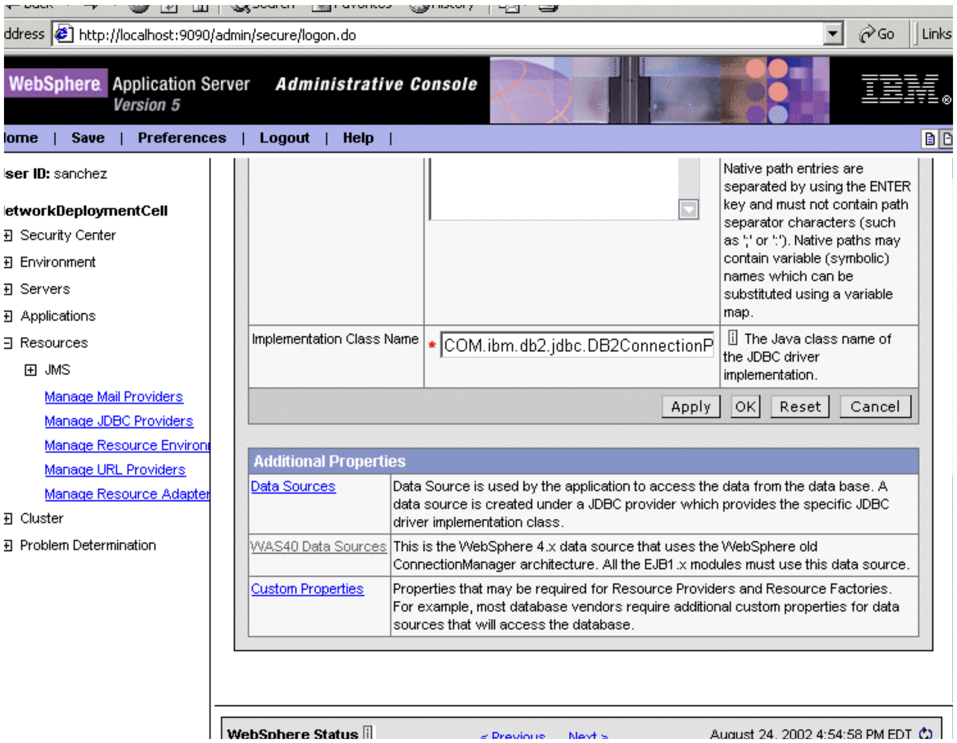
1. Expand **Resources** from the navigation tree.
2. Click on **Manage JDBC Providers**.
3. Select the **Scope** and click **Apply**.

   This shows all the JDBC Providers created at that scope. See Section 18.7.1, “Creating a JDBC provider” on page 700, for more information on the scope setting.

4. Select the JDBC Provider to be used by your WAS 4 datasource.

   This brings up the configuration panel for that JDBC Provider.

5. Select **WAS40_Data_Sources** under **Additional Properties** as shown in Figure 18-34:



		Native path entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Native paths may contain variable (symbolic) names which can be substituted using a variable map.
Implementation Class Name	* COM.ibm.db2.jdbc.DB2ConnectionP	The Java class name of the JDBC driver implementation.

Additional Properties

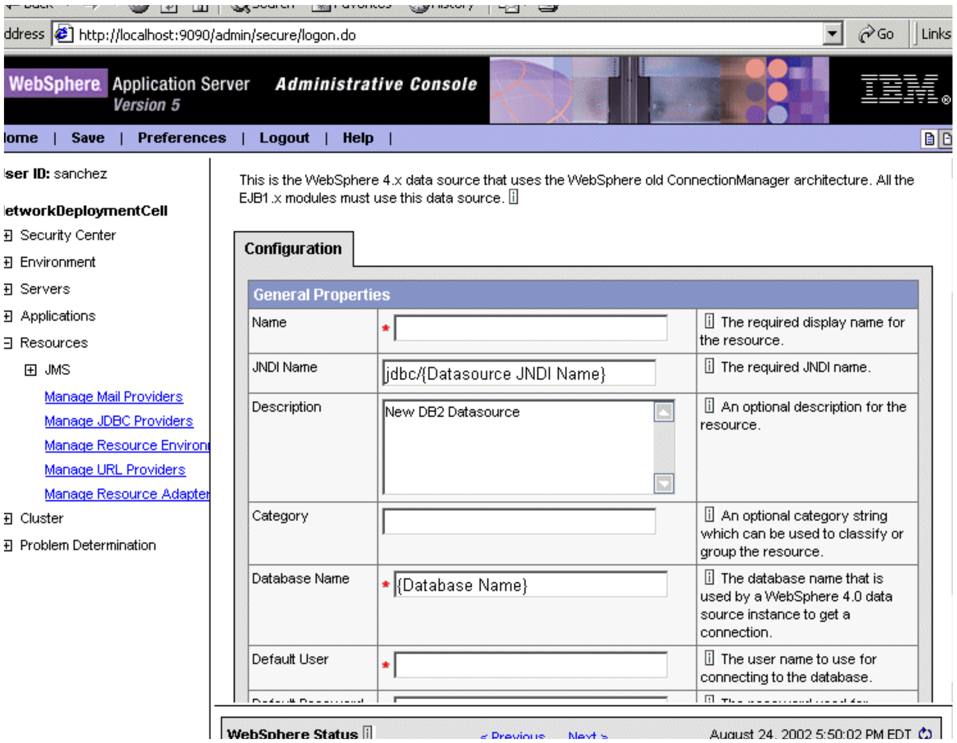
Data Sources	Data Source is used by the application to access the data from the data base. A data source is created under a JDBC provider which provides the specific JDBC driver implementation class.
WAS40 Data Sources	This is the WebSphere 4.x data source that uses the WebSphere old ConnectionManager architecture. All the EJB1.x modules must use this data source.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Figure 18-34 Configurable resources for the JDBC provider

The **WAS40 Data Source page** will show all the WAS40 Data Sources configured for that particular JDBC Provider.

6. Click **`New`** to create a new WAS4 datasource, or select an existing one to modify the datasource properties.

   The **WAS40_Data_Source** properties are shown in Figure 18-35:


ddress http://localhost:9090/admin/secure/logon.do

This is the WebSphere 4.x data source that uses the WebSphere old ConnectionManager architecture. All the EJB1.x modules must use this data source.

General Properties

Name	*	The required display name for the resource.
JNDI Name	jdbc/{Datasource JNDI Name}	The required JNDI name.
Description	New DB2 Datasource	An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Database Name	* {Database Name}	The database name that is used by a WebSphere 4.0 data source instance to get a connection.
Default User	*	The user name to use for connecting to the database.

Figure 18-35 WAS40_Datasource properties

a. **Name**

A name by which to administer the datasource. Use a name that is suggestive of the database you will use to store data, such as SampleDataSource, where SAMPLE is the database name.

b. **JNDI name**

The datasource's name as registered in the application server's name space.

When installing an application that contains modules with JDBC resource references, the resources defined by the deployment descriptor of the module need to be bound to the actual JNDI name of the resources.

Tip: As a convention, use the value of the **Name** property prefixed with "jdbc/"

c. **Description**

A description of the datasource, for your administrative records.

d. **Category**

   Specifies a category that you can use to classify or group the datasource.

e. **Database name**

   The name of the database that this datasource will access. Following with the example above, this could be the SAMPLE database.

f. **Default User**

   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.

g. **Default 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.

7. In Figure 18-35, under **Additional Properties,** select **Custom Properties**. Some database vendors might require additional custom properties for datasources that will access their database. These are a set of name-value pairs describing properties of the data source. Check the database vendor documentation.

   For example you would need to configure the “URL” property for the Oracle thin driver provider in order to provide database connection details:

   – **Name**. URL

   – **Value**. jdbc:oracle:thin:@<hostname>:<port number>:<databasename>

8. Click `OK`, when finished configuring the datasource.
9. Click `Save`, to save the configuration.

Configuring connection pooling for a WAS 4 datasource

Establishing JDBC connections is resource expensive. Performance can be improved significantly when connection pooling is used. Connection pooling means that connections are reused rather than created each time a connection is requested. For a WAS 4 datasource the connection pooling is handled by the *WebSphere JDBC connection manager*.

Whether connection pooling is used or not, it does not affect the application code. The application does not require any code changes. All the application does is to perform a lookup on a JNDI name of a previously registered datasource and then issue a *getConnection()*. If the datasource class implements connection pooling, then the client will obtain a connection from a pool of connections.

To configure the connection pool for a WAS 4 datasource, complete these steps:

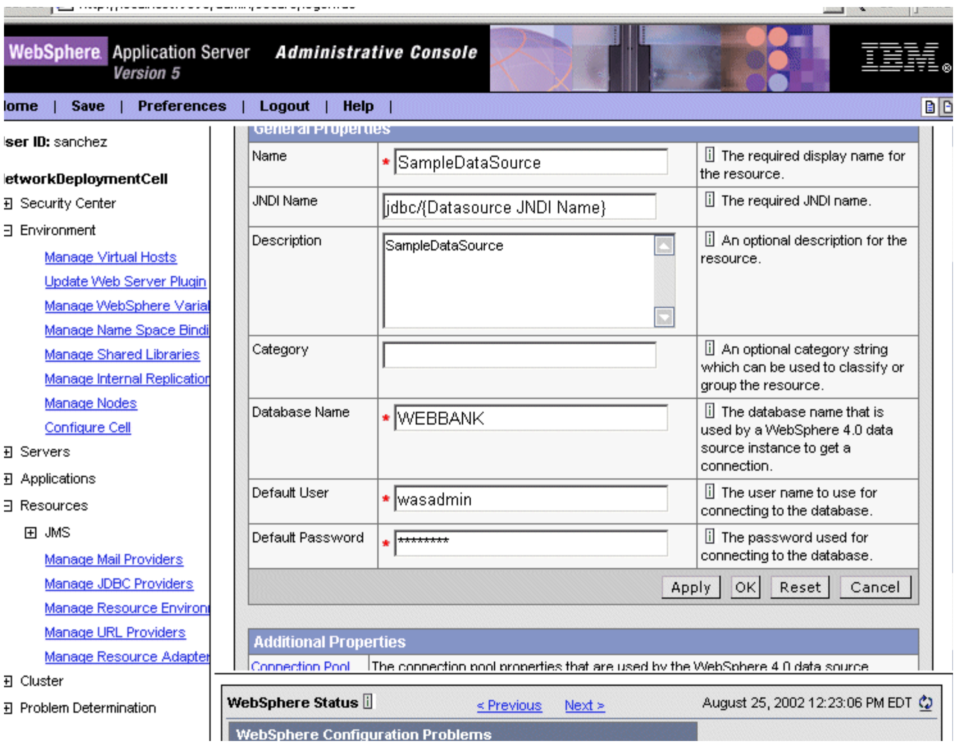
1. Expand **Resources** from the navigation tree.
2. Click on **Manage JDBC Providers**.
3. Select the **Scope** and click **Apply**.

   This shows all the JDBC Providers created at that scope. See Section 18.7.1, “Creating a JDBC provider” on page 700, for more information on the scope setting.
4. Select the JDBC Provider used by your WAS 4 datasource.

   This brings the Configuration panel for that JDBC Provider.
5. Select **WAS40_Data_Sources** under **Additional Properties**.

   The content pane will show all the WAS 4 datasources configured for that provider.
6. Select the datasource for which you want to configure connection pooling.

   This brings the Configuration page for the selected datasource, as shown in Figure 18-36.



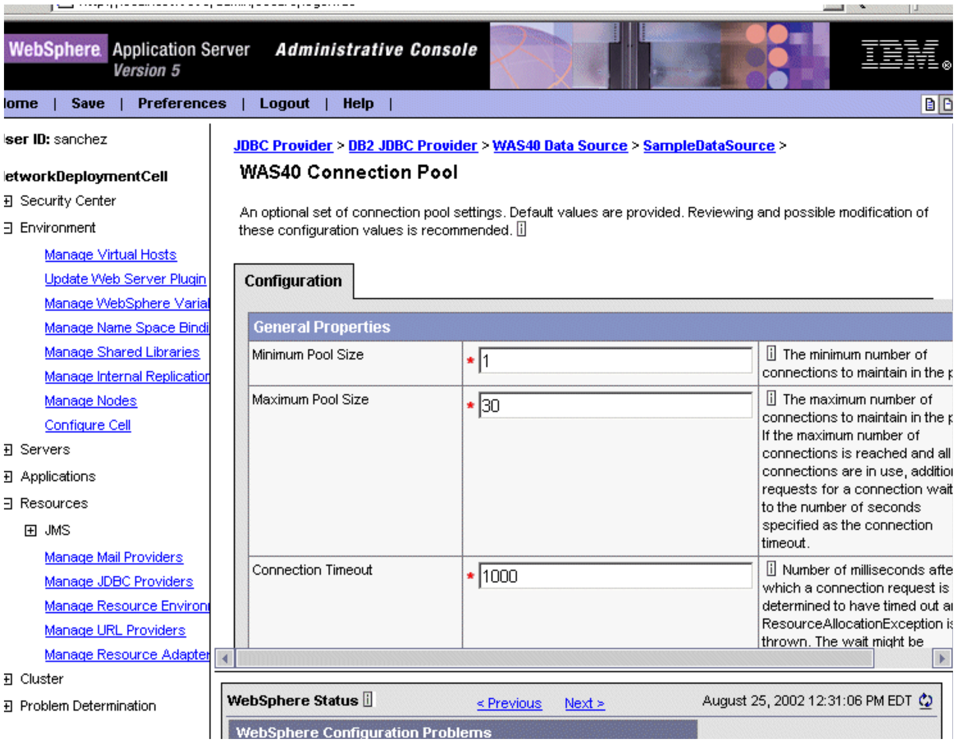
Name	* SampleDataSource	The required display name for the resource.
JNDI Name	jdbc/{Datasource JNDI Name}	The required JNDI name.
Description	SampleDataSource	An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Database Name	* WEBBANK	The database name that is used by a WebSphere 4.0 data source instance to get a connection.
Default User	* wasadmin	The user name to use for connecting to the database.
Default Password	* ********	The password used for connecting to the database.

Connection Pool	The connection pool properties that are used by the WebSphere 4.0 data source.

Figure 18-36 WAS4 datasource configuration page

7. Under **`Additional Properties`**, select **`Connection Pool`**.

   The properties to configure for a WAS40 datasource connection pool are as shown in Figure 18-37:



WAS40 Connection Pool

An optional set of connection pool settings. Default values are provided. Reviewing and possible modification of these configuration values is recommended.

General Properties

Minimum Pool Size	* 1	The minimum number of connections to maintain in the
Maximum Pool Size	* 30	The maximum number of connections to maintain in the If the maximum number of connections is reached and all connections are in use, additio requests for a connection wait to the number of seconds specified as the connection timeout.
Connection Timeout	* 1000	Number of milliseconds afte which a connection request is determined to have timed out a ResourceAllocationException i thrown. The wait might be

Figure 18-37 WAS 4 datasource connection pool properties

   a. **Minimum pool size**

      The minimum number of connections to maintain in the pool.

      The minimum pool size can affect the performance of an application. Smaller pools require less overhead when the demand is low because fewer connections are being held open to the database. On the other hand, when the demand is high, the first applications will experience a slow response because new connections will have to be created if all others in the pool are in use.

Note: The pool does not start out with the minimum number of connections. As applications request connections, the pool grows up to the minimum number. After the pool has reached the minimum number of connections, it does not shrink beyond this minimum unless an exception occurs that requires the entire pool to be destroyed.

b. **Maximum pool size**

The maximum number of connections to maintain in the pool. If the maximum number of connections is reached and all connections are in use, additional requests for a connection will wait up to the number of seconds specified in the **Connection timeout** property.

The maximum pool size can affect the performance of an application. Larger pools require more overhead when demand is high because there are more connections open to the database at peak demand. These connections persist until they are idled out of the pool as specified by the **Idle timeout** property.

On the other hand, if the maximum is smaller, there might be longer wait times or possible connection timeout errors during peak times. The database must be able to support the maximum number of connections in the application server in addition to any load that it may have outside of the application server.

c. **Connection timeout**

The maximum number of seconds that an application waits for a connection from the pool before timing out and throwing a *ConnectionWaitTimeoutException* to the application. This can occur when the pool is at its maximum and all of the connections are in use by other applications for the duration of the wait.

In addition, there are no connections currently in use that the application can share, because either the user name and password are different or it is in a different transaction.Setting this value to 0 disables the connection timeout.

d. **Idle timeout**

The maximum number of seconds that an idle (unallocated) connection can remain in the pool before the connection is removed from the pool and closed.

Connections need to be idled out of the pool because keeping connections open to the database can cause memory problems with the database in some cases. However, not all connections are idled out of the pool, even if they are older than the idle timeout number of seconds. A connection is not idled if removing the connection would cause the pool to shrink below its minimum size. Setting this value to 0 disables the idle timeout.

e. **Orphan timeout**

   The maximum number of seconds that an application can hold a connection without using it before the connection is returned to the pool.

   If there has been no activity on an allocated connection for longer than the orphan timeout number of seconds, the connection is marked for orphaning. After another orphan timeout number of seconds, if the connection still has had no activity, the connection is returned to the pool. If the application tries to use the connection again, it is thrown a *StaleConnectionException*. Connections that are enlisted in a transaction are not orphaned. Setting this value to 0 disables the orphan timeout.

   The orphan timeout is a provision to handle an application's failure to release connections or other similar failures.

   Note: The actual amount of time before a connection is closed is approximately twice the orphan timeout value.

f. **Statement cache size**

   The maximum number of cached prepared statements to keep per connection.

   Statement caching improves performance. Retrieval of a matching statement from the cache takes less overhead than creating a new prepared statement. The statement cache size does not change the programming model, only the performance of the application. The statement cache size is the number of cached statements *for each connection.*

g. **Disable AutoConnection cleanup**

   Tells the connection pooling software whether or not to automatically close connections from this datasource at the end of a transaction.

   The default is *false*, which indicates that when a transaction is completed, WebSphere connection pooling closes the connection and returns it to the pool. This means that any use of the connection after the transaction has ended results in a *StaleConnectionException*, because the connection has been closed and returned to the pool.

   This mechanism ensures that connections are not held indefinitely by the application. If the value is set to *true*, the connection is not returned to the pool at the end of a transaction. In this case, the application must return the connection to the pool by explicitly calling *close()*. If the application does not close the connection, the pool can run out of connections for other applications to use. This is different from orphaning connections, because connections in a transaction cannot be orphaned.

8. Click **OK**, when finished configuring the datasource.
9. Click **Save**, to save the configuration.

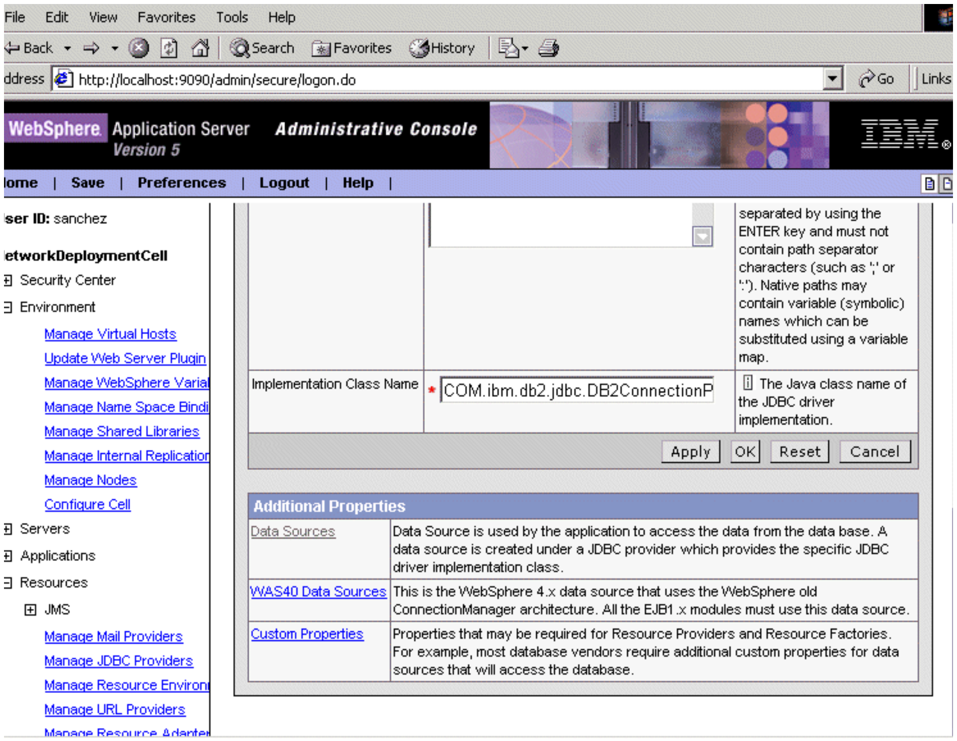
Creating a WAS 5 datasource

To create a WAS 5 datasource complete the following steps:

1. Expand **Resources** from the navigation tree.
2. Click on **Manage JDBC Providers**.
3. Select the **Scope** and click **Apply**.

   This shows all the JDBC Providers created at that scope. See Section 18.7.1, “Creating a JDBC provider” on page 700, for more information on the scope setting.
4. Select the JDBC Provider to be used by your WAS 5 datasource.

   This brings the Configuration panel for that JDBC Provider.
5. Select **Data_Sources** under **Additional Properties** as shown in Figure 18-38:



separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Native paths may contain variable (symbolic) names which can be substituted using a variable map.

Implementation Class Name	* COM.ibm.db2.jdbc.DB2ConnectionP	The Java class name of the JDBC driver implementation.

Additional Properties

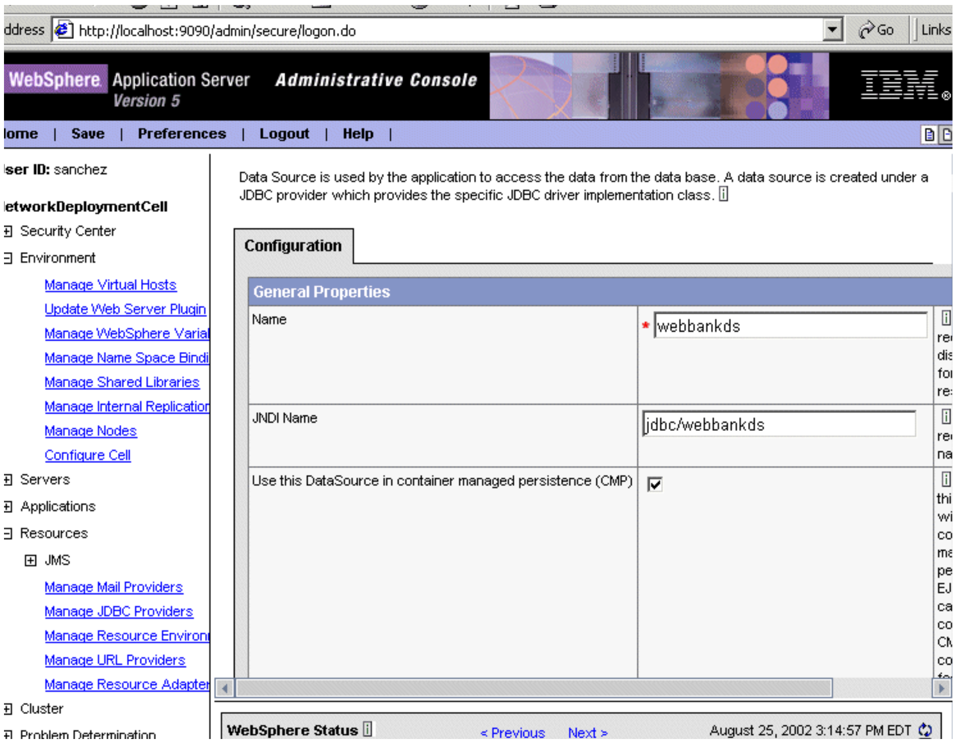
Data Sources	Data Source is used by the application to access the data from the data base. A data source is created under a JDBC provider which provides the specific JDBC driver implementation class.
WAS40 Data Sources	This is the WebSphere 4.x data source that uses the WebSphere old ConnectionManager architecture. All the EJB1.x modules must use this data source.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Figure 18-38 Configurable resources for the JDBC provider

The **Data Source** page will show all the Data Sources (WAS 5 datasources) configured for that particular JDBC Provider.

6. Click **`New`** to create a new WAS 5 datasource, or select an existing one to modify the datasource properties.

   The **Data Source** (WAS 5 datasource) properties page is shown in Figure 18-39:



Data Source is used by the application to access the data from the data base. A data source is created under a JDBC provider which provides the specific JDBC driver implementation class.

Name	* webbankds
JNDI Name	jdbc/webbankds
Use this DataSource in container managed persistence (CMP)	✓

Figure 18-39 WAS5 datasource properties

a. **Name**

A name by which to administer the datasource. Use a name that is suggestive of the database you will use to store data, such as webbankds, see Figure 18-39, where WEBBANK is the database name.

b. **JNDI name**

The datasource’s name as registered in the application server’s name space.

When installing an application that contains modules with JDBC resource references, the resources defined by the deployment descriptor of the module need to be bound to the actual JNDI name of the resources.

Tip: As a convention, use the value of the **Name** property prefixed with “jdbc/”

c. **Use this DataSource in container managed persistence (CMP)**

Specifies if the datasource is used for container managed persistence of EJB beans.

> **Note:** You need to select this checkbox if your application uses CMP 2.0 beans. It is not necessary to check this box if you only want BMP or JDBC access.

This setting is enabled by clicking on the checkbox. This causes a **CMP Connector Factory** corresponding to this datasource to be created for the relational resource adapter. The Connector Factory created has the name `datasourcename_CF` and is registered in JNDI under the entry `eis/datasourcename_CMP`.

> **Tip:** You can see the properties of the just created connection factory by selecting **`Resources`** -> **`Manage Resource Adapters`** -> **`WebSphere Relational Resource Adapter`** -> **`CMP Connection Factories`** from the administrative console.

CMP 2.0 EJBs beans will use this CMP Connection Factory in order to get J2C connections. The Persistence Manager will use this JCA Connection Factory at runtime. The look up of the connection factory is totally transparent to the programmer.

In Example 18-2, we see how a CMP 2.0 EJB, `webbank/ejb/CustomerAccount`, needs it *CMPConnectionFactoryBinding* bound to the JNDI name of the connection factory, `eis/webbankds_CMP`, in the example.

> **Note:** Make sure that when you specify the datasource to use for a CMP 2.0 EJB, that you ultimately bind your component to the connection factory. This binding is done during application installation.

The `WebBank` application was deployed in Chapter 20, “Deploying an application” on page 835. Refer to it for more information.

Author Comment: Some datasource configuration parameters such as how credentials are passed to the datasource are associated to the binding. Refer to Isabelle’s deployment chapter for resource reference binding attributes such as isolation level, shareability of connections or resource authorization. Or make sure this is covered somewhere

Example 18-2 ibm-ejb-jar-bnd.xmi file for webbankEntityEJBs.jar

```
<ejbBindings
xmi:id="CustomerAccount_Bnd"jndiName="webbank/ejb/CustomerAccount">
    <enterpriseBean xmi:type="ejb:ContainerManagedEntity"
href="META-INF/ejb-jar.xml#CustomerAccount"/>
    <cmpConnectionFactory xmi:id="CMPConnectionFactoryBinding_2"
jndiName="eis/webbankds_CMP" resAuth="Per_Connection_Factory"/>
  </ejbBindings>
```

Notice also that the *resources.xml* file, at the selected scope, contains two entries, one for the *CMPConnectorFactory* object, Example 18-3, and another one for the *DataSource objec*t, Example 18-4. The same can be seen if you use **dumpnamespace** to dump the JNDI name space of an application server under the selected scope.

Example 18-3 CMPConnectionFactory resource reference in the resources.xml file

```
<factories xmi:type="resources.jdbc:CMPConnectorFactory"
xmi:id="CMPConnectorFactory_1" name="webbankds_CF"
authMechanismPreference="BASIC_PASSWORD" cmpDatasource="DataSource_1">
      <authDataAlias xsi:nil="true"/>
      <propertySet xmi:id="J2EEResourcePropertySet_1">
        <resourceProperties xmi:id="J2EEResourceProperty_1"
name="TransactionResourceRegistration" type="java.lang.String" value="dynamic"
description="Type of transaction resource registration (enlistment).  Valid
values are either &#34;static&#34; (immediate) or &#34;dynamic&#34;
(deferred)."/>
        <resourceProperties xmi:id="J2EEResourceProperty_2"
name="InactiveConnectionSupport" type="java.lang.Boolean" value="true"
description="Specify whether connection handles support implicit reactivation.
(Smart Handle support). Value may be &#34;true&#34; or &#34;false&#34;."/>
      </propertySet>
</factories>
```

Example 18-4 Datasource resource reference in the resources.xml file

```
<factories xmi:type="resources.jdbc:DataSource" xmi:id="DataSource_1"
name="webbankds" jndiName="webbankds" description="New JDBC Datasource"
statementCacheSize="10"
datasourceHelperClassname="com.ibm.websphere.rsadapter.DB2DataStoreHelper"
relationalResourceAdapter="builtin_rra">
      <propertySet xmi:id="J2EEResourcePropertySet_2">
       <resourceProperties xmi:id="J2EEResourceProperty_3" name="databaseName"
type="java.lang.String" value="WEBBANK" description="This is a required
property. The database name. For example, enter sample to make your Data Source
point to jdbc:db2:sample." required="false"/>
        <resourceProperties xmi:id="J2EEResourceProperty_8" name="user"
type="java.lang.String" value="wasadmin" required="false"/>
```

```
        <resourceProperties xmi:id="J2EEResourceProperty_9" name="password" 
type="java.lang.String" value="wasadmin" required="false"/>
      </propertySet>
      <connectionPool xmi:id="ConnectionPool_4" connectionTimeout="1800" 
maxConnections="10" minConnections="1" reapTime="180" unusedTimeout="1800" 
purgePolicy="EntirePool"/>
</factories>
```

d. **Description**

   A description of the datasource, for your administrative records.

e. **Category**

   Specifies a category that you can use to classify or group the datasource.

f. **Statement Cache Size**

   Specifies the number of prepared statements that are cached per connection.

g. **Datasource Helper Classname**

   Specifies the datastore helper that is used to perform database specific functions.

   This is used by the Relational Resource Adapter at runtime. The default DataStoreHelper implementation class is set based on the JDBC driver implementation class, using the structure: com.ibm.websphere.rsadapter.<database>DataStoreHelper. For example, if the JDBC provider is DB2, then the default DataStoreHelper class is:

   `com.ibm.websphere.rsadapter.DB2DataStoreHelper`

   You can change to value to refer to a specific a subclass of this DataStoreHelper if necessary.

h. **Authentication Data Alias**

   Specifies an alias to an authentication data module (consisting of a user ID and password) used by Java 2 Connector security.

   Each user ID and password entry is identified by a unique alias string.

   The alias is created under **`Security Center`** -> **`JAAS Configuration`** -> **`J2C Auth Entries`** from the administrative console.

7. Click **`OK`**.
8. Click **Custom Properties**, under **Additional Properties,** to provide other datasource properties such as:

   a. **databaseName**

      The name of the database that this datasource will access.

b. **user**

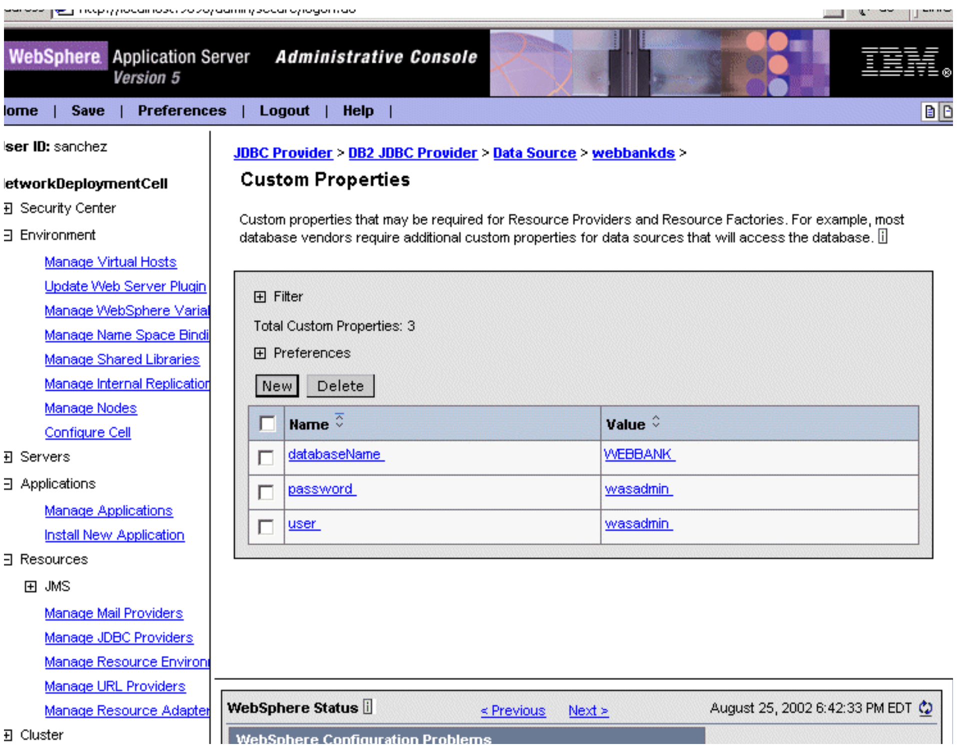
The user name for connecting to the database when no user ID and password pair is specified by the application. If the user ID is specified, the password must also be specified.

c. **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.

Click **`New`** to create each one of them and click **`OK`** afterwards.

9. Figure 18-40 shows custom properties configured for a WAS 5 datasource connecting to a DB2 database (the only required parameter is databaseName). Other database vendors might require different custom properties. Check the database vendor documentation.



JDBC Provider > DB2 JDBC Provider > Data Source > webbankds >

Custom Properties

Custom properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Filter

Total Custom Properties: 3

Preferences

New Delete

	Name	Value
☐	databaseName	WEBBANK
☐	password	wasadmin
☐	user	wasadmin

WebSphere Status < Previous Next > August 25, 2002 6:42:33 PM EDT

WebSphere Configuration Problems

Figure 18-40 Datasource custom properties

10. Click **`OK`**, when finished configuring the datasource.

11. Click **`Save`**, to save the configuration.

Configuring connection pooling for a WAS 5 datasource

To configure the connection pool for a WAS 5 datasource, complete the following steps:

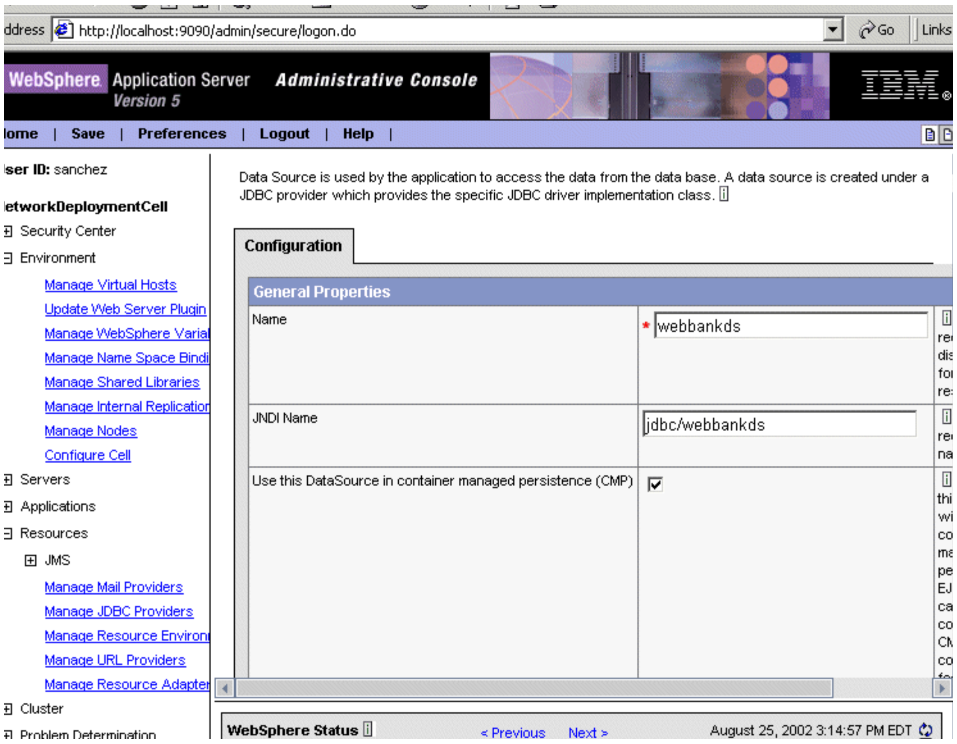
1. Expand **`Resources`** from the navigation tree.
2. Click on **`Manage JDBC Providers`**.
3. Select the **`Scope`** and click **`Apply`**.

   This shows all the JDBC Providers created at that scope. See Section 18.7.1, “Creating a JDBC provider” on page 700, for more information on the scope setting.
4. Select the JDBC Provider used by your WAS 5 datasource.

   This brings the Configuration panel for that JDBC Provider.
5. Select **Data_Sources** under **`Additional Properties`**

   The content pane will show all the WAS 5 datasources configured for that provider.
6. Select the datasource for which you want to configure connection pooling.

   This brings the Configuration page for the selected datasource, as shown in Figure 18-41.



Data Source is used by the application to access the data from the data base. A data source is created under a JDBC provider which provides the specific JDBC driver implementation class.

Configuration

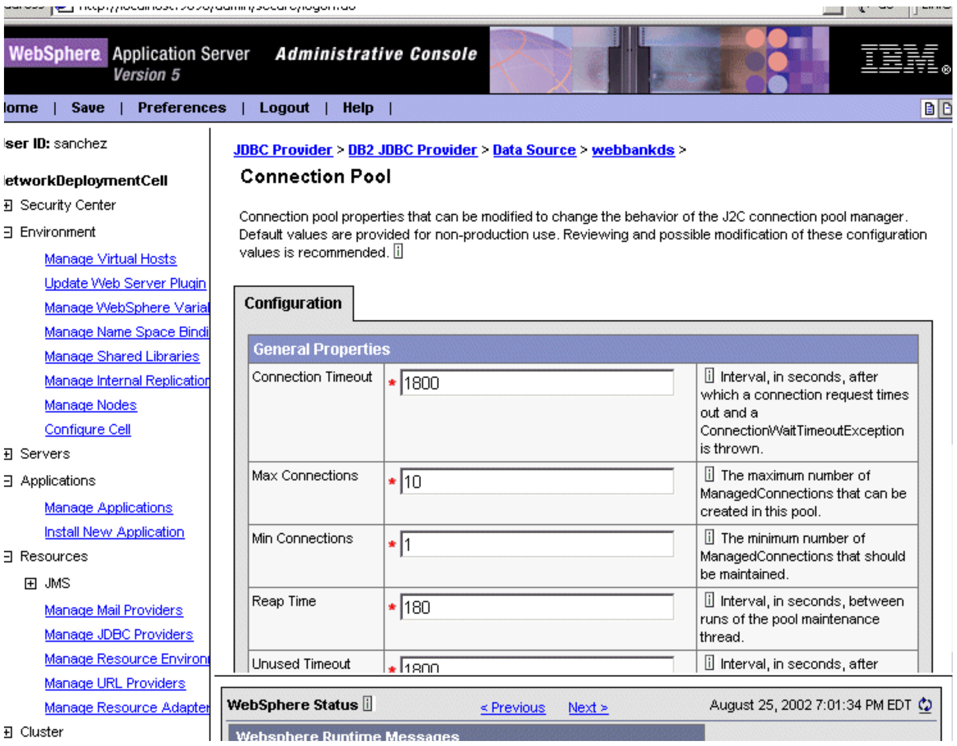
General Properties

Name	* webbankds
JNDI Name	jdbc/webbankds
Use this DataSource in container managed persistence (CMP)	☑

Figure 18-41 WAS5 datasource configuration page

7. Under **`Additional Properties`**, select **`Connection Pool`**

   The properties to configure for a WAS5 datasource connection pool are as shown in Figure 18-42:



JDBC Provider > DB2 JDBC Provider > Data Source > webbankds >

Connection Pool

Connection pool properties that can be modified to change the behavior of the J2C connection pool manager. Default values are provided for non-production use. Reviewing and possible modification of these configuration values is recommended.

Configuration

General Properties

Connection Timeout	* 1800	Interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.
Max Connections	* 10	The maximum number of ManagedConnections that can be created in this pool.
Min Connections	* 1	The minimum number of ManagedConnections that should be maintained.
Reap Time	* 180	Interval, in seconds, between runs of the pool maintenance thread.
Unused Timeout	* 1800	Interval, in seconds, after

Figure 18-42 WAS 5 datasource connection pool properties

a. **Connection Timeout**

Specifies the interval, in seconds, after which a connection request times out and a *ConnectionWaitTimeoutException* is thrown. This can occur when the pool is at its maximum (**Max Connections**) and all of the connections are in use by other applications for the duration of the wait.

For example, if Connection Timeout is set to 300 and the maximum number of connections is reached, the Pool Manager waits for 300 seconds for an available physical connection. If a physical connection is not available within this time, the Pool Manager throws a *ConnectionWaitTimeoutException*.

Tip: If Connection Timeout is set to 0, the Pool Manager waits as long as necessary until a connection is allocated.

b. **Max Connections**

Specifies the maximum number of physical connections that can be created in this pool.

These are the physical connections to the backend database. Once this number is reached, no new physical connections are created and the requester waits until a physical connection that is currently in use is returned to the pool or a *ConnectionWaitTimeoutException* is thrown.

For example, if Max Connections is set to 5, and there are 5 physical connections in use, the Pool Manager waits for the amount of time specified in Connection Timeout for a physical connection to become free. If after that time there are still no free connections, the Pool Manager throws a *ConnectionWaitTimeoutException* to the application.

c. **Min Connections**

Specifies the minimum number of physical connections to be maintained.

Until this number is reached, the pool maintenence thread does not discard any physical connections. However, no attempt is made to bring the number of connections up to this number.

For example if Min Connections is set to 3, and one physical connection is created, that connection is not discarded by the **Unused Timeout** thread. By the same token, the thread does not automatically create two additional physical connections to reach the Min Connections setting.

d. **Reap Time**

Specifies the interval, in seconds, between runs of the pool maintenence thread.

For example, if Reap Time is set to 60, the pool maintenence thread runs every 60 seconds. The Reap Time interval affects the accuracy of the **Unused Timeout** and **Aged Timeout** settings. The smaller the interval, the greater the accuracy. When the pool maintenance thread runs it discards any connections that have been unused for longer than the time value specified in Unused Timeout, until it reaches the number of connections specified in Min Connections. The pool maintenence thread also discards any connections that remain active longer than the time value specified in Aged Timeout.

Tip: If the pool maintenence thread is enabled, set the Reap Time value less than the values of Unused Timeout and Aged Timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenence thread runs more often and degrades performance.

e. **Unused Timeout**

Specifies the interval in seconds after which an unused or idle connection is discarded.

Tip: Set the Unused Timeout value higher than the Reap Timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections not in use exceeds the Min Connections setting.

For example, if the unused timeout value is set to 120, and the pool maintenence thread is enabled (Reap Time is not 0), any physical connection that remains unused for two minutes is discarded. Note that accuracy of this timeout, as well as performance, is affected by the Reap Time value. See Reap Time for more information.

f. **Aged Timeout**

Specifies the interval in seconds before a physical connection is discarded, regardless of recent usage activity.

Setting Aged Timeout to 0 allows active physical connections to remain in the pool indefinitely. For example if the Aged Timeout value is set to 1200, and the Reap Time value is not 0, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that accuracy of this timeout, as well as performance, is affected by the Reap Time value. See Reap Time for more information.

Tip: Set the Aged Timeout value higher than the Reap Timeout value for optimal performance.

g. **Purge Policy**

Specifies how to purge connections when a stale connection or fatal connection error is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**. If EntirePool is specified all physical connections in the pool are destroyed when a stale connection is detected. If FailingConnectionOnly is specified the pool attempts to destroy only the stale connection, the other connections remain in the pool. Final destruction of connections which are in use at the time of the error might be delayed. However, those connections are never returned to the pool.

8. Click `OK`, when finished configuring the datasource.
9. Click `Save`, to save the configuration.

18.7.3 More information

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

- WebSphere InfoCenter, “Datasources” and “Creating a data source with the administrative console”.

  `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`
- JCA API documentation

  `http://java.sun.com/j2ee/connector/`
- EJB 2.0 documentation -persistence manager

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

18.8 J2EE Connector Architecture (JCA) overview

The J2EE Connector architecture (JCA) defines a standard architecture for connecting the J2EE platform to heterogeneous Enterprise Information Systems (EISs). Examples of EISs include ERP, mainframe transaction processing, database systems, and legacy applications not written in the Java programming language. By defining a set of scalable, secure, and transactional mechanisms, the J2EE Connector architecture enables the integration of EISs with application servers and enterprise applications.

JCA is part of J2EE 1.3, and as such, WebSphere Version 5 provides a complete implementation of the JCA 1.0 specification.

The diagram in Figure 18-43 illustrates the J2EE Connector architecture.

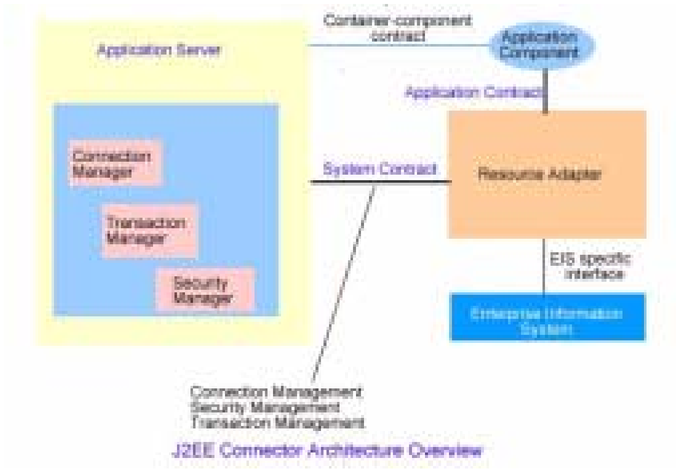


Figure 18-43 J2EE Connector Architecture

The **JCA Resource Adapter** shown in Figure 18-43, is a system level software driver supplied by enterprise information system (EIS) vendors or other third-party vendors. It provides the following functionality:

- Provides connectivity between J2EE components (an application server or an application client) and an EIS.
- Plugs into an application server.
- Collaborates with the application server to provide important services such as connection pooling, transaction and security services.

  The J2EE Connector architecture defines the following set of system-level contracts between an application server and EIS:

  – A **Connection Management contract** that lets an application server pool connections to an underlying EIS, and lets application components connect to an EIS. This leads to a scalable application environment that can support a large number of clients requiring access to EISs.

  – A **Transaction Management contract** between the transaction manager and an EIS that supports transactional access to EIS resource managers. This contract lets an application server use a transaction manager to manage transactions across multiple resource managers. This contract also supports transactions that are managed internal to an EIS resource manager without the necessity of involving an external transaction manager.

  – A **Security contract** that enables a secure access to an EIS. This contract provides support for a secure application environment, which reduces security threats to the EIS and protects valuable information resources managed by the EIS.

    The resource adapter implements the EIS-side of these system-level contracts.
- Implements the **Common Client Interface** (CCI) for EIS access. The CCI defines a standard client API through which a J2EE component accesses the EIS. This simplifies writing code to connect to an EIS's data store.

  The resource adapter provides connectivity between the EIS, the application server and the enterprise application via the Common Client Interface.
- Implements the standard **Service Provider Interface** (SPI) for integrating the transaction, security and connection management facilities of an application server (JCA Connection Manager) with those of a transactional resource manager.

A resource adapter is used within the address space of the application server and multiple resource adapters (that is, one resource adapter per type of EIS) are pluggable into an application server. This capability enables application components deployed on the application server to access the underlying EISs. This is shown in Figure 18-44:

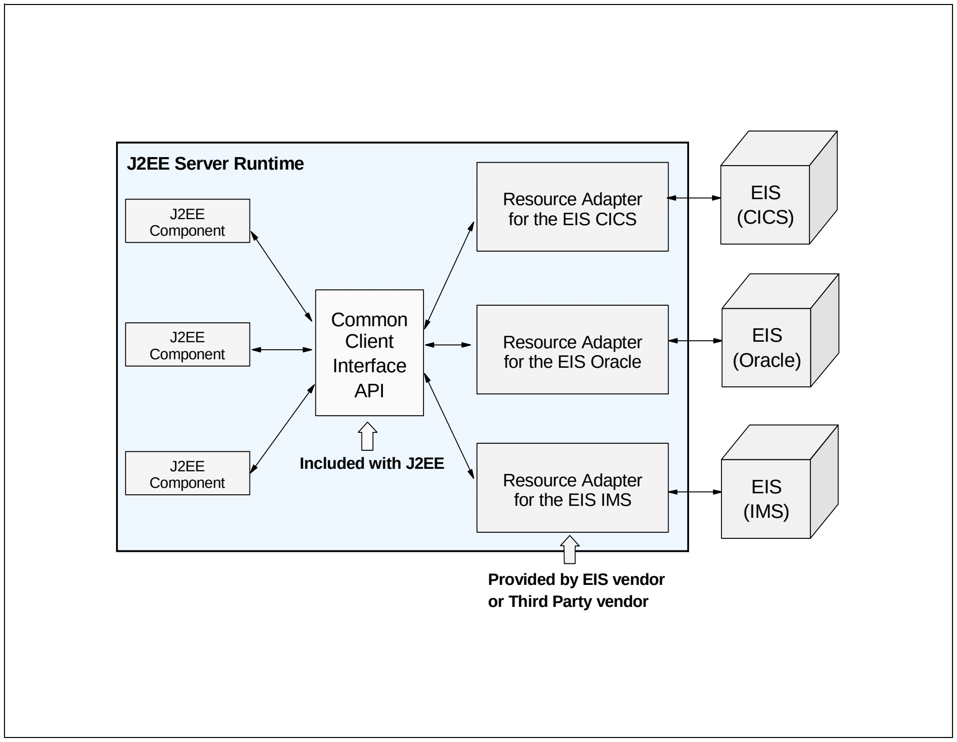

J2EE Server Runtime
J2EE Component
J2EE Component
J2EE Component
Common Client Interface API
Included with J2EE
Resource Adapter for the EIS CICS
Resource Adapter for the EIS Oracle
Resource Adapter for the EIS IMS
Provided by EIS vendor or Third Party vendor
EIS (CICS)
EIS (Oracle)
EIS (IMS)

Figure 18-44 Common Client Interface API

Benefits of the J2EE Connector architecture include:

- Once an application server implements JCA, any JCA-compliant resource adapter can plug in.
- Once a resource adapter implements JCA, it can plug in to any JCA compliant application server.
- Each EIS requires just one implementation of the resource adapter.
- The common client interface simplifies application integration with diverse EISs.

For more information on the J2EE Connector Architecture refer to the J2EE Connector Architecture documentation:

`http://java.sun.com/j2ee/connector/`

18.9 Using JCA resource adapters and connection factories with an application server

In WebSphere Version 5 two type of objects are configured for JCA support:

1. **Resource Adapter**.
2. **Connection Factory**.

The role of the WebSphere administrator is to:

1. Install and define the resource adapter.
2. Define one or more connection factories associated with the resource adapter.

From the application point of view, the application using the resource adapter will request a connection from the connection factory through a JNDI lookup. A code example is shown in Section 18.12, “Using Resource Adapters from an application Code sample” on page 747.The connection factory connects the application to the resource adapter.

18.9.1 Resource adapter

A WebSphere resource adapter administrative object represents the library that supplies implementation code for connecting applications to a specific EIS.

Resource adapters are stored in a Resource Adapter Archive (RAR) file, which is a Java archive (JAR) file used to package a resource adapter for the Connector Architecture. The file has a standard file extension .rar.

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 for developer documentation, not used for runtime.
- Connector architecture (J2C) common client interfaces, such as cci.jar.
- A mandatory deployment descriptor (ra.xml), which instructs the application server about how to use the resource adapter in an application server environment.

The deployment descriptor contains information about the resource adapter, including security and transactional capabilities, and the *ManagedConnectionFactory* class name, see Example 18-6 on page 736.

In WebSphere Version 5, RAR files are installed using the administrative console. However the RAR file or JCA resource adapter needs to be provided by your EIS vendor. The only JCA resource adapter provided by WebSphere is the *WebSphere Relational Resource Adapter*, which is used to connect to relational databases using JDBC.

Author Comment: Any recommendations as to whether the application should be packaged with the rar file or not?

18.9.2 Connection Factory

The WebSphere connection factory administrative object represents the configuration of a specific connection to the EIS supported by the resource adapter. The connection factory can be thought of as a holder of a list of connection configuration properties.

Application components, such as CMP 2.0 enterprise beans, have *cmpConnectionFactory* descriptors that refer to a specific connection factory, not to the resource adapter, see Example 18-5. In this example, the enterprise bean `CustomerAccount` is bound to the `eis/webbankds_CMP` connection factory.

Example 18-5 ibm-ejb-jar-bnd.xmi file for webbankEntityEJBs.jar

```
<ejbBindings
xmi:id="CustomerAccount_Bnd"jndiName="webbank/ejb/CustomerAccount">
    <enterpriseBean xmi:type="ejb:ContainerManagedEntity"
href="META-INF/ejb-jar.xml#CustomerAccount"/>
    <cmpConnectionFactory xmi:id="CMPConnectionFactoryBinding_2"
jndiName="eis/webbankds_CMP" resAuth="Per_Connection_Factory"/>
  </ejbBindings>
```

The `WebBank` application is deployed in Chapter 20, “Deploying an application” on page 835. Refer to it for more information.

18.10 Installing and configuring JCA resource adapters

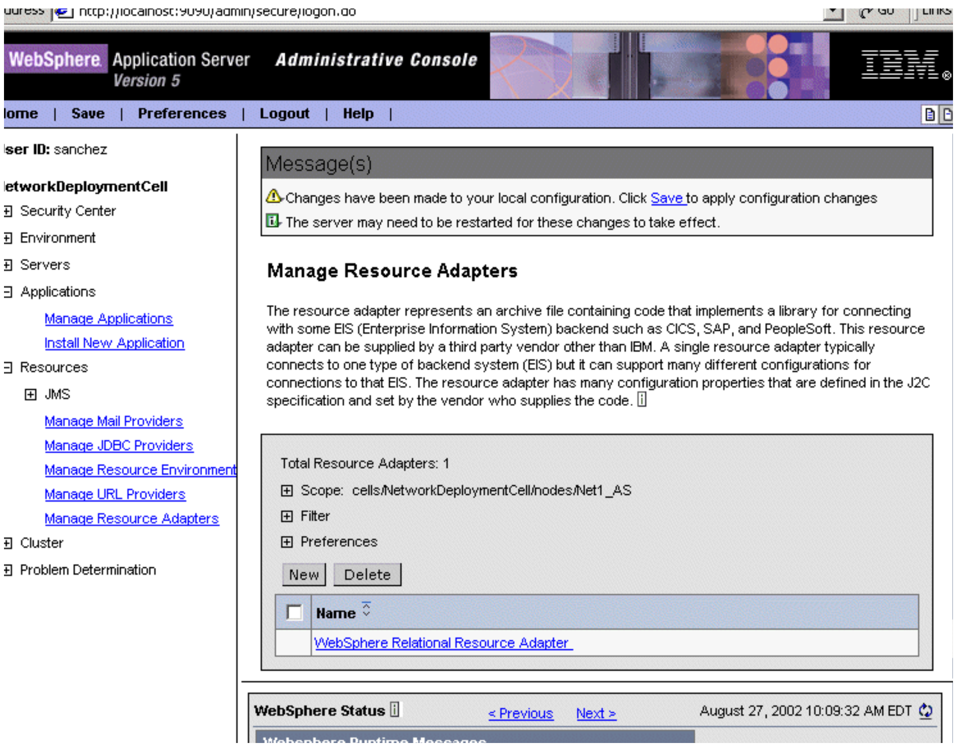
A JCA resource adapter object, represents the library that supplies implementation code for connecting applications to a specific EIS (Enterprise Information System) backend such as CICS or SAP.

To install a JCA resource adapter (.rar file) onto one of the nodes, and configure the JCA resource adapter object, complete the following steps:

1. From the administrative console, expand **`Resources`** from the navigation tree.
2. Click on **`Manage Resource Adapters`**.
3. Select the **`Scope`** and click **`Apply`**.

   The console will show all the JCA resource adapter objects configured at that scope level, see Figure 18-45. In this example we only have one JCA resource adapter configured, the `Built-In WebSphere Relational Resource Adapter`. This adapter is configured at the cell level (*resources.xml*)

Note: The WebSphere Relational Resource Adapter is installed with WebSphere and is used to connect to relational databases using JDBC. See Section , "WebSphere 5 datasource" on page 695.



Message(s)

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

The server may need to be restarted for these changes to take effect.

Manage Resource Adapters

The resource adapter represents an archive file containing code that implements a library for connecting with some EIS (Enterprise Information System) backend such as CICS, SAP, and PeopleSoft. This resource adapter can be supplied by a third party vendor other than IBM. A single resource adapter typically connects to one type of backend system (EIS) but it can support many different configurations for connections to that EIS. The resource adapter has many configuration properties that are defined in the J2C specification and set by the vendor who supplies the code.

Total Resource Adapters: 1

Scope: cells/NetworkDeploymentCell/nodes/Net1_AS

Filter

Preferences

	Name
	WebSphere Relational Resource Adapter

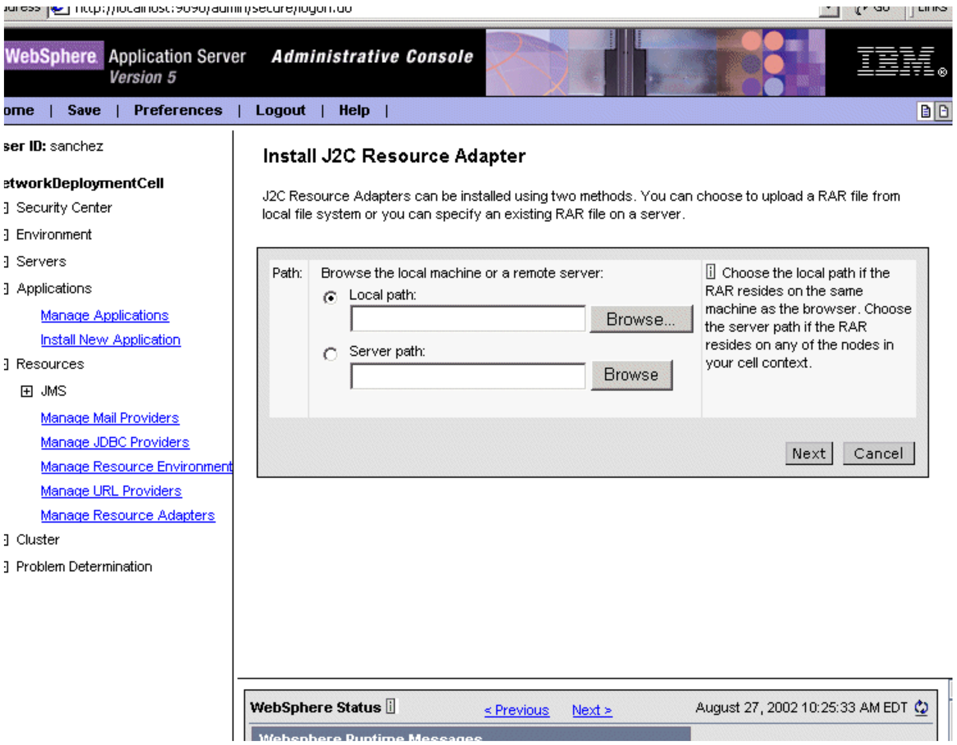
August 27, 2002 10:09:32 AM EDT

Figure 18-45 JCA Resource adapters created at the selected scope

4. Select **New** to install a new JCA resource adapter.
5. From the **Install J2C Resource Adapter** screen, Figure 18-46, give the path to the RAR file containing the resource adapter code. The RAR file will be supplied by your EIS vendor.

   It can reside locally, on the same machine as the browser, or remotely, on any of the nodes in your cell.

   > **Note:** This is where the source RAR file is located. WebSphere will install the RAR file to a different location.



Install J2C Resource Adapter

J2C Resource Adapters can be installed using two methods. You can choose to upload a RAR file from local file system or you can specify an existing RAR file on a server.

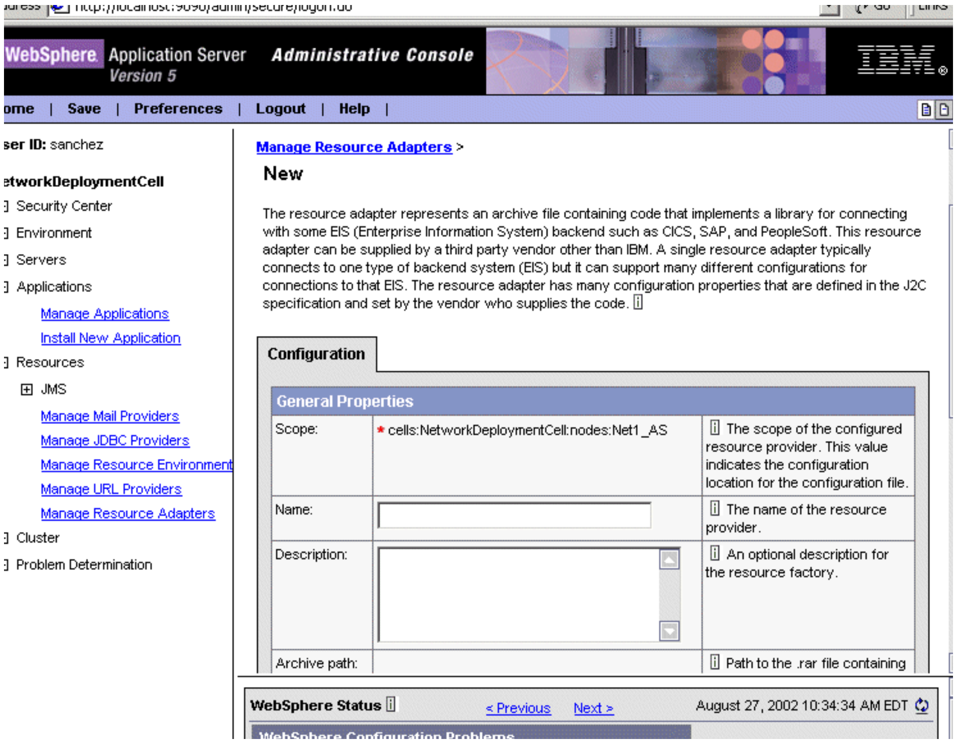
Path: Browse the local machine or a remote server:
[✓ Local path:]
Server path:

Choose the local path if the RAR resides on the same machine as the browser. Choose the server path if the RAR resides on any of the nodes in your cell context.

Figure 18-46 RAR file location

Author Comment: I have seen that in the new built there is an extra setting here: Scope. It appears that you can only install the adapter in a particular node. This is on the Install J2C Resource adapter screen:
<Scope> ..standalone resource adapters are scoped to a particular node, and installed on that node. So you cannot install it to all nodes in a cell for example...This new setting needs to be documented here together with the new screen shot.

6. Click `Next`. The **Configuration** page for the JCA resource adapter selected is displayed. This is shown in Figure 18-47:



Manage Resource Adapters >

New

The resource adapter represents an archive file containing code that implements a library for connecting with some EIS (Enterprise Information System) backend such as CICS, SAP, and PeopleSoft. This resource adapter can be supplied by a third party vendor other than IBM. A single resource adapter typically connects to one type of backend system (EIS) but it can support many different configurations for connections to that EIS. The resource adapter has many configuration properties that are defined in the J2C specification and set by the vendor who supplies the code.

General Properties

Scope:	* cells:NetworkDeploymentCell:nodes:Net1_AS	The scope of the configured resource provider. This value indicates the configuration location for the configuration file.
Name:		The name of the resource provider.
Description:		An optional description for the resource factory.
Archive path:		Path to the .rar file containing

Figure 18-47 JCA resource adapter properties

The properties to be configured are:

a. **Name**. An administrative name for the resource adapter.

b. **Description**. An optional description of the resource adapter, for your administrative records.

c. **Archive path**. The path where the Resource Adapter Archive (RAR) file is going to be installed. If this property is not specified, the archive will be extracted to the absolute path represented by the symbolic name “CONNECTOR_INSTALL_ROOT”, where “CONNECTOR_INSTALL_ROOT” is a variable defined in the *variables.xml* file for the node in which the resource adapter is being installed. The default is <WAS_INSTALL_ROOT>/installedConnectors/<adaptername.rar>

Author Comment: Not sure if the adaptername is the administrative name for the resource adapter or the .rar file name. It is the name of the .rar file in build m0232.11. Need to verify this with later builds.

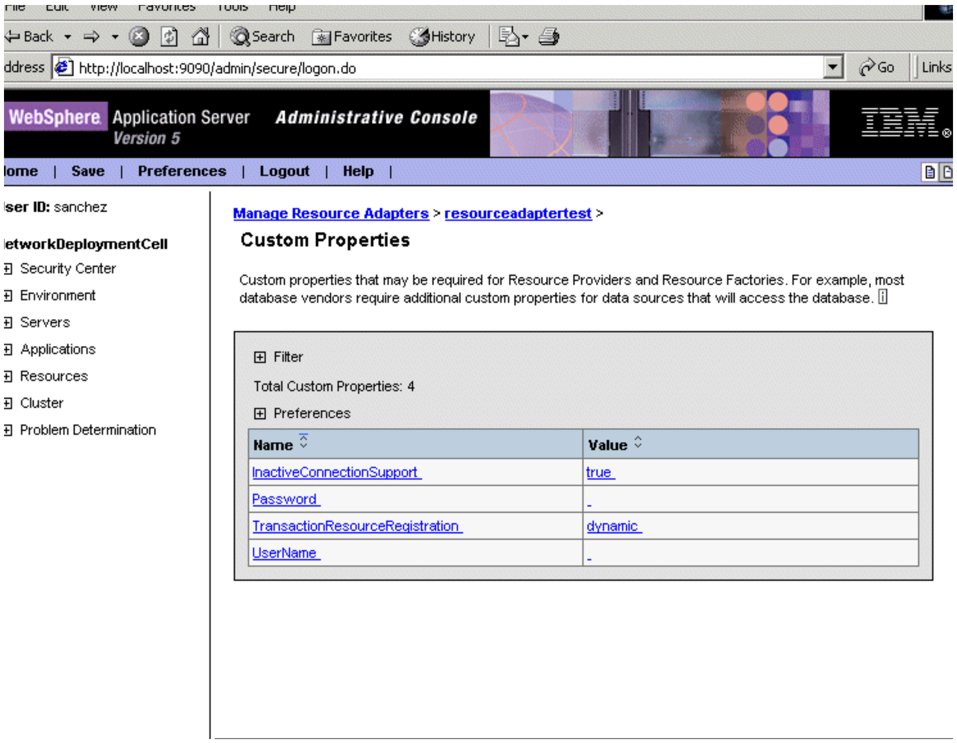
Note: The JCA resource adapter is only installed for the node selected in the Install screen, and only the *resources.xml* file for that node gets updated. The relational resource adapter provided with WebSphere is the only one configured at the cell level (*resources.xml*), and is installed under WAS_INSTALL_ROOT/lib.

   d. **Classpath**. A list of paths or jar file names which together form the location for the resource adapter classes. The resource adapter codebase itself (.rar file) is automatically added to the classpath.

   e. **Native path**. A list of paths which together form the location for the resource adapter native libraries (.dll's, .so's)

7. Click **OK**
8. Select the resource adapter just created.
9. Under **`Additional Properties`**, select **`Custom Properties`**.

   The console page, Figure 18-48, shows configuration properties defined for the resource adapter in the deployment descriptor file, *ra.xml*. The deployment descriptor file is partially shown in Example 18-6.



Manage Resource Adapters > resourceadaptertest >

Custom Properties

Custom properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Filter

Total Custom Properties: 4

Preferences

Name	Value
InactiveConnectionSupport	true
Password	
TransactionResourceRegistration	dynamic
UserName	

Figure 18-48 Resource adapter custom properties

Example 18-6 ra.xml deployment descriptor file for the resource adapter

```
<?xml version="1.0" encoding="UTF-8" ?>
  <!DOCTYPE connector (View Source for full doctype...)>
- <connector>
  <display-name>WS_RdbResourceAdapter</display-name>
  <description>IBM Relational ResourceAdapter</description>
............
............
<resourceadapter>
<managedconnectionfactory-class>com.ibm.ws.rsadapter.spi.WSManagedConnectionFac
toryImpl</managedconnectionfactory-class>
<connectionfactory-interface>javax.resource.cci.ConnectionFactory</connectionfa
ctory-interface>
................
.................

<config-property>
```

```
  <description>Specify whether connection handles support implicit
reactivation. (Smart Handle support). Value may be "true" or
"false".</description>
  <config-property-name>InactiveConnectionSupport</config-property-name>
  <config-property-type>java.lang.Boolean</config-property-type>
  <config-property-value>true</config-property-value>
  </config-property>

- <config-property>
  <description>Type of transaction resource registration (enlistment). Valid
values are either "static" (immediate) or "dynamic" (deferred).</description>
  <config-property-name>TransactionResourceRegistration</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value>dynamic</config-property-value>
  </config-property>
..........
..........
</resourceadapter>
  </connector>
```

10. Click **Save**, to save the configuration.

Note: You need to install and configure the J2C resource adapter on each one of the nodes where the adapter is going to be used to connect to an EIS backend.

18.11 Configuring JCA connection factories

A JCA connection factory represents a set of connection configuration values. Application components such as EJBs have <resource-ref> descriptors that refer to the ConnectionFactory, not the Resource Adapter. The ConnectionFactory is really just a holder of a list of connection 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 ConnectionFactory. These standard properties are used by the JCA connection pool manager in the application server runtime and are not used by the vendor supplied resource adapter code.

To create a JCA connection factory complete the following steps:

1. Expand **Resources** from the navigation tree.
2. Click on **Manage Resource Adapters**.

3. Select the **`Scope`** and click **`Apply`**.

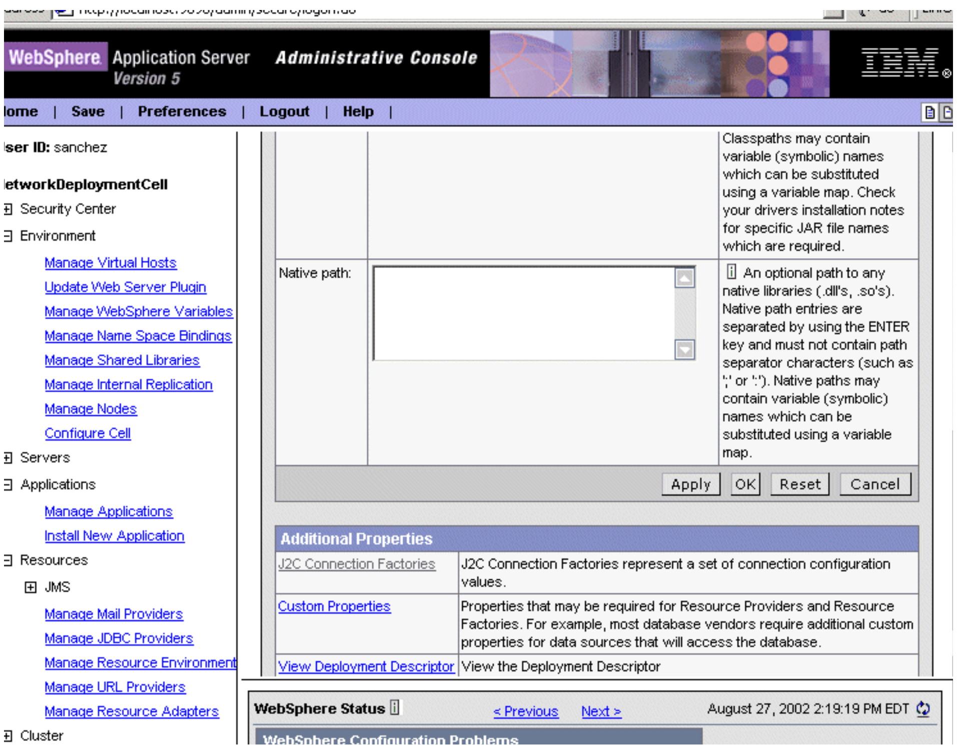
   This shows all the resource adapters created at that scope.

4. Select the resource adapter for which you want to configure a JCA connection factory.

   This brings the Configuration page for the JCA resource adapter.

5. Select **`J2C Connection Factories`** under **`Additional Properties`** as shown in Figure 18-49:

> **Note:** The terms J2C and JCA both refer to J2EE Connector Architecutre and they are used here interchangeably.


WebSphere Application Server Version 5
Administrative Console
IBM
ome | Save | Preferences | Logout | Help |

ser ID: sanchez

etworkDeploymentCell
- Security Center
- Environment
  - Manage Virtual Hosts
  - Update Web Server Plugin
  - Manage WebSphere Variables
  - Manage Name Space Bindings
  - Manage Shared Libraries
  - Manage Internal Replication
  - Manage Nodes
  - Configure Cell
- Servers
- Applications
  - Manage Applications
  - Install New Application
- Resources
  - JMS
  - Manage Mail Providers
  - Manage JDBC Providers
  - Manage Resource Environment
  - Manage URL Providers
  - Manage Resource Adapters
- Cluster

Classpaths may contain variable (symbolic) names which can be substituted using a variable map. Check your drivers installation notes for specific JAR file names which are required.

Native path: An optional path to any native libraries (.dll's, .so's). Native path entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Native paths may contain variable (symbolic) names which can be substituted using a variable map.

Additional Properties	
J2C Connection Factories	J2C Connection Factories represent a set of connection configuration values.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.
View Deployment Descriptor	View the Deployment Descriptor

WebSphere Status < Previous Next > August 27, 2002 2:19:19 PM EDT

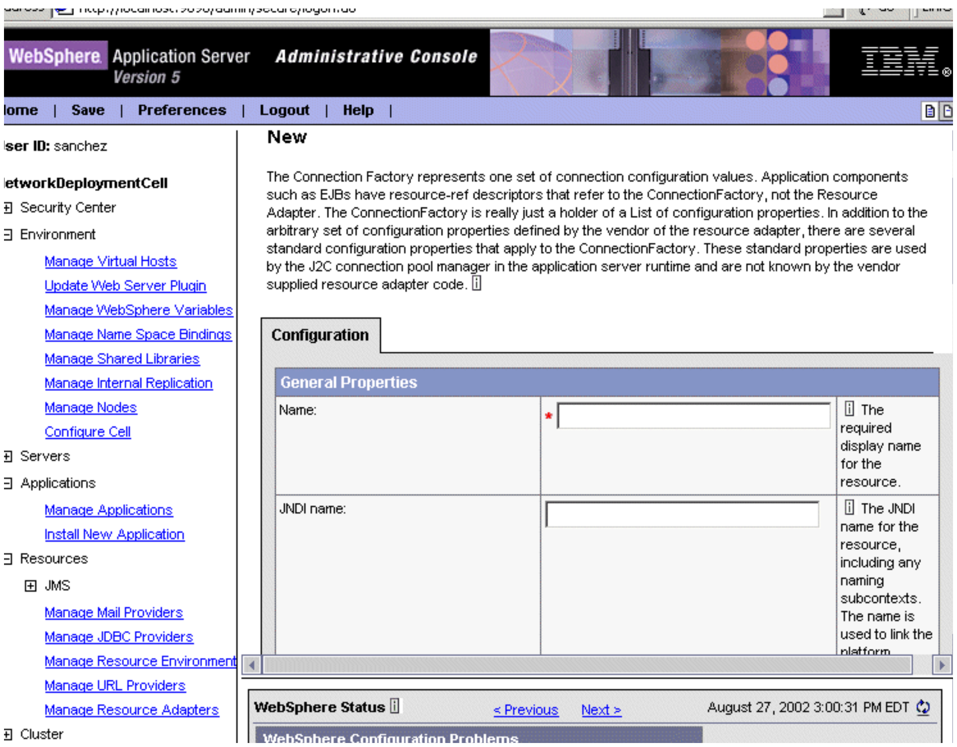
WebSphere Configuration Problems

Figure 18-49 Configurable resources for the JCA connector

The **J2C Connection Factories** page will show all the J2C connection factories configured for that particular JCA resource adapter.

6. Click **`New`** to create a new connection factory, or select an existing one to modify the connection factory properties.

The **J2C Connection Factory Configuration** page is shown in Figure 18-50:



New

The Connection Factory represents one set of connection configuration values. Application components such as EJBs have resource-ref descriptors that refer to the ConnectionFactory, not the Resource Adapter. The ConnectionFactory is really just 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 ConnectionFactory. 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.

Configuration

General Properties

Name:	*	The required display name for the resource.
JNDI name:		The JNDI name for the resource, including any naming subcontexts. The name is used to link the platform

August 27, 2002 3:00:31 PM EDT

Figure 18-50 J2C connection factory properties

The general properties are:

a. **Name**. An administrative name for the J2C connection factory

b. **JNDI name**. The connection factory name to be registered in the application server's name space, including any naming subcontext.

When installing an application that contains modules with J2C resource references, the resources defined by the deployment descriptor of the module need to be bound to the actual JNDI name of the resource.

Tip: As a convention, use the value of the **Name** property prefixed with "eis/" (such as "eis/ConnectionFactoryName").

c. **Description**. An optional description of the J2C connection factory, for your administrative records.

d. **Category**. Specifies a category that you can use to classify or group the connection factory.

e. **Authentication mechanism preference**. Specifies which of the authentication mechanisms that are defined for the corresponding resource adapter applies to this connector factory.

   For example, if two authentication mechanism entries have been defined for a resource adapter (`KerbV5` and `Basic Password`), this will specify one of those two types. If the authentication mechanism preference specified is not an authentication mechanism available on the corresponding resource adapter, it is ignored.

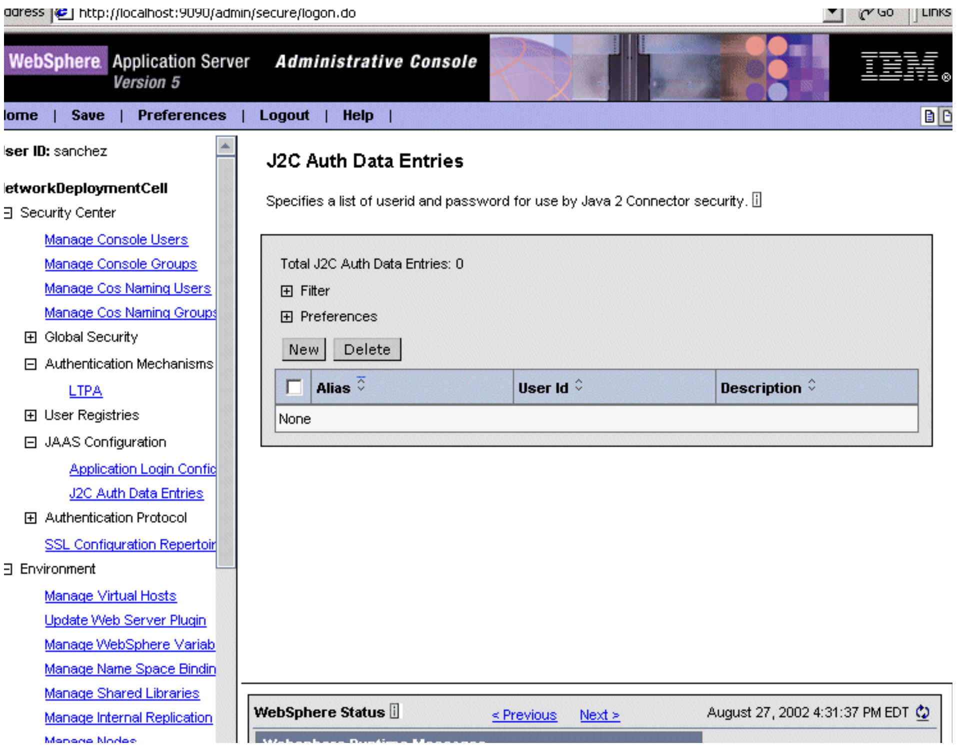
   Note: The authentication mechanisms defined for a resource adapter can be seen in the resource adapter descriptor file (ra.xml). For example:

```
<authentication-mechanism>
<description>BasicPassword authentication</description>

<authentication-mechanism-type>BasicPassword</authentication-mechanism
-type>

<credential-interface>javax.resource.spi.security.PasswordCredential</
credential-interface>
  </authentication-mechanism>

- <authentication-mechanism>
  <description>Kerbv5 Authentication</description>

<authentication-mechanism-type>Kerbv5</authentication-mechanism-type>

<credential-interface>javax.resource.spi.security.GenericCredential</c
redential-interface>
  </authentication-mechanism>
```

f. **Container-managed authentication alias**. The authentication alias used for container-managed signon to the resource. The alias references a J2C Authentication Data Entry from the list of authDataEntries in *security.xml*.

   To create a J2C Authentication Data Entry:

   i. Select **Security** -> **JAAS Configuration** -> **J2C Auth Data Entries** from the administrative console, as shown in Figure 18-51.



J2C Auth Data Entries

Specifies a list of userid and password for use by Java 2 Connector security.

Total J2C Auth Data Entries: 0

Filter

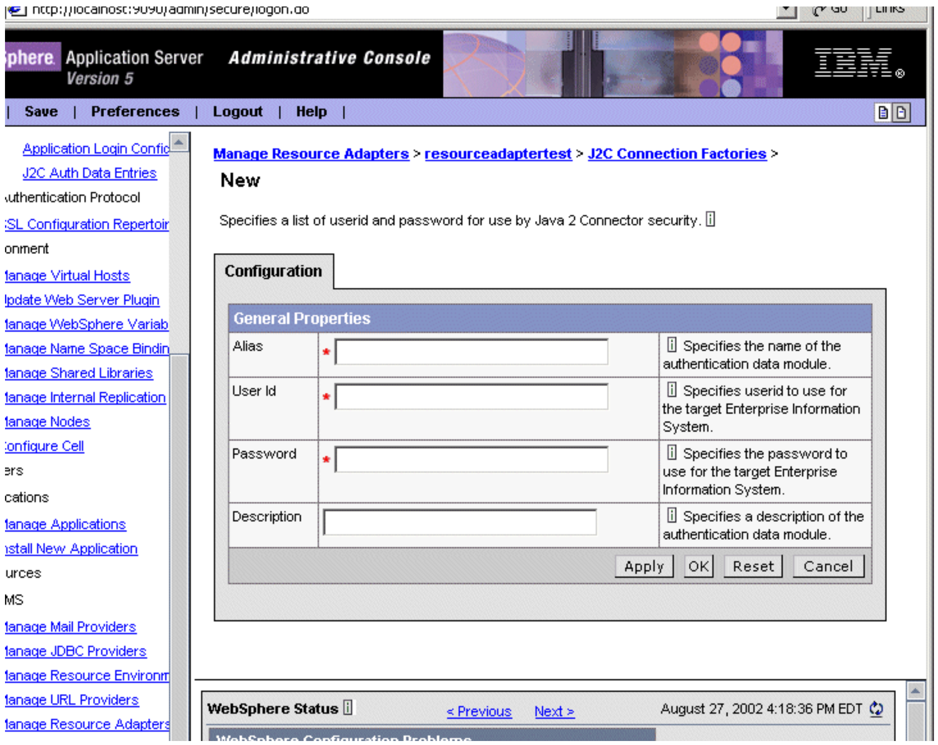
Preferences

	Alias	User Id	Description
None			

Figure 18-51 J2C authentication data

ii. Select **New** to create a new J2C authentication data entry.

iii. Specify <alias>, <userid> and <password> to be used by Java 2 connector security. This is shown in Figure 18-52:



Manage Resource Adapters > resourceadaptertest > J2C Connection Factories >

New

Specifies a list of userid and password for use by Java 2 Connector security.

Configuration

General Properties		
Alias	*	Specifies the name of the authentication data module.
User Id	*	Specifies userid to use for the target Enterprise Information System.
Password	*	Specifies the password to use for the target Enterprise Information System.
Description		Specifies a description of the authentication data module.

August 27, 2002 4:18:36 PM EDT

Figure 18-52 Configuration page for J2C authentication data entries

g. **Component-managed authentication alias**. The authentication alias used for component-managed signon to the resource. The alias references a J2C Authentication Data Entry from the list of authDataEntries in *security.xml*.

   See **Container-managed authentication alias** above, for information on how to create J2C authentication data entries.

7. Click **OK**
8. Click **Save**, to save the configuration.

18.11.1 Configuring connection pooling for a J2C connection factory

Connection pool properties affect the behavior of the J2C connection manager. Though default values are provided, it is recommended that you review these settings so ensure they are appropriate to your production environment.

To configure the connection pool for a J2C connection factory, complete the following steps:

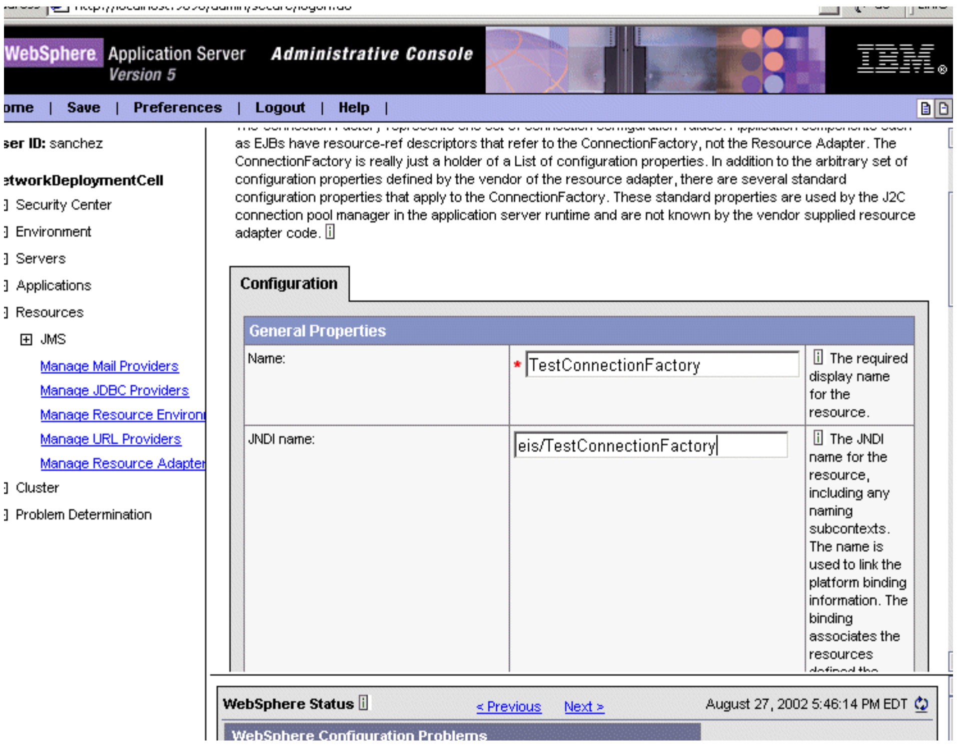
1. Expand **`Resources`** from the navigation tree.
2. Click on **`Manage Resource Adapters`**.
3. Select the **`Scope`** and click **`Apply`**.

   This shows all the JCA resource adapters created at that scope.
4. Select the JCA resource adapter used with your J2C connection factory.

   This brings the Configuration panel for that JCA resource adapter.
5. Select **J2C Connection Factories** under **`Additional Properties`**

   The content pane will show all the J2C connection factories configured for that J2C resource adapter.
6. Select the J2C connection factory for which you want to configure connection pooling.

   This brings the Configuration page for the selected J2C connection factory, as shown in Figure 18-53.



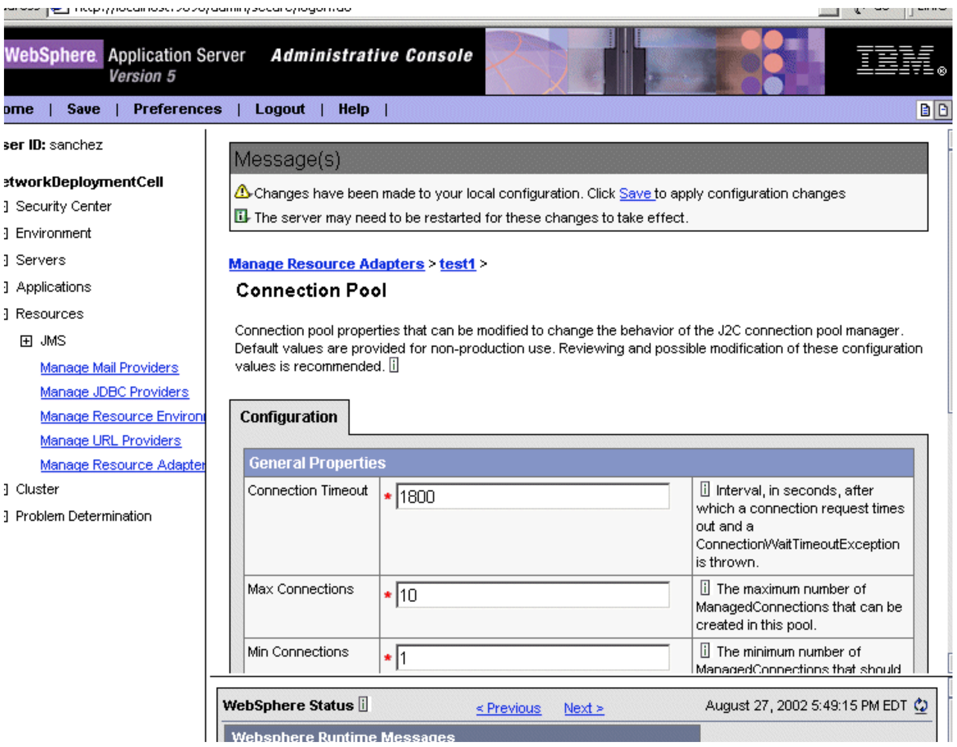
as EJBs have resource-ref descriptors that refer to the ConnectionFactory, not the Resource Adapter. The ConnectionFactory is really just a holder of a List of configuration properties. In addition to the arbitrary set of configuration properties defined by the vendor of the resource adapter, there are several standard configuration properties that apply to the ConnectionFactory. 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.

Name:	* TestConnectionFactory	The required display name for the resource.
JNDI name:	eis/TestConnectionFactory	The JNDI name for the resource, including any naming subcontexts. The name is used to link the platform binding information. The binding associates the resources

Figure 18-53 J2C connection factory configuration page

7. Under **`Additional Properties`**, select **`Connection Pool`**

   The properties to configure for a J2C connection factory connection pool are as shown in Figure 18-54:



Message(s)

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

The server may need to be restarted for these changes to take effect.

Manage Resource Adapters > test1 >

Connection Pool

Connection pool properties that can be modified to change the behavior of the J2C connection pool manager. Default values are provided for non-production use. Reviewing and possible modification of these configuration values is recommended.

Connection Timeout	* 1800	Interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.
Max Connections	* 10	The maximum number of ManagedConnections that can be created in this pool.
Min Connections	* 1	The minimum number of ManagedConnections that should

August 27, 2002 5:49:15 PM EDT

Figure 18-54 J2C connection factory connection pool properties

   a. **Connection Timeout**

      Specifies the interval, in seconds, after which a connection request times out and a *ConnectionWaitTimeoutException* is thrown. This can occur when the pool is at its maximum (**Max Connections**) and all of the connections are in use by other applications for the duration of the wait.

Author Comment: I am not sure whether it throws a ResourceAllocationException or a ConnectionWaitTimeoutException. If so it may need to be corrected for both the connection pool in WAS5 and here...

For example, if Connection Timeout is set to 300 and the maximum number of connections is reached, the Pool Manager waits for 300 seconds for an available physical connection. If a physical connection is not available within this time, the Pool Manager throws a *ConnectionWaitTimeoutException*.

Tip: If Connection Timeout is set to 0, the Pool Manager waits as long as necessary until a connection is allocated.

b. **Max Connections**

Specifies the maximum number of ManagedConnections that can be created in this pool.

These are the physical connections to the backend database. Once this number is reached, no new physical connections are created and the requester waits until a physical connection that is currently in use is returned to the pool or a *ConnectionWaitTimeoutException* is thrown.

For example, if Max Connections is set to 5, and there are 5 physical connections in use, the Pool Manager waits for the amount of time specified in Connection Timeout for a physical connection to become free. If after that time there are still no free connections, the Pool Manager throws a *ConnectionWaitTimeoutException* to the application.

Author Comment: See comment above about the actual exception

c. **Min Connections**

Specifies the minimum number of ManagedConnections (physical connections) to be maintained.

Until this number is reached, the pool maintenence thread does not discard any physical connections. However, no attempt is made to bring the number of connections up to this number.

For example if Min Connections is set to 3, and one physical connection is created, that connection is not discarded by the **Unused Timeout** thread.

By the same token, the thread does not automatically create two additional physical connections to reach the Min Connections setting.

d. **Reap Time**

Specifies the interval, in seconds, between runs of the pool maintenence thread.

For example, if Reap Time is set to 60, the pool maintenence thread runs every 60 seconds. The Reap Time interval affects the accuracy of the **Unused Timeout** and **Aged Timeout** settings. The smaller the interval, the greater the accuracy. When the pool maintenance thread runs it discards any connections that have been unused for longer than the time value specified in Unused Timeout, until it reaches the number of connections specified in Min Connections. The pool maintenence thread also discards any connections that remain active longer than the time value specified in Aged Timeout.

Tip: If the pool maintenence thread is enabled, set the Reap Time value less than the values of Unused Timeout and Aged Timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenence thread runs more often and degrades performance.

e. **Unused Timeout**

Specifies the interval in seconds after which an unused or idle connection is discarded.

Tip: Set the Unused Timeout value higher than the Reap Timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections not in use exceeds the Min Connections setting.

For example, if the unused timeout value is set to 120, and the pool maintenence thread is enabled (Reap Time is not 0), any physical connection that remains unused for two minutes is discarded. Note that accuracy of this timeout, as well as performance, is affected by the Reap Time value. See Reap Time for more information.

f. **Aged Timeout**

Specifies the interval in seconds before a physical connection is discarded, regardless of recent usage activity.

Setting Aged Timeout to 0 allows active physical connections to remain in the pool indefinitely. For example if the Aged Timeout value is set to 1200, and the Reap Time value is not 0, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that accuracy of this timeout, as well as performance, is affected by the Reap Time value. See Reap Time for more information.

Tip: Set the Aged Timeout value higher than the Reap Timeout value for optimal performance.

g. **Purge Policy**

Specifies how to purge connections when a stale connection or fatal connection error is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**. If EntirePool is specified all physical connections in the pool are destroyed when a stale connection is detected. If FailingConnectionOnly is specified the pool attempts to destroy only the stale connection, the other connections remain in the pool. Final destruction of connections which are in use at the time of the error might be delayed. However, those connections are never returned to the pool.

8. Click **OK**, when finished configuring the connection pool.
9. Click **Save**, to save the configuration.

18.12 Using Resource Adapters from an application Code sample

The following code example shows how you might access an Oracle database via the resource adapter.

Author Comment: The example has been taken from http://www.datadirect-technologies.com/download/docs/jdbc/jdbcref/usejdbc.htm#1003516 ...therefore uses datadirect's jca software... An IBM specific example code is required.

Example 18-7 Using resource adapters from an application. Code sample

```
import java.util.Hashtable;
import java.sql.Connection;
import javax.sql.DataSource;
import javax.naming.*;

import com.ddtek.resource.jdbc.JCAConnectionFactory;
import com.ddtek.resource.jdbc.spi.*;

public class RAExample {

    static public void main(String[] args) {
        try {

           //  Create a connection factory instance
          OracleManagedConnectionFactory  managedFactory =
              new OracleManagedConnectionFactory();

          managedFactory.setServerName("MyOracleServer");
           managedFactory.setPortNumber("1521");
            managedFactory.setSID("TEST");

            JCAConnectionFactory factory = (JCAConnectionFactory)
                   ManagedFactory.createConnectionFactory();

            // Get an InitialContext.  Using File System JNDI Service
            // Provider as an example
            Hashtable   env = new Hashtable();

            env.put(Context.INITIAL_CONTEXT_FACTORY,
           "com.sun.jndi.fscontext.RefFSContextFactory");
              env.put(Context.PROVIDER_URL,
  "file:c:/ConnectionFactories");

            Context connectorContext = new InitialContext(env);

            //  Bind the connection factory
            try {
                 connectorContext.bind("OracleConnectionFactory",
  factory);
            }
            catch (NameAlreadyBoundException except) {
                 connectorContext.rebind("OracleConnectionFactory",
                           factory);
             }
         }
         catch (Exception except) {
             System.out.println("Error creating DataSource");
          System.exit(0);
         }

         // Connect via the DataSource

         try {

             // Get an InitialContext.  Using File System JNDI Service
          // Provider as an example
          Hashtable   env = new Hashtable();

          env.put(Context.INITIAL_CONTEXT_FACTORY,
     "com.sun.jndi.fscontext.RefFSContextFactory");
          env.put(Context.PROVIDER_URL,
     "file:c:/ConnectionFactories");

          Context connectorContext = new InitialContext(env);

          //  Lookup the connection factory
          DataSource dataSource = (DataSource)
       connectorContext.lookup("OracleConnectionFactory");
          Connection connection =
              dataSource.getConnection("scott", "tiger");
         }
         catch (Exception except) {
             System.out.println("Looking up connection factory");
         }
       }
 }
```

18.12.1 More information

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

- WebSphere InfoCenter. Search on JCA.

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

- J2EE Connector Architecture documentation

  `http://java.sun.com/j2ee/connector`

18.13 JavaMail and JavaMail service providers

The **JavaMail** APIs provide a platform and protocol-independent framework for building Java-based mail client applications.

The JavaMail APIs are generic for sending and receiving mail. They require **Service Providers**, known in WebSphere as **Protocol providers**, to interact with mail servers that run the pertaining protocols.

A **JavaMail provider** encapsulates a collection of protocol providers. WebSphere Application Server has a **Built-in Mail Provider** that encompasses three protocol providers: SMTP, IMAP and POP3. These protocol providers are installed as the default and should be sufficient for most applications.

1. **Simple Mail Transfer Protocol (SMTP)**. This is a popular transport protocol for sending mail. JavaMail applications can connect to an SMTP server and send mail through it by using this SMTP protocol provider.
2. **Post Office Protocol (POP3)**. This is the standard protocol for receiving mail.
3. **Internet Message Access Protocol (IMAP)**. This is an alternative protocol to POP3 for receiving mail.

   To use other protocols, you must install the appropriate service provider for those protocols.

Note: In this section the terms JavaMail provider and mail provider are used interchangeably.

In addition to service providers, JavaMail requires the **Java Activation Framework (JAF)** as the underlying framework to deal with complex data types that are not plain text, like `Multipurpose Internet Mail Extensions` (`MIME`), `Uniform Resource Locator` (`URL`) pages, and file attachments.

The JavaMail APIs, the JAF, the service providers and the protocols are shipped as part of WebSphere Application Server using the following Sun licensed packages:

- **mail.jar**. Contains the JavaMail APIs, and the SMTP, IMAP, and POP3 service providers.
- **activation.jar**. Contains the JavaBeans Activation Framework.

These JAR files can be found in <WAS_INSTALL_ROOT>/java/jre/lib/ext directory along with all the other Java extension packages provided with WebSphere.

Figure 18-55 illustrates the relationship among the different JavaMail components:

- JavaMail APIs
- JavaBeans Activation Framework
- Service providers
- Mail protocols

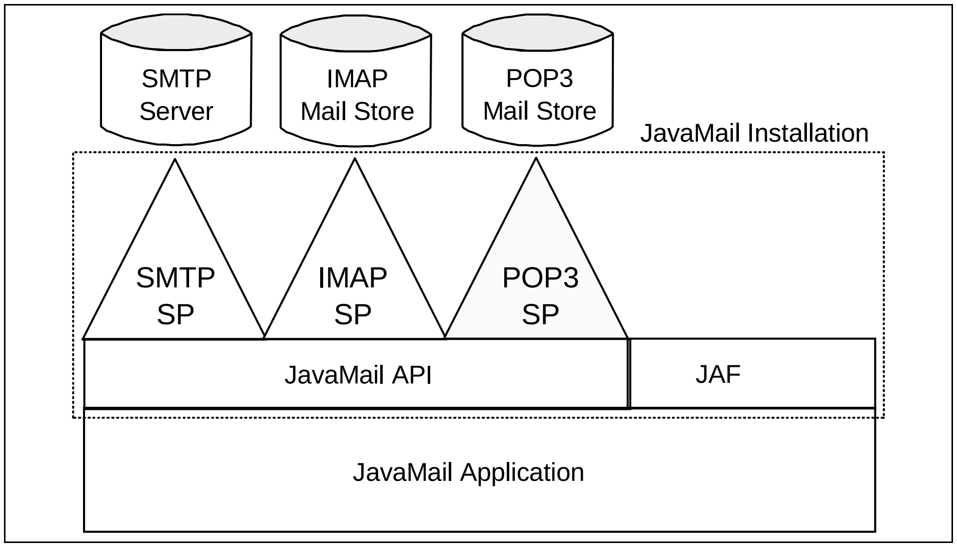


Figure 18-55 JavaMail components

WebSphere Application Server Version 5 supports JavaMail Version 1.2 and the JavaBeans Activation Framework (JAF) Version 1.0. All Web components of WebSphere (servlets, JSPs, EJBs, and application clients), support JavaMail.

18.14 JavaMail sessions

A JavaMail session object (or Session administrative object) is a resource used by the application to obtain connections to a mail server. A mail session object manages the configuration options and user authentication information used to interact with the mail system. Refer to the JavaMail API Design Specification, Version 1.2 document for more information on the *Session* class:

```
http://java.sun.com/products/javamail/JavaMail-1.2.pdf
```

JavaMail sessions are configured to use a particular JavaMail provider.

18.15 Configuring JavaMail resources

In this section we describe how to configure the following types of JavaMail resources from the WebSphere administrative console:

1. Mail Provider (and the protocol providers that come with it)
2. Mail session

18.15.1 Configuring the mail provider

As described in Section 18.13, “JavaMail and JavaMail service providers” on page 750, a mail provider encapsulates a collection of protocol providers. Protocol providers interact with JavaMail APIs and mail servers running those protocols.

WebSphere Application Server has a **Built-in Mail Provider** that encompasses three protocol providers: SMTP, IMAP and POP3. These protocol providers are installed by default and should be sufficient for most applications.

However you can configure a new provider if necessary. Refer to the JavaMail API Design Specification, Version 1.2 document for more information on how service providers (mail providers) can register their protocol implementations so they can be used by JavaMail APIs:

`http://java.sun.com/products/javamail/JavaMail-1.2.pdf`

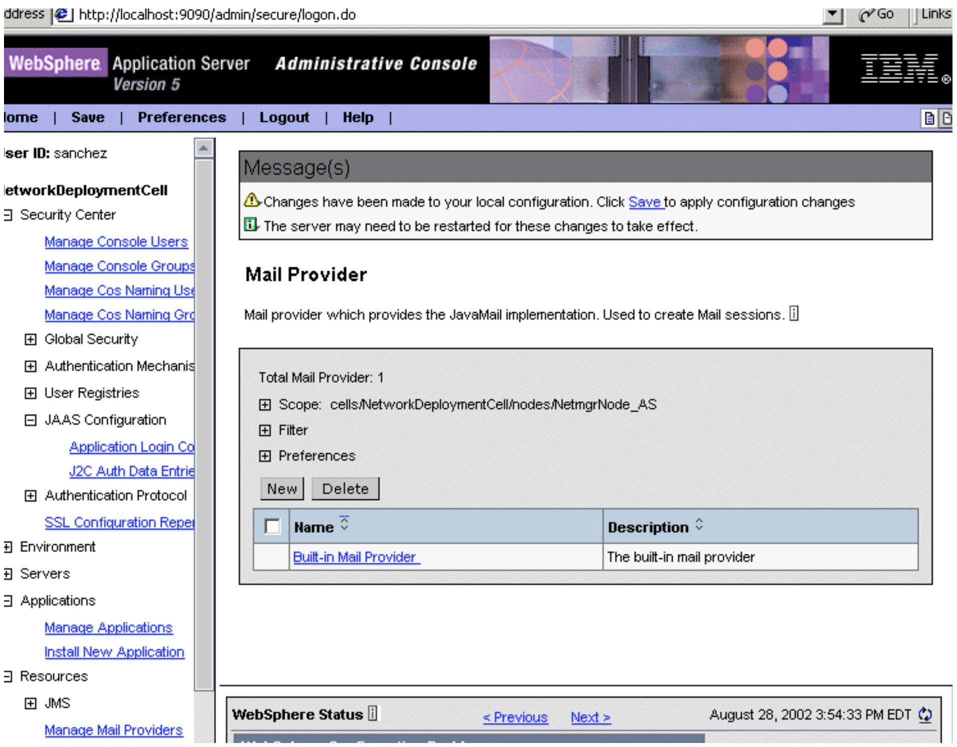
To configure a new mail provider with WebSphere complete the following steps from the administrative console:

1. Expand **`Resources`** from the navigation tree.
2. Click on **`Manage Mail Providers`**.
3. Select the **`Scope`** and click **`Apply`**.

   The scope determines whether JavaMail resources configured to use this provider will be available at the cell, node or the application server level.

   Note: The **Built-in Mail Provider** is configured at the cell level (*resources.xml*). Resources configured for this provider will be available to all the application servers in the cell.

   Figure 18-56, shows that there is one mail provider installed with WebSphere. This is the *Built-in Mail Provider*:


ddress http://localhost:9090/admin/secure/logon.do

WebSphere Application Server Version 5 Administrative Console

lome | Save | Preferences | Logout | Help |

ser ID: sanchez

etworkDeploymentCell
- Security Center
  - Manage Console Users
  - Manage Console Groups
  - Manage Cos Naming Use
  - Manage Cos Naming Gro
  - Global Security
  - Authentication Mechanis
  - User Registries
  - JAAS Configuration
    - Application Login Co
    - J2C Auth Data Entrie
  - Authentication Protocol
  - SSL Configuration Reper
- Environment
- Servers
- Applications
  - Manage Applications
  - Install New Application
- Resources
  - JMS
  - Manage Mail Providers

Message(s)

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

The server may need to be restarted for these changes to take effect.

Mail Provider

Mail provider which provides the JavaMail implementation. Used to create Mail sessions.

Total Mail Provider: 1

Scope: cells/NetworkDeploymentCell/nodes/NetmgrNode_AS

Filter

Preferences

New Delete

	Name	Description
	Built-in Mail Provider	The built-in mail provider

WebSphere Status < Previous Next > August 28, 2002 3:54:33 PM EDT

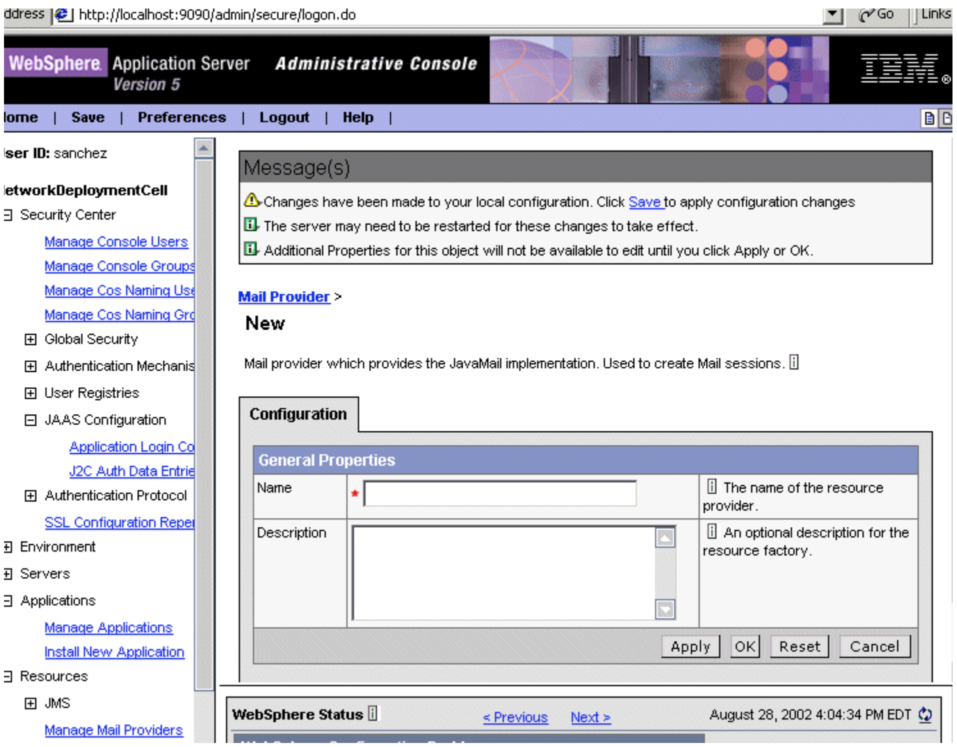
Figure 18-56 Mail Provider page

4. Click **New** to configure a new mail provider, or select an existing one to edit it.

   The **General Configuration properties** required to configure a new mail provider are shown in Figure 18-57. These properties are:

   a. **Name**. An administrative name for the mail provider.

   b. **Description**. An optional description of the mail provider, for your administrative records.



Message(s)

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

The server may need to be restarted for these changes to take effect.

Additional Properties for this object will not be available to edit until you click Apply or OK.

Mail Provider >

New

Mail provider which provides the JavaMail implementation. Used to create Mail sessions.

Configuration

General Properties

Name	*	The name of the resource provider.
Description		An optional description for the resource factory.

Apply OK Reset Cancel

WebSphere Status < Previous Next > August 28, 2002 4:04:34 PM EDT

Figure 18-57 Mail Provider General Properties

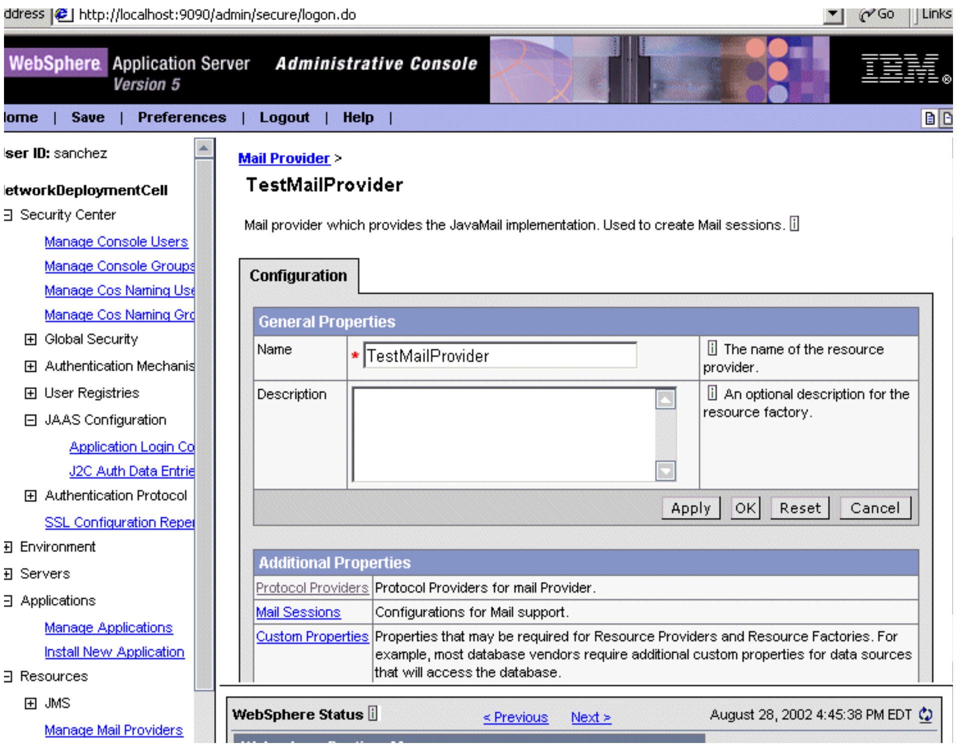
5. Click **OK**.
6. Click **Save**, to save the configuration.

Configuring Protocol Providers

To configure which protocol providers are going to be implemented by the mail provider, complete the following steps from the administrative console:

1. Expand **Resources** from the navigation tree.
2. Click on **Manage Mail Providers**.
3. Select the **Scope** and click **Apply**.

   This will show all the mail providers created at that scope.
4. Select the mail provider that will be implementing the protocol provider.
5. On the **Configuration** page for the mail provider, select **Protocol Providers** under **Additional Properties**, see Figure 18-58:



Mail Provider >

TestMailProvider

Mail provider which provides the JavaMail implementation. Used to create Mail sessions.

Configuration

General Properties

Name	* TestMailProvider	The name of the resource provider.
Description		An optional description for the resource factory.

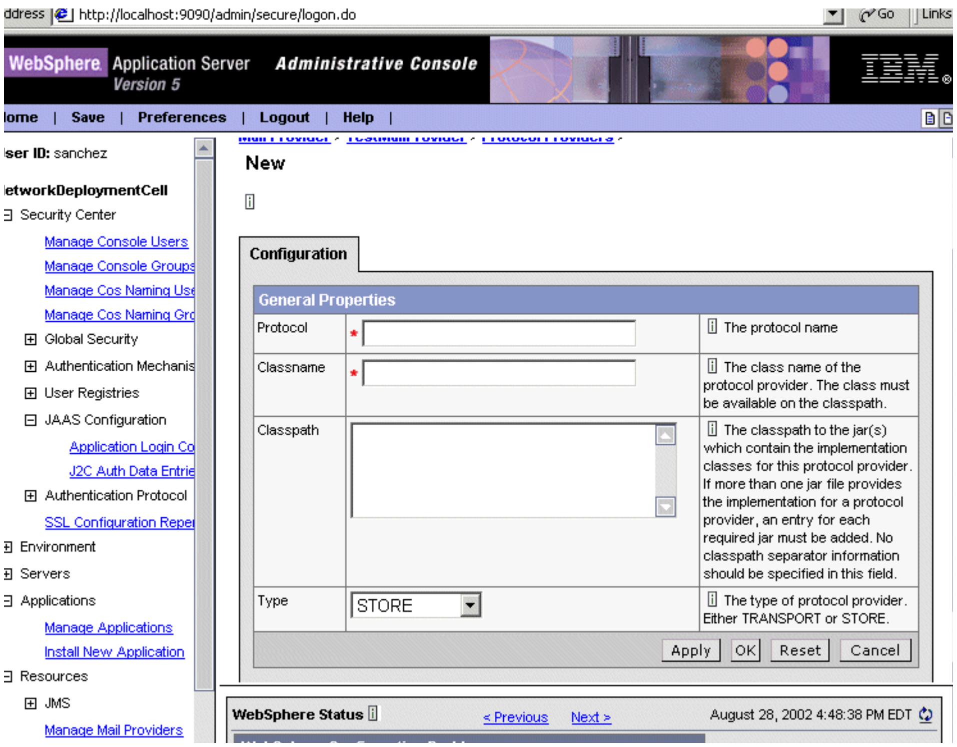
Additional Properties

Protocol Providers	Protocol Providers for mail Provider.
Mail Sessions	Configurations for Mail support.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Figure 18-58 Mail provider configuration page

6. Select **`New`** on the **Protocol Providers** page, or select an existing one to edit it.

   This brings the **Configuration Properties page** for the Protocol provider, as shown in Figure 18-59.



Protocol	*	The protocol name
Classname	*	The class name of the protocol provider. The class must be available on the classpath.
Classpath		The classpath to the jar(s) which contain the implementation classes for this protocol provider. If more than one jar file provides the implementation for a protocol provider, an entry for each required jar must be added. No classpath separator information should be specified in this field.
Type	STORE	The type of protocol provider. Either TRANSPORT or STORE.

Figure 18-59 Protocol provider configuration page

The properties to configure are:

a. **Protocol**. Specifies the protocol name.

b. **Classname**. Specifies the implementation class for the specific protocol provider. The class must be available in the classpath.

c. **Classpath**. Specifies the path to the JAR files that contain the implementation classes for this protocol provider.

d. **Type**. Specifies the type of protocol provider. Valid options are:

   i. STORE. The protocol is used for receiving mail (Message Access Protocol).

   ii. TRANSPORT. The protocol is used for sending mail (Transport Protocol).

Table 18-2 shows the list of protocol providers implemented with the Built-In Mail Provider:

Table 18-2 Protocol providers implemented with the Built-In Mail Provider

Protocol	Classname	Description	Type
imap	com.sun.mail.imap .IMAPStore	Provides access to an IMAP message store	STORE
pop3	com.sun.mail.pop3 .POP3Store	Provides access to a POP3 message store	STORE
smtp	com.sun.mail.smtp .SMTPTransport	Provides access to a POP3 message store	TRANSPORT

Refer to the JavaMail API documentation for more information on the above protocol providers:

`http://java.sun.com/products/javamail/1.3/docs/javadocs/`

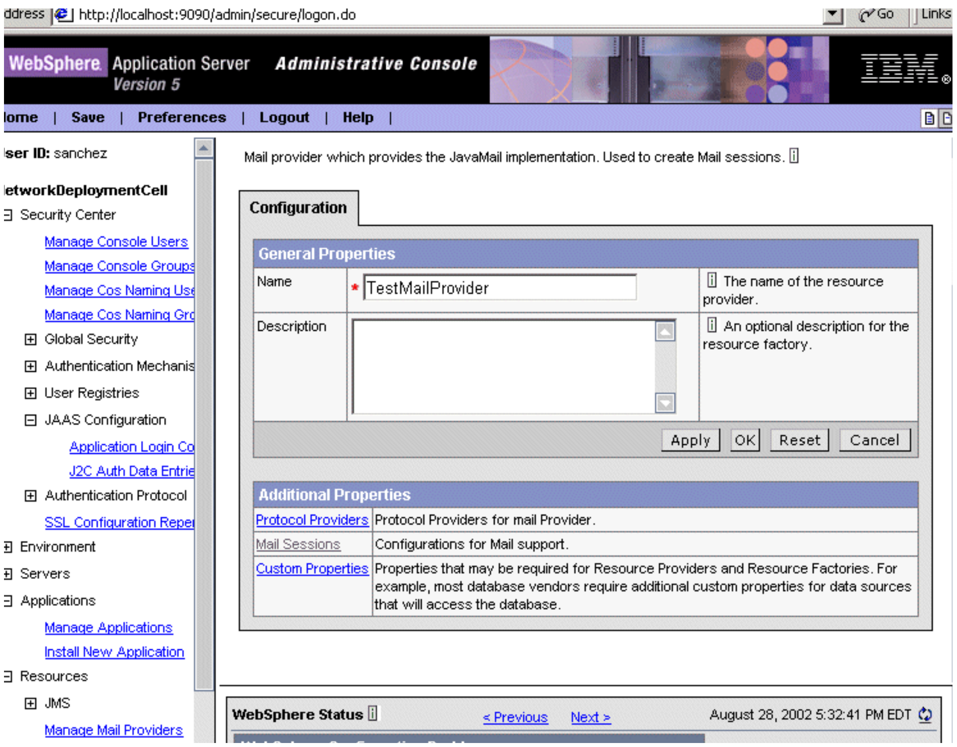
7. Click **`OK`**.
8. Click **`Save`**, to save the configuration.

18.15.2 Configuring JavaMail sessions

To configure JavaMail sessions with a particular mail provider complete the following steps from the administrative console:

1. Expand **`Resources`** from the navigation tree.
2. Click on **`Manage Mail Providers`**.
3. Select the **`Scope`** and click **`Apply`**.

   This will show all the mail providers created at that scope.
4. Select the mail provider used by the JavaMail session.
5. On the mail provider **Configuration** page, under **`Additional Properties`**, select **`Mail Sessions`**, see Figure 18-60:



Mail provider which provides the JavaMail implementation. Used to create Mail sessions.

General Properties

Name	* TestMailProvider	The name of the resource provider.
Description		An optional description for the resource factory.

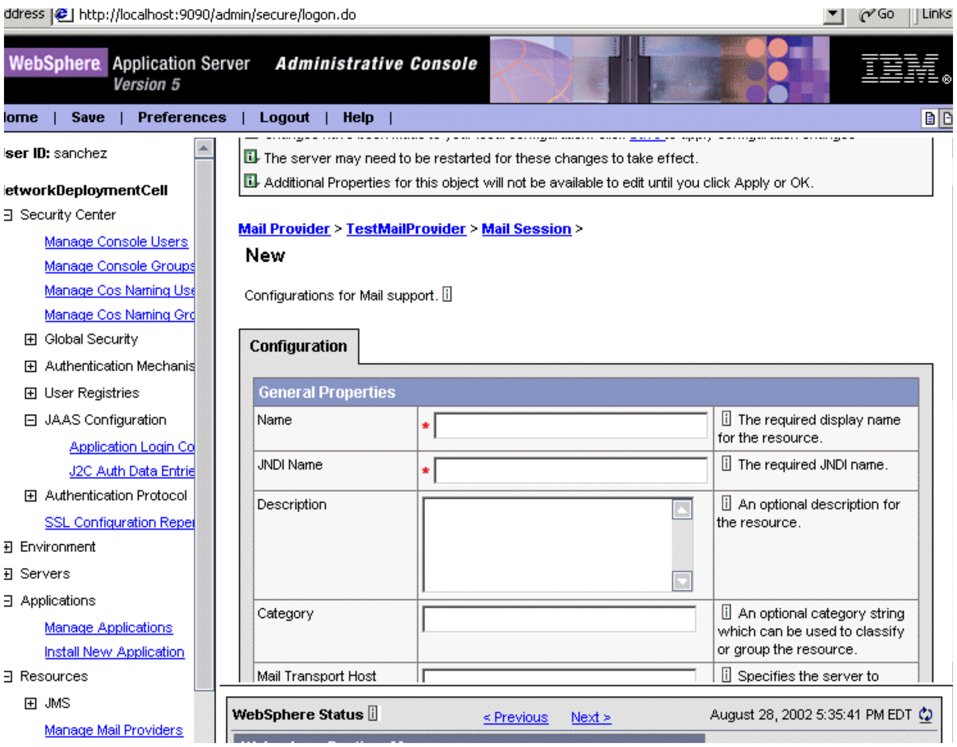
Additional Properties

Protocol Providers	Protocol Providers for mail Provider.
Mail Sessions	Configurations for Mail support.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Figure 18-60 Mail provider configuration page

6. On the **Mail Session page**, select **`New`** to create a new mail session object, or select an existing one to update it.

   This brings the **Configuration Properties page** for the Mail session, see Figure 18-61.


ddress http://localhost:9090/admin/secure/logon.do

WebSphere Application Server Version 5 Administrative Console

lome | Save | Preferences | Logout | Help |

ser ID: sanchez

etworkDeploymentCell

Security Center

Manage Console Users
Manage Console Groups
Manage Cos Naming Use
Manage Cos Naming Grc
Global Security
Authentication Mechanis
User Registries
JAAS Configuration
Application Login Co
J2C Auth Data Entrie
Authentication Protocol
SSL Configuration Reper
Environment
Servers
Applications
Manage Applications
Install New Application
Resources
JMS
Manage Mail Providers

The server may need to be restarted for these changes to take effect.
Additional Properties for this object will not be available to edit until you click Apply or OK.

Mail Provider > TestMailProvider > Mail Session >

New

Configurations for Mail support.

Configuration

General Properties

Name	*	The required display name for the resource.
JNDI Name	*	The required JNDI name.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Mail Transport Host		Specifies the server to

WebSphere Status < Previous Next > August 28, 2002 5:35:41 PM EDT

Figure 18-61 Configuration page for the mail session

The properties to configure are:

a. **Name**. The administrative name of the JavaMail session object.

b. **JNDI Name**. The JavaMail mail session object name as registered in the application servers name space, including any naming subcontext.

When installing an application that contains modules with JavaMail resource references, the resources defined by the deployment descriptor of the module need to be bound to the actual JNDI name of the resources.

Tip: As a convention, use the value of the **Name** property prefixed with “mail/” (such as “mail/mailsessionName”).

c. **Description**. An optional description of the JavaMail session object, for your administrative records.

d. **Category**. Specifies a category that you can use to classify or group the JavaMail session.

e. **Mail Transport Host**. Specifies the server to connect to when sending mail. Specify the fully qualified Internet host name of the mail server.

f. **Mail Transport Protocol**. Specifies the transport protocol to use when sending mail. It could be SMTP or any transport protocol for which the user has installed a provider.

g. **Mail Transport userid**. Specifies the user ID to provide when connecting to the mail transport host. This setting is not generally used for most mail servers. Leave this field blank unless you use a mail server that requires a user ID and password.

h. **Mail Transport password**. Specifies the password to provide when connecting to the mail transport host. Like the userid, this setting is rarely used by most mail servers. Leave this field blank unless you use a mail server that requires a user ID and password.

i. **Mail From**. Specifies the mail originator. This value represents the Internet e-mail address that, by default, displays in the received message, as either the “From” or the “Reply-To” address. The recipient's reply comes to this address

j. **Mail Store Host**. Specifies the server to which to connect when receiving mail. This setting 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, then the mail store host is itso.ibm.com.

k. **Mail Store Protocol**. Specifies the protocol to be used when receiving mail. It could be IMAP, POP3 or any store protocol for which the user has installed a provider.

l. **Mail Store userid**. Specifies the user ID to use when connecting to the mail store. This setting combines with the mail store host and password to represent a valid mail account. For example, if the mail account is itso@itso.ibm.com then the userid is *itso*.

m. **Mail Store password**. Specifies the password to use when connecting to the mail store host. This property combines with the mail store userid and host to represent a valid mail account . For example, if the mail account is itso@itso.ibm.com then enter the password for userid *itso*.

n. **Debug**. Toggles debug mode on and off for this mail session. When true, JavaMail’s interaction with mail servers, along with this mail session’s properties will be printed to <stdout>.

7. Click **OK**.
8. Click **Save**, to save the configuration.

18.15.3 Example code

The code segment shown in Example 18-8, illustrates how an application component sends a message and saves it to the **Sent** folder.

Example 18-8 JavaMail application code

```
//get JavaMail session

javax.naming.InitialContext ctx = new javax.naming.InitialContext();
javax.mail.Session mail_session = (javax.mail.Session)
ctx.lookup("java:comp/env/mail/MailSession");

      //prepare message

       MimeMessage msg = new MimeMessage(mail_session);
       msg.setRecipients(Message.RecipientType.TO,
InternetAddress.parse("bob@coldmail.net"));
msg.setFrom(new InternetAddress("alice@mail.eedge.com"));
       msg.setSubject("Important message from eEdge.com");
       msg.setText(msg_text);

      //send message

      Transport.send(msg);

      //save message in “Sent” folder

      Store store = mail_session.getStore();
      store.connect();
     Folder f = store.getFolder("Sent");
     if (!f.exists()) f.create(Folder.HOLDS_MESSAGES);
     f.appendMessages(new Message[] {msg});
```

Refer to the JavaMail API Design Specification, Version 1.2 document for other examples:

`http://java.sun.com/products/javamail/JavaMail-1.2.pdf`

18.15.4 More information

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

- WebSphere InfoCenter. Search on “JavaMail”.

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

- JavaMail API design specification

  `http://java.sun.com/products/javamail/JavaMail-1.2.pdf`

18.16 URLs -an overview

A **Uniform Resource Locator** (URL) is an identifier that points to a *resource* that is accessible electronically, such as a directory on a network machine, a file in a directory or a document stored in a database.

URLs are in the format *scheme:scheme_information*:

- **Scheme**. A scheme might be `http`, `ftp`, `file`, or another term that identifies the mechanism or protocol by which the resource can be accessed.

  In a World Wide Web browser location or address box, a URL for a file available using HyperText Transfer Protocol (HTTP) starts with “http:”. An example is:

  `http:``//www-3.ibm.com/software/webservers/appserv/infocenter.html`

  Files available using File Transfer Protocol (FTP) start with “ftp:”. Files available locally start with “file:”.

- **Scheme_information**. The **scheme_information** commonly identifies the Internet machine making a resource available, the path to that resource, and the resource name.

  The scheme_information for HTTP, FTP and File generally starts with two slashes (//), then provides the Internet address separated from the resource path name with one slash (/). For example:

  `http://www-4.ibm.com/software/webservers/appserv/library.html`

  Note: For HTTP and FTP, the path name ends in a slash when the URL points to a directory. In such cases, the server generally returns the default index for the directory.

For example, the URL:
`http://www.ibm.com/software/webservers/appserv/library.html`

has the scheme `http`, and the scheme_information `//www.ibm.com/software/webservers/appserv/library.html`. From the scheme information we can identify the machine where the information resides (`www.ibm.com`), the path (`/software/webservers/appserv/`), and the resource name (`library.html`).

Note: A URL can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for http is 80:

`http://w3.ibm.com:80/` is equivalent to `http://w3.ibm.com/`

More information on the types of URLs and their formats can be found at:

```
http://java.sun.com/products/jdk/1.2/docs/api/java/net/URL.html
```

18.17 URL providers

A URL provider implements the functionality for a particular URL protocol, such as HTTP, by extending the *java.net.URLStreamHandler* and *java.net.URLConnection* classes. It enables communication between the application and a URL resource that is served by that particular protocol.

A URL provider named *Default URL Provider* is included in the initial WebSphere configuration. This provider utilizes the URL support provided by the IBM JDK. Any URL resource with protocols based on the Java 2 Standard Edition 1.3.1, such as HTTP , FTP or File, can use the default URL provider.

Author Comment: This doesn't appear to be the case at the moment, or at least I cannot see the Default URL Provider from the console. Checked entire configuration of cell for XML files containing its definitions. No definition of a default URL provider!!!

Customers can also "plug-in" their own URL providers that implement other protocols not supported by the JDK.

Author Comment: Could we provide with the list of url protocols provided by the ibm jdk? Cannot find a list.

18.18 Configuring URL providers and URLs

URL resource objects are administrative objects used by an application to communicate with an URL. These resource objects are used to read from an URL or to write to an URL.

URL resource objects use URL providers for class implementation.

In this section we describe how to configure an URL provider and how to configure URL resource objects associated with that provider.

18.18.1 Configuring an URL Provider

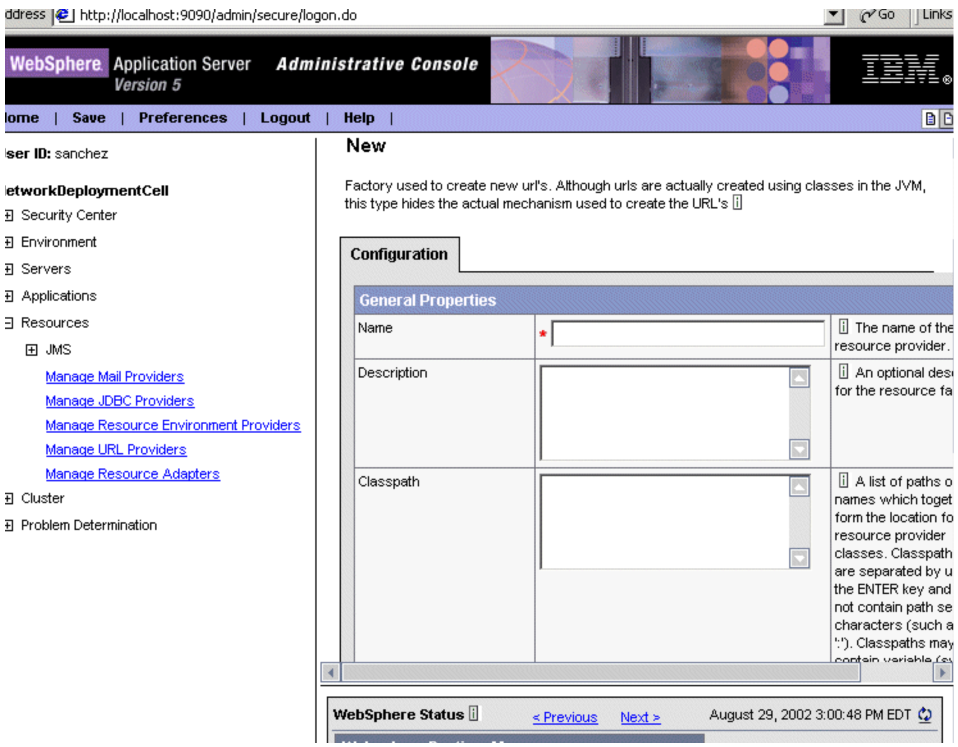
To configure or create an URL provider from the administrative console, complete the following tasks:

1. Expand **`Resources`** from the navigation tree.
2. Click on **`Manage URL Providers`**.
3. Select the **`Scope`** and click **`Apply`**.

   The scope determines whether URL resources configured to use this provider will be available at the cell, node or the application server level.

4. Click **`New`** to configure a new URL provider, or select an existing one to edit it.

   The **Configuration** page for a new URL provider is shown in Figure 18-57.



Factory used to create new url's. Although urls are actually created using classes in the JVM, this type hides the actual mechanism used to create the URL's

Name	*	The name of the resource provider.
Description		An optional des for the resource fa
Classpath		A list of paths o names which toget form the location fo resource provider classes. Classpath are separated by u the ENTER key and not contain path se characters (such a ':'). Classpaths may

Figure 18-62 URL provider configuration page

The properties to configure are:

a. **Name**. An administrative name for the URL provider.

b. **Description**. An optional description of the URL provider, for your administrative records.

c. **Classpath**. A list of paths or JAR file names which together form the location for the URL provider classes.

d. **Stream Handler Class Name**. Specifies the fully qualified name of the Java class that implements the stream handler for the protocol specified by the **Protocol** property. A stream protocol handler knows how to make a connection for a particular protocol type, such as HTTP or FTP. It extends the *java.net.URLStreamHandler* class for that particular protocol.

e. **Protocol**. Specifies the protocol supported by this stream handler. For example, "http" or "`ftp`".

5. Click **`OK`**.
6. Click **`Save`**, to save the configuration.

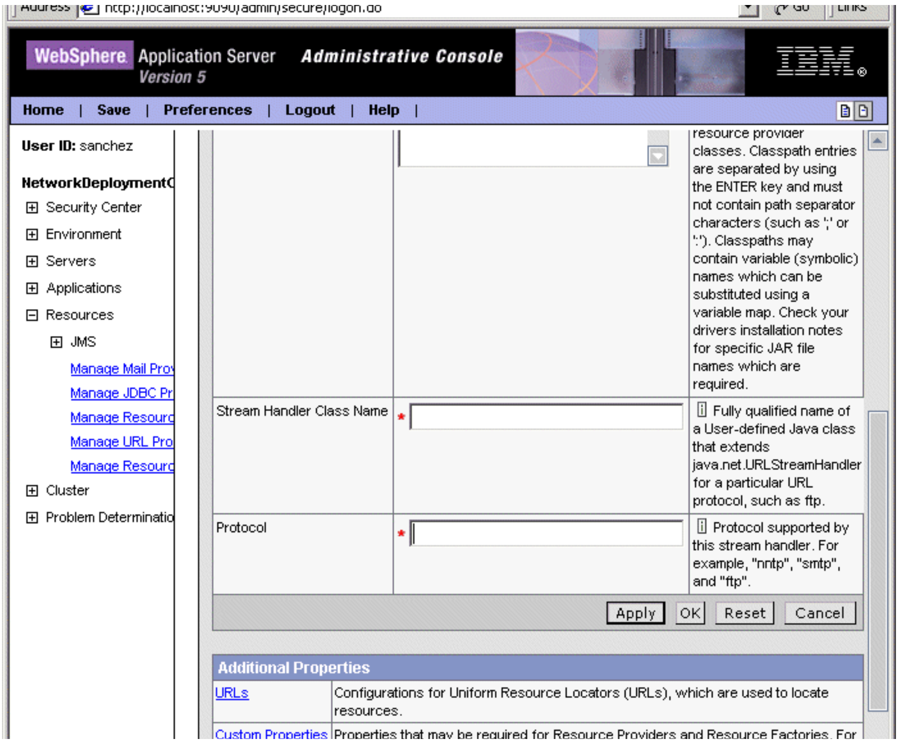
Important: You need to manually install the URL provider (a set of JARs) on each node where the URL provider is going to be used and ensure that it is included in the classpath above.

18.18.2 Configuring URLs

To configure an URL administrative object that points to an electronically accessible resource, such as a directory or a file on a network machine, complete the following steps from the administrative console:

1. Expand **`Resources`** from the navigation tree.
2. Click on **`Manage URL Providers`**.
3. Select the **`Scope`** and click **`Apply`**.

   This will show all the URL providers created at that particular scope.
4. Select the URL provider that implements the protocol required to access the URL resource.
5. On the URL provider **Configuration page**, under **`Additional Properties`**, select **`URLs`**, see Figure 18-63:


Address http://localhost:9090/admin/secure/logon.do

WebSphere Application Server Version 5 Administrative Console

Home | Save | Preferences | Logout | Help |

User ID: sanchez

NetworkDeployment

Security Center
Environment
Servers
Applications
Resources
JMS
Manage Mail Prov
Manage JDBC Pr
Manage Resourc
Manage URL Pro
Manage Resourc
Cluster
Problem Determinatio

		resource provider classes. Classpath entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Classpaths may contain variable (symbolic) names which can be substituted using a variable map. Check your drivers installation notes for specific JAR file names which are required.
Stream Handler Class Name	*	Fully qualified name of a User-defined Java class that extends java.net.URLStreamHandler for a particular URL protocol, such as ftp.
Protocol	*	Protocol supported by this stream handler. For example, "nntp", "smtp", and "ftp".

Apply OK Reset Cancel

Additional Properties

URLs	Configurations for Uniform Resource Locators (URLs), which are used to locate resources.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For

Figure 18-63 URL provider configurable resources

The properties to configure are:

a. **Name**. Specifies the administrative name for the URL resource object.

b. **JNDI Name**. The URL session object name as registered in the application servers name space, including any naming subcontext.

When installing an application that contains modules with URL resource references, the resources defined by the deployment descriptor of the module need to be bound to the actual JNDI name of the resources.

Tip: As a convention, use the value of the **Name** property prefixed with “url/” (such as “url/UrlName”).

c. **Description**. An optional description of the URL object, for your administrative records.

d. **Category**. Specifies a category that you can use to classify or group the URL object.

   e. **Spec**. Specifies the URL resource to which this URL object is bound to, for example “`file:///d:/url/motd.txt`”.

6. Click **OK**.
7. Click **Save**, to save the configuration.

18.18.3 URL provider sample

Example 18-9 provides a code sample making use of the URL provider and URL resources.

Note that the Web module resource reference, `myHttpUrl`, is bound to the URL resource JNDI name, `url/MotdUrl`, during application assembly or deployment.

Example 18-9 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 18-9 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 18-10. 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 18-10 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 a URL string.

18.18.4 More information

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

- WebSphere InfoCenter. Search on URL.

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

- URL class documentation.

  `http://java.sun.com/j2se/1.3/docs/api/`

18.19 Resource Environment Providers

Author Comment: This section needs to be written properly.

The *java:comp/env* environment provides a single mechanism by which both JNDI namespace objects and local application environment objects can be looked up. The WebSphere 5.0 runtime provides a number of local environment entries by default.

The Resource Environment Provider is the WebSphere 5.0 mechanism for defining custom (non-default) environment entries at the cell, node or application server scope.

Each Resource Environment Provider resource encapsulates the settings necessary to resolve a JNDI name, within the local application (java:comp/env) environment, to the type of the returned object instance.

The required settings are:

1) **JNDI name.** The JNDI name of the entry within the local application (java:comp/env) environment.

2) **Environment entry factory class.** The classname of the factory class accessed via the *java:comp/env* entry. The factory returns an instance of a class that implements the interface defined as the return type.

3) **Environment entry returned Java type.** The Java type (usually an interface) of the instance returned by the factory.

For example, if

```
JNDI name = myapp/MyLogWriter

Factory class = com.ibm.itso.test.LogWriterFactory

Returned Java type = com.ibm.itso.test.LogWriter
```

the application can then use the `myapp/MyLogWriter` JNDI name from within the code to instantiate an object that implements the `LogWriter` interface, instead of hardcoding the factory class name within the application. In this example, the object is instantiated by the factory class `LogWriterFactory`.

By using this approach you hide specifics of class implementation from the application. You could for example change the factory implementation class without having to change the application code. You would only need to change the resource environment entry from the administrative console.

Example 18-11 and Example 18-12 shows the difference in using one approach over another.

Example 18-11 Code sample using a resource enviroment entry

```
import com.ibm.itso.test.*;
....
InitialContext ctx = new InitialContext();
LogWriter mylog = (LogWriter) ctx.lookup(“java:comp/env/myapp/MyLogWriter”);
mylog.write(msg);
....
```

Example 18-12 Code sample using factory class name specifics

```
import com.ibm.itso.test.*;
....
LogWriter mylog = (LogWriter) LogWriterFactory.createInstance();
mylog.write(msg);
....
```

18.20 Configuring the Resource environment provider

Use this page to create settings for a resource environment provider.

To view this administrative console page, click **`Resources`** >**`Manage Resource Environment Provider`** in the console navigation tree.

Select one of the resource environment providers in your collection.

Configuration tab.

Name

The name of the resource provider.

Description

A text description for the resource factory.

18.21 Configuring Referenceables

Use this page to set the factoryClassname of the factory that will convert information in the name space into a class instance for the type of resource desired.

To view this administrative console page, click **`Resources`** > **`Manage Resource Environment Provider`** in the console navigation tree.

Select one of the providers listed.

On the panel that appears, click **`Referenceables`** in the **`Additional Properties`** area.

Configuration tab

Factory Classname

Contains a javax.naming.ObjectFactory implementation class name

Classname

The Java type that a Referenceable provides access to, for binding validation and to create the reference Data type String.

18.22 Configuring Resource Environment Entries

Use this page to set resource environment entries.

To view this administrative console page, click **`Resources`** > **`Manage Resource Environment Provider`** in the console navigation tree.

On the Resource Environment Provider page, click on one of the resource providers in your collection, or click on the **`New`** button.

On the Resource Environment Provider you selected, click **`Resource Env Entrys`** in the **`Additional Properties`** area.

On the **Resource Env Entry** page, select one of the resources listed in your collection.

Configuration tab

Name

(Required) A display name for the resource.

JNDI name

(Required) The JNDI name for the resource, including any naming subcontexts.

This name is used as the link between the platform's binding information for resources defined by a module's deployment descriptor and actual resources bound into JNDI by the platform.

Description

A text description for the resource.

Category

A category string that can be used to classify or group the resource.

Referenceable

The referenceable holds the factoryClassname of the factory that will convert information in the name space into a class instance for the type of resource desired, and for the classname of the type to be returned. Data type: Dropdown menu.

19

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 WebSphere Studio Application Developer are also capable of packaging J2EE applications. Please refer to redbook <Reference to V5 AD Handbook goes here> for more details.

19.1 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 19-1 below, it lets you create EJB modules, Web modules, RAR modules, as well as application client modules. You can start the AAT from the command line using the **`assembly.bat/assembly.sh`** commands. You can also assemble existing modules inside an enterprise application. You can find more information on the AAT in the InfoCenter.

AAT Version 5.0 supports by default the J2EE 1.3 specification. Anything you *create* with it follows that specification level (Servlet 2.3, EJB 2.0, etc.). However, you can edit J2EE 1.2 applications with the AAT. You can also import existing J2EE 1.2 modules (such as EJB 1.1 EJB modules) in a J2EE 1.3/1.2 application.

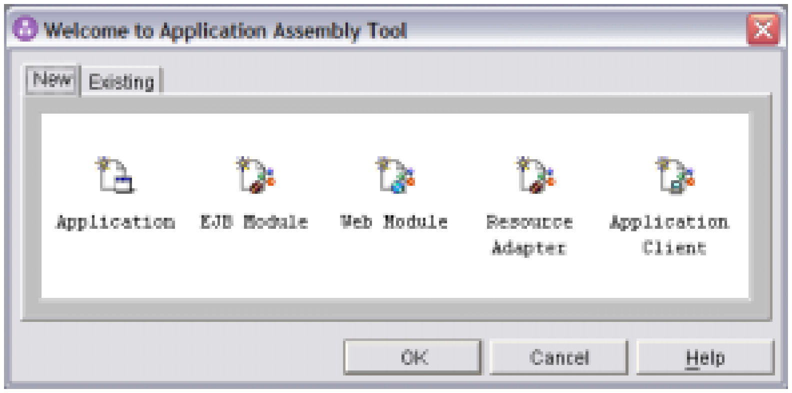


Figure 19-1 Application Assembly Tool welcome window

In this chapter, we explain how to create an entreprise application and add various modules to it. First, you will create an EJB module, assuming we only have the necessary implementation classes. Then, we create a Web module, then the client application modules.

Tip: If you need to edit a module which contains lots (hundreds) of files, you may experience some opening delays or even crashes if you do not increase the AAT maximum heap size (as with any Java application). This can be done by adding the **`-Xmx<size>m`** parameter to the assembly.bat/sh file.

19.2 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. The application is described in details in the Appendix A, “Webbank Application Overview” on page 17. We highly recommend that you read this appendix to understand the application design and implementation.

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 webbankSample/src/ejbModule folder. This folder also contains the session EJBs helper classes (used by servlets and clients).
2. The web application is stored under the webbankSample/src/Web Content folder. This includes the servlets classes, along with the JSP files, libraries used by servlets only (such as the Struts framework), and the static files of the application (HTML and images).
3. Common classes, which are used by both enterprise beans and servlets, are packaged in a webbankCommon.jar file, stored in the webbankSample\jars folder.
4. Finally, the stand-alone clients class files are stored under the webbankSample/src/appClientModule folder.

Figure 19-2 on page 776 depicts the code organization.

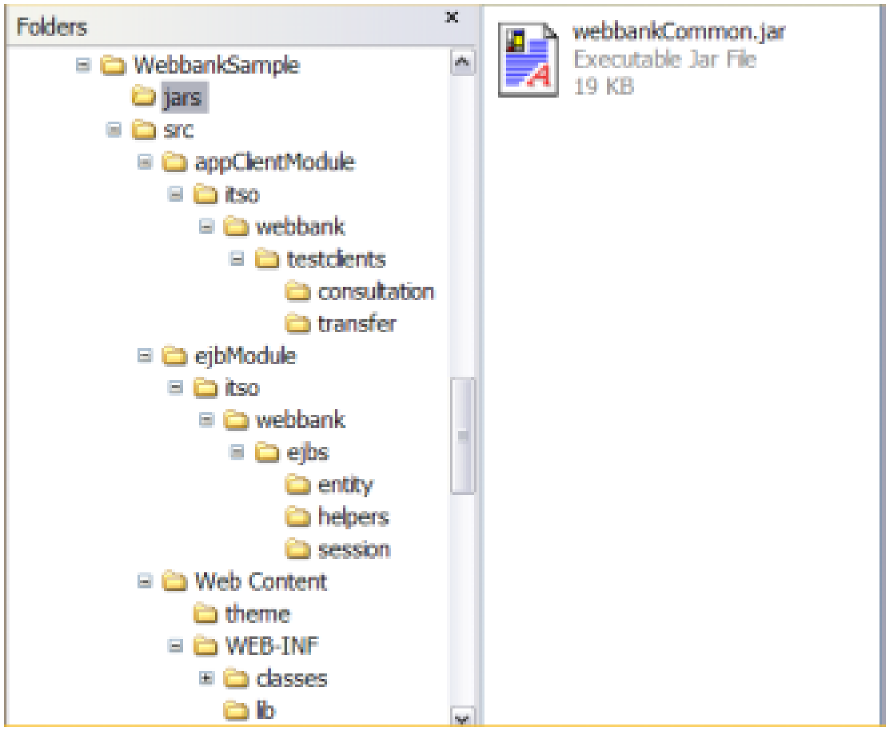


Figure 19-2 Source Code Organization before packaging

Instructions below assume that you have unpacked the webbank.zip file under E:. 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

19.3 Creating the Webbank enterprise application

You will now use the Enterprise Application wizard to create the archive for the Webbank enterprise application.

1. Start the AAT, and choose to create a new application. You can specify the following properties for the application:
   - Display name

     The name used to identify this application, such as webbankApplicationV5.
   - Description

An optional description of what this application does. We recommend that you put a meaningful description of the application for the application deployer.

The next step is to add the WebbankCommon.jar file at the root of the EAR file, tomake it available to modules which depend on it.

2. Select the **Files** entry, and invoke **Add Files...** from its contextual menu.
3. Click on the **Browse** button, and select the E:\WebbankSample\jars directory.
4. Select the webbankCmmon.jar file, and click on **Add**.
5. Click on **OK**.
6. Save the enterprise application as **E:\WebbankSample\jars\webbank.ear**.

You are now ready to add modules to the enterprise application. In the following steps, you should regularly save the EAR file to commit changes, even if this is not specifically mentioned in the packaging instructions!

19.4 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 (or a tool such as WebSphere Studio Application Developer) to update these files. However, we have documented the contents of the generated files to help you maintain these files using a text editor.

19.4.1 Creating the EJB module

To create an EJB module in an enterprise application, you must select the EJB Modules entry, and invoke **New**. As shown in Figure 19-3, the following properties can be set for an EJB module:

- File name

The name of the JAR file the EJB module will be saved in, for example webbankEJBs.jar

- 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

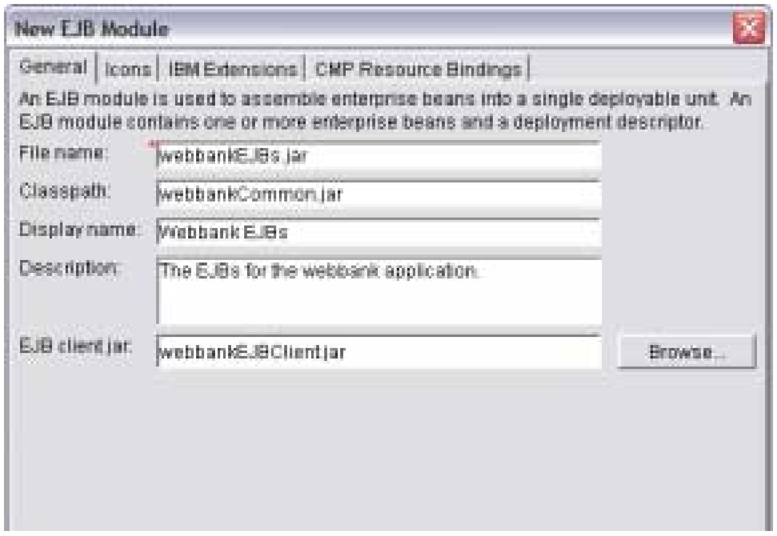
  The JAR file containing the classes that a client application would need.

Author Comment: Need to be more precise here..This can be used in place of the Manifest classpath approach mentioned below. But we do not have a tool to generate this file out of the standard EJB JAR file....

- 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
```



An EJB module is used to assemble enterprise beans into a single deployable unit. An EJB module contains one or more enterprise beans and a deployment descriptor.

File name: webbankEJBs.jar
Classpath: webbankCommon.jar
Display name: Webbank EJBs
Description: The EJBs for the webbank application.
EJB client jar: webbankEJBClient.jar

Figure 19-3 Creating an EJB Module

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. Any type used by an EJB method must be available on the classpath. This includes exceptions thrown by the method, method return types, or types used as method paramaters. As an example, 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.

You must reference the webbankCommon.jar file using a relative path (absolute paths, such as E:\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 (which we did previously).

19.4.2 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 at the EJB module level (not at the EAR level), and select **Add Files** from the pop-up menu.
2. Click **Browse...** and select the E:/WebbankSample/src/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 E:\WebbankSample\src\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**.

19.4.3 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 19-4 on page 781 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.

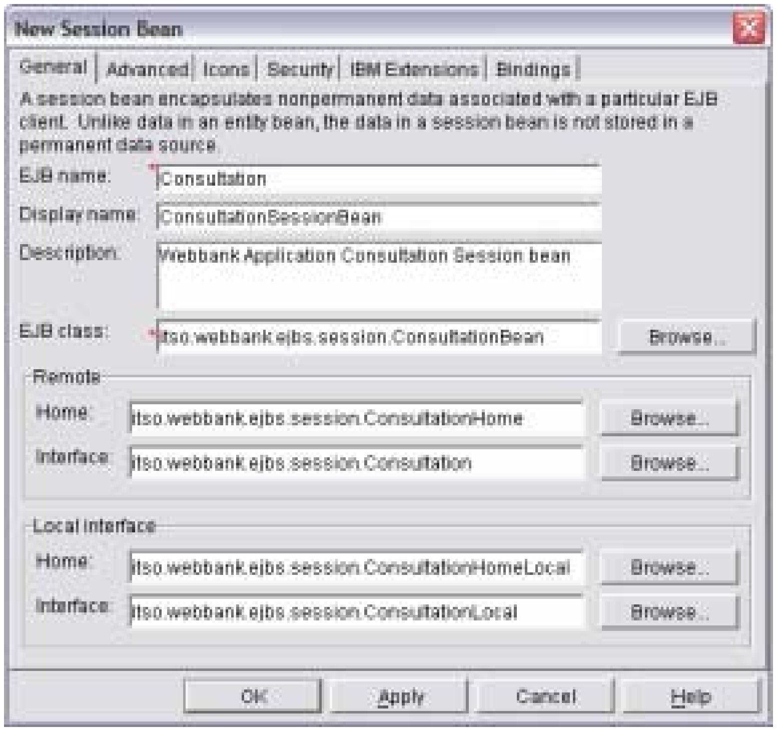
   J2EE 1.3 introduces the concept of remote and local interfaces/homes (you can refer to Chapter 3, “The Java 2 platform” on page 33 for more details). In this sample, we have chosen to create both remote and local interfaces/homes for session beans and only local interfaces to entity beans. Therefore, for session beans you must provide information for both the local and remote interfaces fields.
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.

Author Comment: THIS DOES NOT WORK FOR NOW....Open PMR!

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. Switch to the Advanced page.
7. You must now select the session EJB type, in our case **Stateless**.

8. 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).
9. Click **OK**.
10. Repeat steps 1 to 8 for the Transfer and the TransferStateless session EJBs. However, for the Transfer session EJB, you must set the Session type to **Stateful** and the Transaction type to **Bean** (the Transfer EJB manages transaction demarcations).



A session bean encapsulates nonpermanent data associated with a particular EJB client. Unlike data in an entity bean, the data in a session bean is not stored in a permanent data source.

EJB name: Consultation
Display name: ConsultationSessionBean
Description: Webbank Application Consultation Session bean
EJB class: itso.webbank.ejbs.session.ConsultationBean

Remote
Home: itso.webbank.ejbs.session.ConsultationHome
Interface: itso.webbank.ejbs.session.Consultation

Local interface
Home: itso.webbank.ejbs.session.ConsultationHomeLocal
Interface: itso.webbank.ejbs.session.ConsultationLocal

Figure 19-4 New Session Bean window

11. Save the EAR file!

Table 19-1 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 19-1 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.
Local Home Interface	The class file that contains the EJB local home interface, such as itso.webbank.ejbs.session.ConsultationHomeLocal (EJB 2.0 only) This corresponds to the `<local-home>` entry.
Local Interface	The class file that contains the enterprise bean local interface, such as itso.webbank.ejbs.session.ConsultationLocal (EJB 2.0 only). This corresponds to the `<local>` 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.

19.4.4 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/local home interface, remote/local interface, as well as the enterprise bean class for the BranchAccount enterprise bean (located under itso/webbank/ejbs).
3. Switch to the Advanced page
4. 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.
5. 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.
6. Set the version to **2.x**. This field is used to check which EJB specification this EJB complies to.
7. Set the abstract schema name to BranchAccount.
8. 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.

9. Right-click the **CMP Fields** entry under the BranchAccount enterprise bean, and select **New.**
10. 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).
11. Repeat step 6 and 7 for all CMP fields.
12. Repeat steps 1 to 8 for the CustomerAccount and Customer enterprise beans.

13. Save the EAR file!

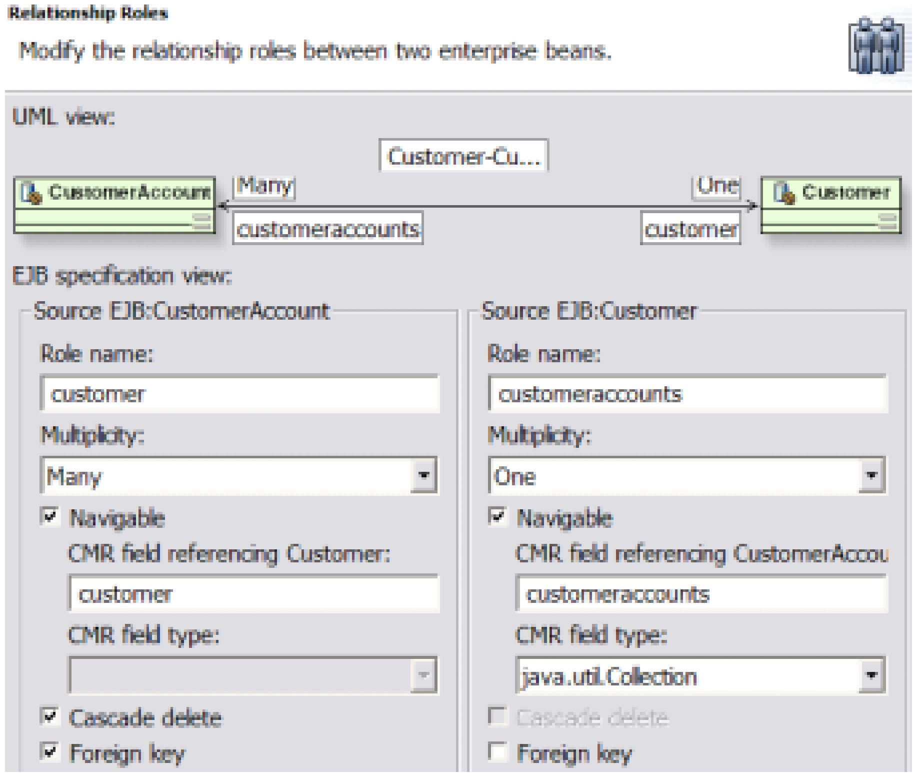
Table 19-2 Entity beans specific properties

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.
Abstract Schema Name	This can be viewed as the name of the table corresponding to the entity bean. This field is especially used when you define EJB QL queries. This corresponds to the `<abstract-schema-name>` entry.
Version	The EJB version, either 2.x or 1.x. This corresponds to the `<cmp-version>` 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>`

19.4.5 Declaring Container Managed Relationships

Container-managed relationships (CMR) are new in the EJB 2.0 specification. In the webbank sample, we have defined a one-to-many bidirectional relationship between the Customer and Customer Account enterprise beans. The UML view of this relationship is depicted in Figure 19-5. This figure shows a view of the relationship editor from WebSphere Studio Application Developer V5, which was used to develop this sample.


Relationship Roles

Modify the relationship roles between two enterprise beans.

UML view:

EJB specification view:

Source EJB:CustomerAccount

Role name: customer

Multiplicity: Many

☑ Navigable

CMR field referencing Customer: customer

CMR field type:

☑ Cascade delete

☑ Foreign key

Source EJB:Customer

Role name: customeraccounts

Multiplicity: One

☑ Navigable

CMR field referencing CustomerAccou

customeraccounts

CMR field type: java.util.Collection

☐ Cascade delete

☐ Foreign key

Figure 19-5 Customer-CustomerAccount CMR as viewed in WebSphere Studio.

A CMR is represented by a solid line between two enterprise beans. This solid line simply means some kind of "link" exists between them. Each association is named (Customer-CustomerAccount) and must be customized to provide the following information:

- **Navigability**. Navigability indicates whether you can traverse the link. In other words, it indicates whether you can retrieve a customer from a customer account, or customer accounts from a customer. An arrow on the relationship link shows the direction in which the link can be traversed. A plain link indicates an association that can be traversed both ways.
- **Role names**. A role name is the name by which a class is known to the other object in the association. A customer sees **customeraccounts**, and a customer account sees a **customer**.
- **Multiplicity**. For each role, multiplicity specifies how many values of an association an object may have. In this model, we have defined a **Many** multiplicity for the CustomerAccount EJB, specifying it can have multiple links

to a Customer. On the other hand, we have specified a **One** multiplicity for the Customer EJB, specifying there is one link to a CustomerAccount. Actually, this also means a CustomerAccount can't exist without a corresponding customer (this link must exist).

- The **Cascade delete** option indicates that all customer accounts should be deleted when the corresponding customer is deleted.
- From a database implementation point of view, the relationship will be expressed via a **foreign key** relationship. A CUSTOMER_ID foreign key has been created for the customer account table, as shown in the SQL definition below:

```
ALTER TABLE ISABELLE.CUSTOMERACCOUNT
ADD CONSTRAINT C2704651 FOREIGN KEY (CUSTOMERID)
REFERENCES ISABELLE.CUSTOMER(ID);
```

The corresponding definition in the ejb-jar.xml can be seen in Example 19-1 below.

Example 19-1 EJB Relationship definition in ejb-jar.xml file.

```
<ejb-relation>
         <description>Customer has many accounts</description>
         <ejb-relation-name>Customer-CustomerAccount</ejb-relation-name>
         <ejb-relationship-role>
            <ejb-relationship-role-name>customer</ejb-relationship-role-name>
            <multiplicity>Many</multiplicity>
            <cascade-delete />
            <relationship-role-source>
               <ejb-name>CustomerAccount</ejb-name>
            </relationship-role-source>
            <cmr-field>
               <cmr-field-name>customer</cmr-field-name>
            </cmr-field>
         </ejb-relationship-role>
         <ejb-relationship-role>

<ejb-relationship-role-name>customeraccounts</ejb-relationship-role-name>
            <multiplicity>One</multiplicity>
            <relationship-role-source>
               <ejb-name>Customer</ejb-name>
            </relationship-role-source>
            <cmr-field>
               <cmr-field-name>customeraccounts</cmr-field-name>
               <cmr-field-type>java.util.Collection</cmr-field-type>
            </cmr-field>
         </ejb-relationship-role>
      </ejb-relation>
```

You can declare a relationship in the AAT by selecting the **EJB Relations** entry for an EJB module, and map the definitions to the ones described above.

Note: CMR require quite some changes to the implementation of your EJBs. For example, the customer enterprise bean should have a `public abstract java.util.Collection getCustomeraccounts()` method defined. Therefore, you are very likely to create CMRs with appropriate tools, as opposed to manually create them using the AAT.

WebSphere Application Server provides an EJB extension called read-ahead that CMR implementations can take advantage of. Please refer to “Using read-ahead hints” on page 829 for details.

19.4.6 Declaring Message Driven beans

The WebbankClerk application is a separate application which can be used by bank managers to monitor transfers over $5000. A message is posted by the Webbank application onto a WEBBANK Queue (via the internal MQSeries server provided in WebSphere V5). A call to a message driven bean (MDB) will be driven each time a new message arrives on the queue. The MDB will analyze the message and log it in a table. The Webbank clerk application displays the contents of the log table for a given day.

In the following steps, we explain how to package message driven beans using the AAT.

Author Comment: Stilll needs to be completed.

The next section describes how to declare find methods for entity beans.

19.4.7 Declaring find methods for entity beans

Find methods are declared on the entity bean home interface. The BranchAccount entity bean has a find method called findByNameLike, which returns a list of all branch accounts based on the branch name.

The way you declare a find method is different depending if you are using EJB 1.1 or EJB 2.0. In EJB 1.1, there is no standard way (in the Enterprise JavaBeans 1.1 specification) to tell an EJB container which SQL query should be executed when the find method is invoked. The EJB 2.0 specification introduces a new language, EJBQL, the EJB query language, to declare find methods in a standard way.

Declaring find methods for EJB 1.1

Two types of finders are supported in WebSphere V5.0.

- User

  Normally, the EJB container is responsible for generating the code for a find Method based on information you provide, such as the FullQueryString or the WhereClauseString (see the example 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. Although this is still supported, we highly recommend to use the “Where Clause” option instead. Please refer to the article *Implementing custom finder helpers for CMP entity beans* in the InfoCenter for more details.

- Where Clause

  This option lets you specify the WHERE part of an SQL query, such as:

  ```
  T1.BRANCHNAME LIKE CONCAT(CONCAT('%',?),'%')
  ```

Example 19-2 Specify a find method SQL query string

```
public interface BranchAccountBeanFinderHelper {
   public static final String findByNameLikeWhereClause =
      "T1.BRANCHNAME LIKE CONCAT(CONCAT('%',?),'%')";
}
```

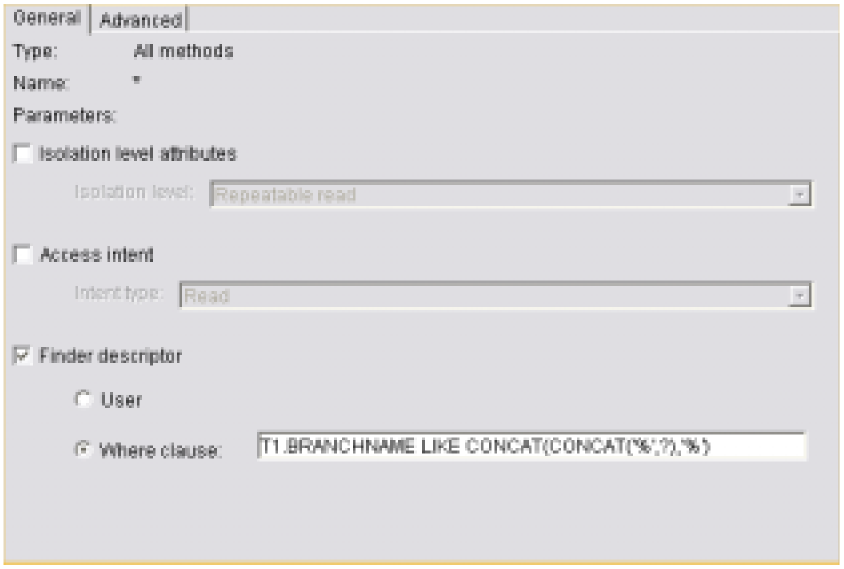
Note: The “Full Select” option which was available in Version 3.5 and deprecated in Version 4.0, is no longer supported. Also, EJB QL is only supported by the EJB 2.0 container.

Specify the find 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 19-6, select the **Where clause** option and enter the following where clause:

   ```
   T1.BRANCHNAME LIKE CONCAT(CONCAT('%',?),'%')
   ```

5. Click **OK**.
6. Save the EJB module!



Type: All methods
Name: *
Parameters:
☐ Isolation level attributes
Isolation level: Repeatable read
☐ Access intent
Intent type: Read
☑ Finder descriptor
○ User
● Where clause: T1.BRANCHNAME LIKE CONCAT(CONCAT('%',?),'%')

Figure 19-6 Adding a find method to an EJB 1.1

When you create a finder, it inserts a definition in the ibm-ejb-jar-ext.xmi file, as shown in Example 19-3.

Example 19-3 Find method 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>
```

Attention: The finder in Example 19-3 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.

Declaring EJB 2.0 SQL Queries

With EJB 2.0, you must use the new EJB Query Language (EJB QL) to define queries. EJB QL lets you create SQL queries manipulating objects, as opposed to providing directly the SQL code (as you would do in Version 4.0). The persistence manager will then translate EJB QL into the actual SQL code that will be executed against the database. Queries can be associated to find methods, which can be invoked remotely or ejbSelect methods which can only be used by the bean itself (they are not exposed on the bean home interface). It is beyond the scope of this book to cover in details find or ejbSelect methods features or teach you EJB QL. We only cover here how to declare queries using the AAT. We assume that we have defined the same findByNameLike find method on the branch account EJB home interface, and explain how to associate this method to the appropriate EJB QL query. To associate a query to the findByNameLike method, you must do the following:

1. Select the BranchAccount bean, select the **Queries** entry, and invoke **`New`** from its contextual menu.
2. Provide the following information, as shown in Figure 19-7.

- ► Name

  The name of the method associated to this query.

- ► Parameters

  The list of parameters (identified by their type) to be passed to the find method.

- ► Result Type

  Applies to EJB Select methods only (to indicate to the container whether returned objects should Local or Remote objects). Leave this field blank for find methods.

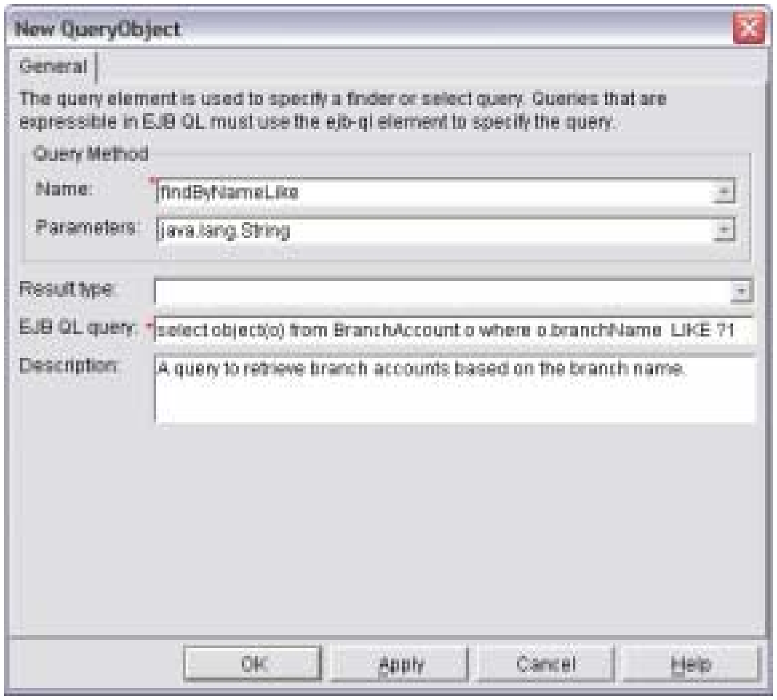
- ► EJB QL Query

  The query to be executed when the find method is invoked, here: `select object(o) from BranchAccount o where o.branchID LIKE ?1`

- ► Description

  An optional description of the query.

3. Click on OK.



The query element is used to specify a finder or select query. Queries that are expressible in EJB QL must use the ejb-ql element to specify the query.

Query Method

Name: findByNameLike

Parameters: java.lang.String

Result type:

EJB QL query: select object(o) from BranchAccount o where o.branchName LIKE ?1

Description: A query to retrieve branch accounts based on the branch name.

Figure 19-7 Creating a EJB QL query

A query is described in the ejb-jar.xml file as shown in Example 19-4.

Example 19-4 Declaration of the findByNameLike query in the ejb-jar.xml file

```
<query>
   <description>A finder to retrieve specific branch accounts</description>
   <query-method>
      <method-name>findByNameLike</method-name>
      <method-params>
         <method-param>java.lang.String</method-param>
      </method-params>
   </query-method>
   <ejb-ql>select object(o) from BranchAccount o where o.branchID LIKE ?1
   </ejb-ql>
</query>
```

Tip: WebSphere Application Server Enterprise Edition provides extensions to the standard EJB QL. It allows you, for example, to invoke EJBs methods in a query (see fs.price() in the example below) , or specify which columns should be returned (as opposed to a record/set of records). For example, you could write something like:

```
select ff.flightNumber, ff.fromAirport.name, ff.duration,
ff.provider.name, ff.toAirport.name, fs.flightNumber,
fs.toAirport.name, fs.provider.name, ff.duration+fs.duration,
fs.price()+fs.price()
from Flight ff, Flight fs
where ff.fromAirport.code='DFW' and fs.toAirport.code='SFO' and
ff.toAirport=fs.fromAirport
```

Please refer to <Reference to V5 enterprise handbook> for more details.

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.

19.4.8 Importing enterprise beans 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/EAR file that contains the enterprise beans you want to import and click **Open**. Two combo boxes are available as the top of the dialog (as shown Figure 19-8). The left one lets you select the EJB module from which you want to import the EJB. The right one lets you choose the EJB (session *or* entity) found by AAT in the JAR file.

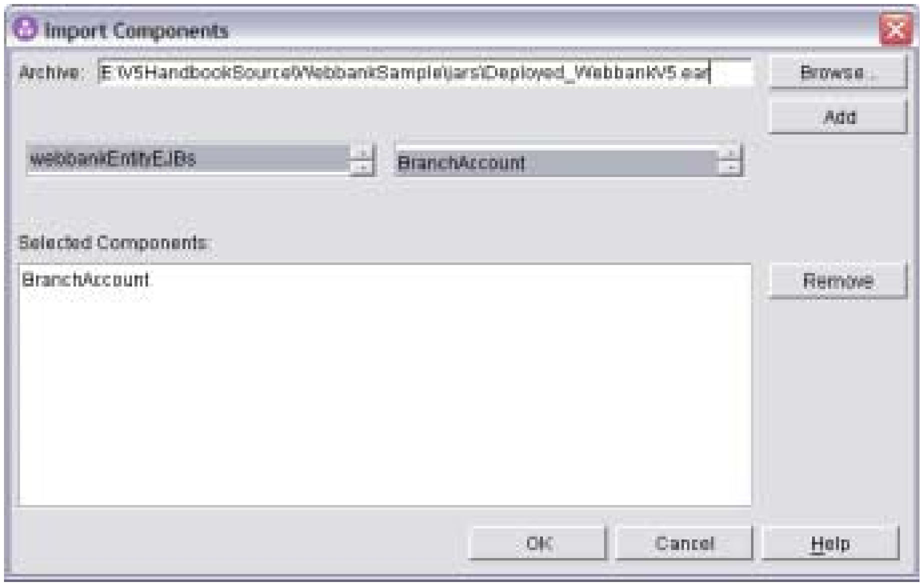


Figure 19-8 Importing the BranchAccount Entity EJB

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.
5. Save the EAR file!

Our EJB module is now complete. Next, we package the Webbank Web module.

19.5 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: three servlets (TransferServlet/Consultation Servlet/Struts action servlet), two JSP pages (webbank.jsp, message.jsp), and a welcome page (webbank.html). This is a very simple example, but enough to demonstrate the principles of Web module packaging.

Note: Note that the Struts action servlet and the webbank.jsp pages are the Struts-based equivalent of the TransferServlet and webbank.html pages. You do not really need to install both.

A Web module is represented by a WAR file (Web archive). A WAR deployment descriptor is stored in a web.xml file. Similarly to EJBs, extensions to the standard deployment descriptor are stored in an ibm-web-bnd.xmi file (for bindings) and an ibm-web-ext.xmi file (for extensions).

19.5.1 Creating a Web module

To create a Web module in an existing EAR file, you must select **New** for the **Web Module** entry of the EAR definition. The following properties can be set tfor a Web module, as shown in Figure 19-9 on page 795:

- File Name

  The name of the WAR (Web Archive) file that will be contain all the Web module artefacts, for example `webbankWeb.war`.

- Context root

  The unique URI for this web module, for example `webbank`. This information is stored in the application.xml file (at the EAR level).

- 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.

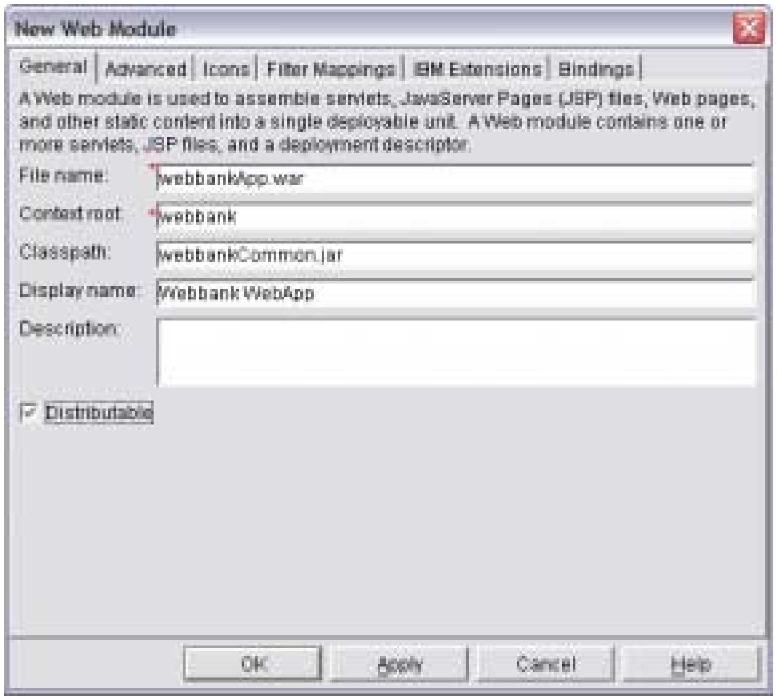
- Distributable

  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
```



A Web module is used to assemble servlets, JavaServer Pages (JSP) files, Web pages, and other static content into a single deployable unit. A Web module contains one or more servlets, JSP files, and a deployment descriptor.

File name: webbankApp.war
Context root: webbank
Classpath: webbankCommon.jar
Display name: Webbank WebApp
Description:
[✓] Distributable

Figure 19-9 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 E:\webbankSample\jars\webbankWeb.war.

You are now ready to add a servlet to your Web module.

19.5.2 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, the ConsultationServlet, and 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 (at the root of 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.

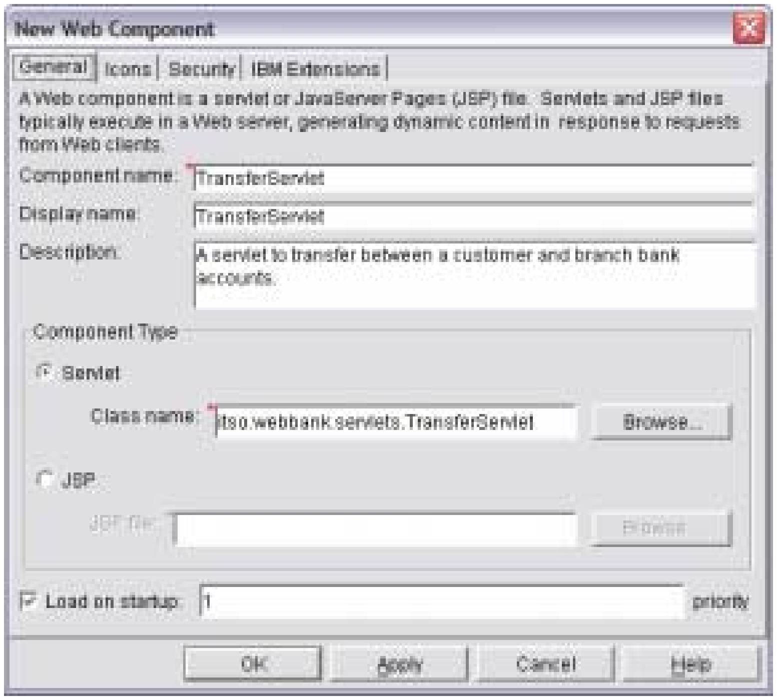
For our sample, we need to add the servlet class files, struts lib files, as well as all the resource files. Proceed as follows to add this different files to the Web module:

1. Select **Files -> Class Files -> Add Files**.
2. Click **Browse...**, and select the E:\webbankSample\src\WebContent directory.
3. Expand the tree under E:\webbankSample\src\WebContent until you reach the servlets folder, and select all servlet classes (including the BasicHTTPServlet class files).
4. Click **Add**.
5. Click **OK**.
6. Select **Files-> Jar Files -> Add Files**.
7. Click **Browse...**, and select the E:\webbankSample\src\WebContent\lib directory.
8. Hold the Ctrl key, select the struts.jar file and click **Add**.
9. Click **OK**.
10. Select **Files-> Resource Files -> Add Files**.
11. Click **Browse...**, and select the E:\webbankSample\src\WebContent directory.
12. Hold the Ctrl key and select all gif, jsp, and html files, and click **Add**.
13. Click **OK**.

19.5.3 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 19-10 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).
8. Click **OK**.



A Web component is a servlet or JavaServer Pages (JSP) file. Servlets and JSP files typically execute in a Web server, generating dynamic content in response to requests from Web clients.

Component name: TransferServlet

Display name: TransferServlet

Description: A servlet to transfer between a customer and branch bank accounts.

(•) Servlet

Class name: itso.webbank.servlets.TransferServlet

( ) JSP

JSP file:

[✓] Load on startup: 1 priority

Figure 19-10 Creating the TransferServlet component

19.5.4 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 19-5 shows how a servlet context parameter is specified in the web.xml file.

Example 19-5 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 19-6 shows the entry added to the web.xml file when defining the tag library.

Example 19-6 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 19.16.4, “Default error page” on page 831) is used.

Adding welcome files

The Servlet API V2.3 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 19.16.1, “File serving servlet” on page 830. 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.

19.5.5 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 19.16.3, “Serve servlets by class name” on page 831.

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.

19.5.6 Declaring Servlet Filters and Events

Author Comment: New in Servlet 2.3. Needs to be documented here.

19.5.7 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 choose to precompile all application JSPs at deployment time.

Example 19-7 shows how a JSP component is declared in the web.xml file.

Example 19-7 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.

19.6 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 (such as security).

In this section, we cover how to package the stand-alone clients for the Webbank application as J2EE clients. Those clients let you to retrieve respectively make transfers and retrieve the branch and customer accounts balance. J2EE clients 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 **E:\webbankSample\src\appClientModule**.
4. Expand the tree view in the left pane until you reach the consultation 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 19-11 on page 803:
   – Display name

     The name used by GUI tools when displaying this client application.

   – Description

     An optional description of the contents of this client application.

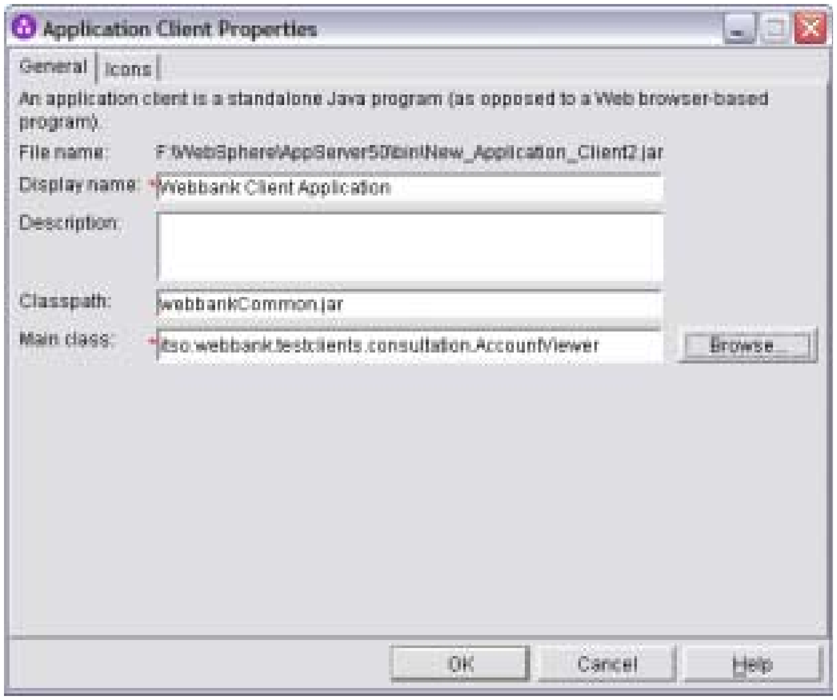
   – Classpath

     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.consultation.AccountViewer class.

6. Save the client application module as E:\webbankSample\jars\webbankConsultationClientApp.jar by selecting **File** -> **Save**.
7. Repeat steps 1 through 6 for the second client application (Transfer).



An application client is a standalone Java program (as opposed to a Web browser-based program).

File name: F:\WebSphere\AppServer50\bin\New_Application_Client2.jar

Display name: *Webbank Client Application

Description:

Classpath: webbankCommon.jar

Main class: *itso.webbank.testclients.consultation.AccountViewer

Figure 19-11 Setting the application client general properties

You have now completed the packaging of the application client. .

19.7 Importing Existing Modules in the AAT

Any module or application *created* with the AAT conforms to the 1.3 level of the specification. However, you can import both J2EE 1.2 and 1.3 applications or modules in the AAT and edit them. The J2EE 1.3 requires that any combination of modules can be packaged in an enterprise application. You can for example mix EJBs at 1.1 and 2.0 levels.

This can easily be done by selecting the specific module entry (such as EJB modules) and invoking Import from the contextual menu.

Note: When you import an EAR/WAR/JAR file which was built for another application server, you might be surprised that this file will be modified when you save it with the AAT. The AAT simply adds IDs to each standard entry (for example, `<ejb-jar` **`id="ejb-jar_ID"`**`>`). IDs are used to link the standard and extensions/bindings deployment descriptors. Those IDs are harmless, and do not make the files non-standard. You can still deploy the modified file to a non-WebSphere J2EE server.

19.8 Declaring environment variables

Environment variables are stored in 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). The same applies for Web modules or application client modules.

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 19-12 on page 805 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.

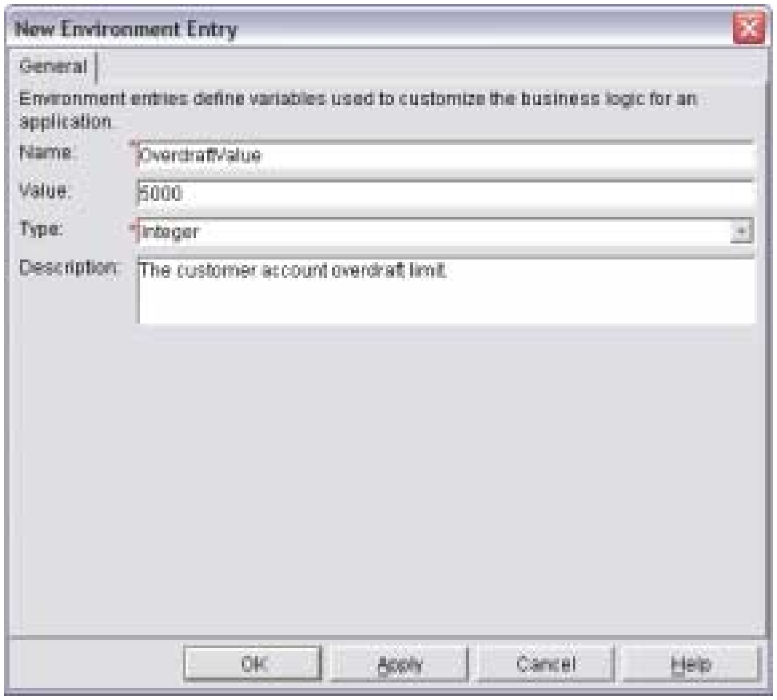
   – Description

     Optionally, you can provide a short description of the value.

Example 19-8 shows how the OverdraftValue environment entry is declared in the ejb-jar.xml file.

Example 19-8 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>
```



Environment entries define variables used to customize the business logic for an application.

Name: OverdraftValue
Value: 5000
Type: Integer
Description: The customer account overdraft limit.

Figure 19-12 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 19-9.

Example 19-9 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");
```

19.9 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.

You can create remote or local references to an EJB, depending whether you use its local interfaces or remote interfaces. Local references should be used whenever the the client of the EJB and the EJB itself runs in the same JVM. Local references are also critical when you use container-managed relationships (CMR): when you create a CMR between two EJBs, local references must be created between those EJBs to “navigate” the relationship.

For the webbank application, you should use EJB references only for the J2EE application clients (which run in a separate JVM). All other references are local.

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.

19.9.1 Using “remote” references

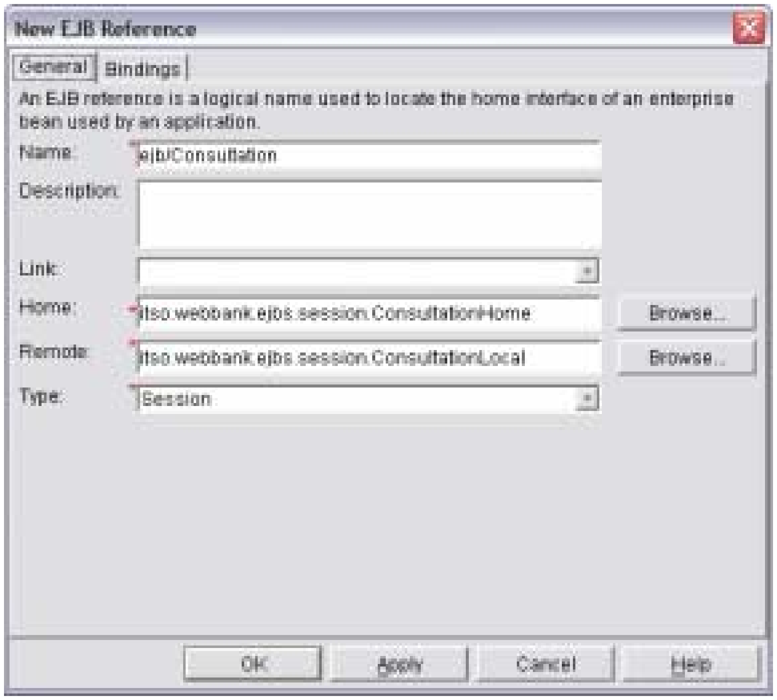
The consultation application client uses the Consultation session bean. Therefore, at packaging time, we need to define an EJB reference to create the "ejb/Consultation" environment entry in the J2EE client application JNDI namespace. At deployment time (see 19.2.3, “Binding EJB references to EJB JNDI names” on page 700), we link this EJB reference to the actual JNDI name of the Consultation EJB. To look up the Consultation EJB, we could then use the code snippet given in Example 19-10. Note that we have to use the narrow() call to retrieve the home object. It is not allowed to do a direct cast on the result of the lookup call (this is only possible and safe via a local reference).

Example 19-10 Lookup up an EJB via a remote reference

```
...
try {
   InitialContext ic = new InitialContext();
   Object o = ic.lookup(Constants.CONSULTATION_JNDINAME);
   consultationHomeProxy =
      (ConsultationHome)
         PortableRemoteObject.narrow(o,ConsultationHome.class);
} catch (NamingException ne)
...
```

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 19-13 on page 807 starts.



An EJB reference is a logical name used to locate the home interface of an enterprise bean used by an application.

Name: ejb/Consultation
Description:
Link:
Home: itso.webbank.ejbs.session.ConsultationHome
Remote: itso.webbank.ejbs.session.ConsultationLocal
Type: Session

Figure 19-13 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/Consultation".
   - Description

     Optionally, provide a description of this EJB reference.
   - Link

     If the target EJB is located in the same EJB module, 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 or a different EJB module in the same EAR file.

– Home

The type of the target enterprise bean home interface, such as itso.webbank.ejbs.ConsultationHome.

– Remote

The type of the target enterprise bean remote interface, such as itso.webbank.ejbs.Consultation.

– 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 19-3, repeat steps 1 to 4 to create all necessary EJB references in the Webbank application.

Example 19-11 shows the Consultation EJB reference entry in the EJB deployment descriptor.

Example 19-11 EJB reference deployment descriptor entry

```
<ejb-ref id="EjbRef_1">
      <description></description>
      <ejb-ref-name>ejb/Consultation</ejb-ref-name>
      <ejb-ref-type>Session</ejb-ref-type>
      <home>itso.webbank.ejbs.session.ConsultationHome</home>
      <remote>itso.webbank.ejbs.session.Consultation</remote>
</ejb-ref>
```

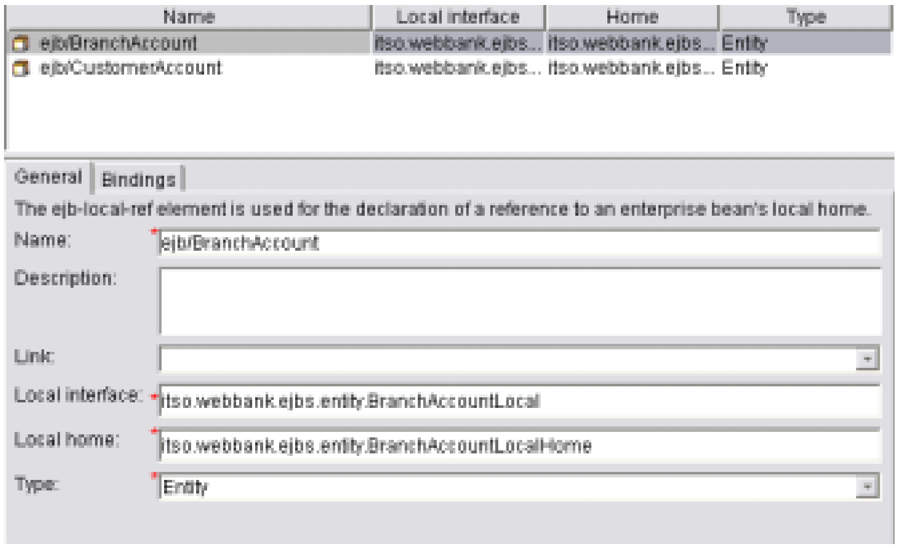
You should use the information supplied below to create all references for the Webbank application.

Table 19-3 EJB references list

From	To	EJB reference name
Consultation Client Application	Consultation Session Bean	ejb/Consultation
Transfer Client Application	Transfer Session Bean	ejb/Transfer

19.9.2 Using Local References (EJB 2.0 only)

You declare a local reference the same way you declare a remote reference, except you point to the local home and local interfaces, as shown in Figure 19-14 below. When you use a local reference, the container performs multiple optimizations. Especially, parameters will be passed by reference as opposed to by value. This has great impact on the application performance, especially if you use large or/and complex objects as parameters.



Name	Local interface	Home	Type
ejb/BranchAccount	itso.webbank.ejbs...	itso.webbank.ejbs...	Entity
ejb/CustomerAccount	itso.webbank.ejbs...	itso.webbank.ejbs...	Entity

General | Bindings

The ejb-local-ref element is used for the declaration of a reference to an enterprise bean's local home.

Name:	ejb/BranchAccount
Description:	
Link:	
Local interface:	itso.webbank.ejbs.entity.BranchAccountLocal
Local home:	itso.webbank.ejbs.entity.BranchAccountLocalHome
Type:	Entity

Figure 19-14 Local reference definition

A local reference is declared as follows in the ejb-jar.xml file:

```
<ejb-local-ref id="EJBLocalRef_2">
   <description>A local ref to the Branch Account EB.</description>
   <ejb-ref-name>ejb/BranchAccount</ejb-ref-name>
   <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>
      itso.webbank.ejbs.entity.BranchAccountLocalHome
   </local-home>
   <local>itso.webbank.ejbs.entity.BranchAccountLocal</local>
</ejb-local-ref>
```

Use the list of local references provided in Table 19-4 below to create all EJB references for the Webbank Application.

Table 19-4 EJB local references list

From	To	EJB reference name
Consultation Session bean	BranchAccount entity bean	ejb/BranchAccount
Consultation Session bean	CustomerAccount entity bean	ejb/CustomerAccount
Consultation Session bean	Customer entity bean	ejb/Customer
Transfer Session bean	BranchAccount entity bean	ejb/BranchAccount
Transfer Session bean	CustomerAccount entity bean	ejb/CustomerAccount
TransferStateless Session bean	BranchAccount entity bean	ejb/BranchAccount
TransferStateless Session bean	CustomerAccount entity bean	ejb/CustomerAccount
Customer entity bean (created for CMR support)	CustomerAccount entity bean	ejb/CustomerAccount
CustomerAccount entity bean (created for CMR support)	Customer entity bean	ejb/Customer
Transfer Servlet	Transfer Session Bean	ejb/Transfer
Consultation Servlet	Consultation Session Bean	ejb/Consultation

19.10 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.

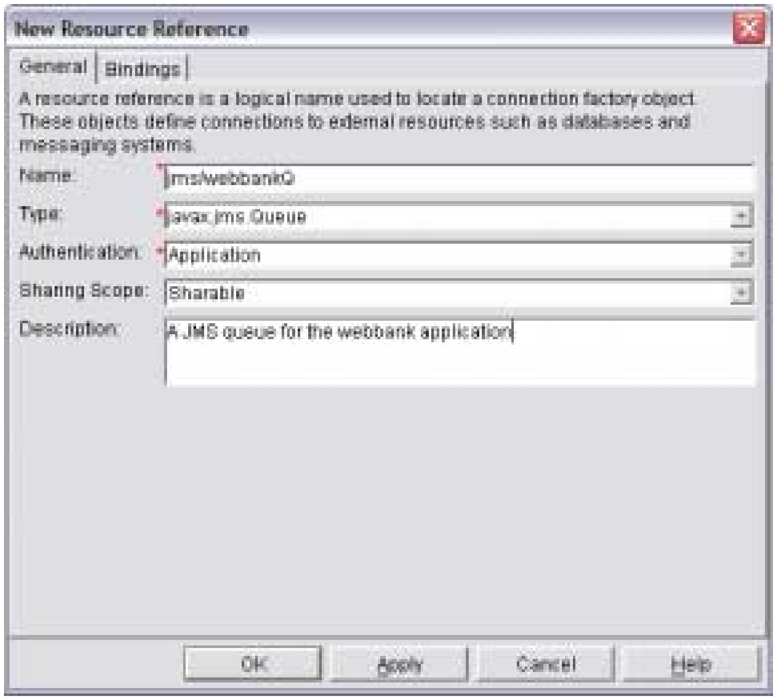
The Transfer EJB uses a resource reference to post a message on a logging queue whenever a large transfer is completed. The snippet of code in Example 19-12 can be used to look up the message queue.

Example 19-12 Retrieving a resource reference

```
javax.naming.Context ic = new javax.naming.InitialContext();
webbankQ = (javax.jms.Queue)
   ic.lookup ("java:comp/env/jms/webbankQ");
```

You can create a 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 19-15 on page 811 appears.



A resource reference is a logical name used to locate a connection factory object. These objects define connections to external resources such as databases and messaging systems.

Name: jms/webbankQ
Type: javax.jms.Queue
Authentication: Application
Sharing Scope: Sharable
Description: A JMS queue for the webbank application

Figure 19-15 Creating a resource reference

2. Provide the following information to create the JMS resource reference:
   – Name

The resource reference name, for example "jms/webbankQCF".

- 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 JMS system. 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.

- Shareable

  This value specifies whether connections obtained through the given resource manager connection factory reference can be shared. The value of this element must be Shareable or Unshareable. By default, connections are assumed to be Shareable.

Example 19-13 shows the resource reference entry in the EJB deployment descriptor.

Example 19-13 Resource reference deployment descriptor entry

```
<resource-ref id="ResourceRef_1037018460168">
   <description>The queue connection factory for use by the Webbank 
Application.</description>
   <res-ref-name>jms/webbankQCF</res-ref-name>
   <res-type>javax.jms.QueueConnectionFactory</res-type>
   <res-auth>Container</res-auth>
   <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

See also Chapter 18, “Configuring WebSphere resources” on page 623.

19.11 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.

19.11.1 Transaction attributes overview

The possible transaction attribute values are listed in Table 19-5.

Table 19-5 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).
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.

19.11.2 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.

19.12 IBM EJB extensions: EJB Caching Options

Multiple IBM extensions are available to influence the behavior of the EJB container at runtime. This section covers EJB instances caching options for entity and stateful session beans.

19.12.1 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 19-14. 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 19-14 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 19-15.

Example 19-15 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.

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 19-16.

Example 19-16 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

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 19-6 to choose the right combination.

Table 19-6 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"/>
```

19.12.2 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.

19.12.3 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
StatefulBeanO(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).

19.13 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 19-7 provide some options for handling such transactions.

Table 19-7 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.
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.

19.14 EJB Access Intents

Access Intents are used by the persistence manager to optimize the access to relational data. Version 5.0 brings new flexibility in the way to manage access to data, and drive things such as transaction isolation levels or database locks. Different levels of service are available epending if you use the EJB 1.1 or EJB 2.0 container.

For EJBs at 1.1 level, you can set the transaction isolation level, as well as marking methods as read-only, tagging methods as being used to drive a database update. You can also define which type of concurrency control you want to use (pessimistic or optimistic modes).

For EJBs at 2.0 level, you must use Access Intents. Access intent policies are specifically designed to supplant the use of isolation level and read-only, method-level modifiers found in the extended deployment descriptor for EJB Version 1.1 enterprise beans. You cannot specify isolation level and read-only modifiers for EJB Version 2.0 enterprise beans. The WebSphere persistence manager uses access intent hints to make decisions about isolation level, cursor management, and more.

19.14.1 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 19-8 shows a mapping between EJB and DB2 isolation levels.

Table 19-8 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.

19.14.2 Concurrency Control Overview

Concurrency control is the management of contention for data resources. A concurrency control scheme is considered pessimistic when it locks a given resource early in the data-access transaction and does not release it until the transaction is closed. A concurrency control scheme is considered optimistic when locks are acquired and released over a very short period of time at the end of a transaction.

The objective of optimistic concurrency is to minimize the time over which a given resource would be unavailable for use by other transactions. This is especially important with long-running transactions, which under a pessimistic scheme would lock up a resource for unacceptably long periods of time.

WebSphere uses *an overqualified update scheme* to test whether the underlying data source has been updated by another transaction since the beginning of the current transaction. With this scheme, the columns marked for update and their original values are added explicitly through a WHERE clause in the UPDATE statement so that the statement fails if the underlying column values have been changed. As a result, this scheme can provide column-level concurrency control; pessimistic schemes can control concurrency at the row level only.

Optimistic schemes typically perform this type of test only at the end of a transaction. If the underlying columns have not been updated since the beginning of the transaction, pending updates to container-managed persistence fields are committed and the locks are released. If locks cannot be acquired or if some other transaction has updated the columns since the beginning of the current transaction, the transaction is rolled back: All work performed within the transaction is lost.

Pessimistic and optimistic concurrency schemes require different transaction isolation levels. Enterprise beans that participate in the same transaction and require different concurrency control schemes cannot operate on the same underlying data connection unless the connection is able to change its isolation lavel on an individual-query basis. Some, but not all, JDBC drivers can do this. For those JDBC drivers that cannot, mixing concurrency controls requires the use of multiple connections within a transaction.

Whether or not to use optimistic concurrency depends on the type of transaction. Transactions with a high penalty for failure might be better managed with a pessimistic scheme. For low-penalty transactions, it is often worth the risk of failure to gain efficiency through the use of an optimistic scheme.

19.14.3 Setting Isolation levels (EJB 1.1 only)

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.

19.14.4 Marking methods as Read-only (EJB 1.1 only)

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 STORE operation at the end of a transaction (regardless of the EJB caching option you are using, A, B, or C).

Note: For EJBs at the 2.0 level, this is taken in account in the concrete implementation of the bean. A bean will be marked as “dirty” whenever a setMethod is used. Only dirty beans will be resynchronized with the underlying datastore.

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.

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 19-17 shows lines inserted in the ibm-ejb-jar-ext.xmi file when you use this flag.

Example 19-17 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>
```

19.14.5 Marking Methods with For Update flag (EJB 1.1 Only)

This flag can be used to force SELECT calls to be done with a FOR UPDATE option. This will set a stronger lock at the database level: this lock will allow updates to data. If you do a simple SELECT, then a lock escalation has to be done by the database if you try to do an UPDATE call in the same transaction. This will typically happen if you write something like:

```
BranchAccountLocal branchAcct = findBranchAccount(branchAcctKey);
branchAcct.debit(amountCent);
```

The find call will typically trigger a SELECT query, taking a typical read lock on the database. The debit method, however, needs to update data, and therefore needs a stronger lock to preserve data integrity. If another transaction has a lock on the same row or table, the current transaction will have to wait. If the other transaction does a similar thing (both want to escalate their locks), you are in a deadlock situation.

By using the forUpdate flag, you take right way a write lock instead of a read one. No lock escalation needs to happen, which prevents potential deadlocks.

This setting is accessible from the Method Extensions entry (for entity beans only). To mark a method with a find-for-update flag:

1. Select one method in the methods list.
2. Check the **Access intent** option.
3. Set the IntentType property to **Update**.
4. Click **Apply**.

Example 19-17 shows lines inserted in the ibm-ejb-jar-ext.xmi file when you use this flag.

Example 19-18 Access intents deployment descriptor entry

```
<accessIntents xmi:id="AccessIntent_2" intentType="UPDATE">
   <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>
```

19.14.6 Using EJB 2.0 Access Intents

Access Intents policies are new to WebSphere 5.0. They let you define in a very flexible and powerful way how relational data will be accessed, should you be using BMP or CMP entity beans. Access intents cover the problems described above such as the transaction isolation level or the find for update flag.

Access Intents Policies Overview

Seven access intent policies are available. They cover a wide number of ways to access data. They are summarized in the table below:

Access Intent Policy	**Concurrency control**	**Access type**	**Transaction isolation level**	**Notes**
wsPessimisticRead	pessimistic	read	read committed	Read locks are held for the duration of the transaction. Updates are not permitted; the generated SELECT query does not include FOR UPDATE
wsPessimisticUpdate	pessimistic	update	For Oracle, read committed. Otherwise, repeatable read	The generated SELECT FOR UPDATE query grabs locks at the beginning of the transaction
wsPessimisticUpdate-Exclusive	pessimistic	update	serializable	SELECT FOR UPDATE is generated; locks are held for the duration of the transaction

Access Intent Policy	Concurrency control	Access type	Transaction isolation level	Notes
wsPessimisticUpdate-NoCollision	pessimistic	update	read committed	The generated SELECT query does not include FOR UPDATE. **No locks are held, but updates are permitted. Do not use this in production.**
wsPessimisticUpdate-WeakestLockAtLoad (DEFAULT VALUE)	pessimistic	update	repeatable read	For Oracle, this is the same as wsPessimisticUpdate. Otherwise, the generated SELECT query does not include FOR UPDATE; locks are escalated by the persistent store at storage time if updates were made.
wsOptimisticRead	optimistic	read	read committed	
wsOptimisticUpdate	optimistic	update	read committed	Generated overqualified-update query forces failure if CMP column values have changed since the beginning of the transaction

There are two critical questions you should ask yourself when using access intents:

- *At which point in my transaction do I access data?* This is critical to choose on which method you must set the access intent.
- *How do I want to access data?* This is critical to choose which is the best access intent to apply on the method.

Choosing where to apply the access intent

This is critical since it is at this point that the WebSphere persistence manager will decide which access intent to use. Let’s take an example to illustrate this point. The Consultation session bean obtains the CustomerAccount balance using the following getBranchAccountBalance:

```
int getCustomerAccountBalance () {
```

```
...
custAcct = (CustomerAccountLocal) custAcctHome.findByPrimaryKey(custAcctKey);
return custAcct.getBranchBalance();
}
```

Now, say you have applied **AccessWriteIntent1** on the findByPrimarykey() method of the CustomerAccount bean and **AccessReadIntent2** on the getBranchBalance() method. Since the first access to the database in the transaction started by the call to getCustomerAccountBalance() is done by the findByPrimaryKey method, then **AccessWriteIntent1** is used for *all calls* within the transaction. AccessReadIntent2 will be ignored by the persistence manager and therefore useless in this case.

This may be fine or not depending on what you want to achieve. The critical point here is that you could use the findPrimaryKey method in read or/and write transactions. If you use it in a write transaction, you probably want to execute it with for example a PessimisticUpdate intent. If you access data only for reading it, this would be an overkill. There are two main solutions to this problem.

The simplest one is to have two findByPrimaryKey methods, one for read access and one for write access, with the proper access intent applied. If you look at the Webbank sample, you will see that both the BranchAccount and CustomerAccount have a customFindByPrimaryKeyForRead() method and the standard findByPrimaryKey() method. The customFindByPrimaryKeyForRead() method is a custom EJB home method which delegates to the default findByPrimaryKey() method.

The customFindByPrimaryForRead method is declared like this on the home interface:

```
public BranchAccountLocal customFindByPrimaryKeyforRead(
      itso.webbank.ejbs.entity.BranchAccountKey primaryKey)
      throws javax.ejb.FinderException;
```

It is implemented as follows in the bean implementation class:

```
public BranchAccountLocal ejbHomeCustomFindByPrimaryKeyforRead(
      itso.webbank.ejbs.entity.BranchAccountKey primaryKey)
      throws javax.ejb.FinderException
{
   return (
      (BranchAccountLocalHome) getEntityContext()
         .getEJBLocalHome())
         .findByPrimaryKey(
      primaryKey);
}
```

You would set the access intent of this finder, say to WSOptimisticRead, and use it in all read transactions. However, you would use the standard findByPrimaryKey method whenever you know you are accessing an entity bean to update data. The default access intent (wsPessimisticUpdate-WeakestLockAtLoad) is probably fine in most cases for the findByPrimaryKey() method (as well as as for other methods).

The other solution is to trick the container by running findByPrimaryKey in its own transaction. This is done by applying a RequiresNew transaction flag on it. Let's look at the sample above again

1. The getCustomerAccountBalance method starts a new transaction
2. FindByPrimaryKey is called. The current transaction is paused, the findByPrimary method executes within its own transaction and therefore own access intent. The call to findByPrimaryKey() returns an unhydrated instance (not activated/no loaded).
3. The transaction initiated by the session bean resumes.
4. getBalance() is called on the instance returned by findByPrimaryKey. The instance is hydrated and the access intent specified for this method is used. Any other method calls within the transaction will execute with the same access intent.

Note: WebSphere Application Server, Enterprise edition provides an extension to access intents called Application Profiles, which handles the problem mentioned above in a more elegant and powerful way. Application profile will allow you to externally specify a set of tasks (i.e. a flow of calls in your code), and specify which access intent should be used for a specific task. For additional information on this, please refer to <Insert reference to V5 Enterprise Handbook>

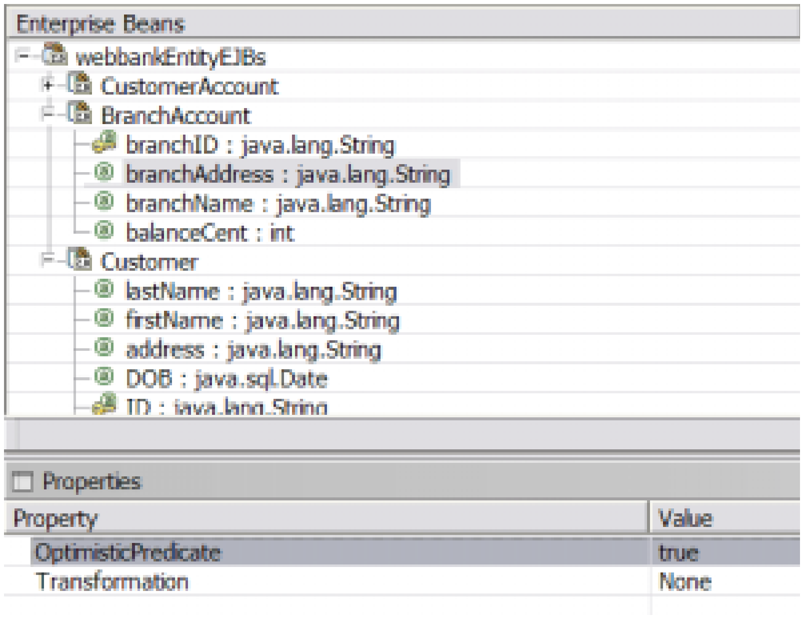
Choosing the right access intent

The main rule is : keep it simple. Start with the default settings (weakestLockAtLoad), and work from there. Specifying access intents on all your business methods could lead to a configuration, debugging, and maintenance nightmare. Also, choose access intents wisely.

- Access intents can be applied to your business methods, to the findbyPrimaryKey() method, as well as the create and remove method. As much as possible, avoid other methods.
- Make sure that no method is configured with more than one access intent policy. Applications that are misconfigured in this way will not be runnable until the configuration errors are fixed. **(Logan: Which Exception will I receive ?)**

- For entity beans that are backed by tables with nullable columns, use optimistic policies with caution. **Nullable columns are automatically excluded from overqualified updates at deployment time**. This means that at commit time, those columns will not be used in the update statement to check whether the data has changed or not.
  Therefore, concurrent changes to a nullable field might result in lost updates. If you use IBM WebSphere Studio Application Developer product, you can set a property on each enterprise bean attribute called *OptimisticPredicate*, as shown in Figure 19-16 below. You can change this property by editing the data mappings of your EJBs. When this property is set, the column, even if it is nullable, will be reflected in the overqualified update statement that is generated in the deployment code to support optimistic policies.

Tip: If you want to check which SQL code is executed for an optimistic update, check the **storeUsingOCC** method in the **<beanname>FunctionSet** generated class.


```
Enterprise Beans
- webbankEntityEJBs
  + CustomerAccount
  - BranchAccount
      branchID : java.lang.String
      branchAddress : java.lang.String
      branchName : java.lang.String
      balanceCent : int
  - Customer
      lastName : java.lang.String
      firstName : java.lang.String
      address : java.lang.String
      DOB : java.sql.Date
      ID : java.lang.String
```

Properties

Property	Value
OptimisticPredicate	true
Transformation	None

Figure 19-16 Optimistic Predicate Property in WebSphere Studio Application Developer

- A method that is configured with a read-only policy that causes a bean to be activated can cause problems if updates are attempted within the same transaction. This is the detected by the container: changes are not be committed, and an exception is be thrown. Otherwise, data integrity could be

compromised.
NEED EXAMPLE

19.14.7 Using read-ahead hints

Read-ahead schemes enable applications to minimize the number of database roundtrips by retrieving a working set of container-managed persistence (CMP) beans for the transaction within one query. Read-ahead involves activating the requested CMP beans and caching the data for their related beans, which ensures that data is present for the beans that are most likely to be needed next by an application.

A read-ahead hint is a canonical representation of the related beans that are to be read. It is associated with a finder method for the requested bean type, which must be an EJB 2.x-compliant CMP entity bean. Currently, only findByPrimaryKey methods can have read-ahead hints. Only beans related to the requested beans by a container-managed relationship (CMR), either directly or indirectly through other beans, can be read ahead.

Read-ahead hints can be set only through the Add Access Intent wizard of the IBM WebSphere Studio Application Developer product. In the wizard, the Read Ahead Hint check box is enabled only with access intent policies with optimistic concurrency.

Read-ahead is limited to optimistic policies because locking persistent data store for all beans represented in the hint would be more likely to cause lock conflicts, and optimistic policies do not obtain locks until immediately before the database operation.

A read-ahead hint takes the form of a character string. You do not have to provide the string; the wizard generates it for you based on CMRs defined for the bean. The following example is provided as supplemental information only.

Suppose a CMP bean type A has a finder method that returns instances of bean A. A read-ahead hint for this method is specified using the following notation:
RelB.RelC; RelD

Interpret the preceding notation as follows:

- Bean type A has a CMR with bean types B and D.
- Bean type B has a CMR with bean type C.

For each bean of type A that is retrieved from the database, its directly-related B and D beans and its indirectly-related C beans are also retrieved. The order of the retrieved bean data columns in each row of the result set is the same as their order in the read-ahead hint: an A bean, a B bean (or null), a C bean (or null), a D bean (or null). For hints in which the same relationship is mentioned more than once (for example, RelB.RelC;RelB.RelE), a bean's data columns appear only once, at the position it first appears in the hint.

The tokens shown in the notation (RelB and so on) must be CMR field names for the relationships as defined in the deployment descriptor for the bean. In indirect relationships such as RelB.RelC, RelC is a CMR field name defined in the deployment descriptor for bean type B.

19.14.8 Tracing Access Intents behavior

Add the following trace specification to your server:

```
com.ibm.ejs.container.*=all=enabled:
com.ibm.ejs.persistence.*=all=enabled:
com.ibm.ws.appprofile.*=all=enabled:
```

19.15 EJB 1.1 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.

19.16 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.

19.16.1 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 (also known as 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.

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.

19.16.2 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.

19.16.3 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.

19.16.4 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.

19.16.5 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.

19.16.6 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>\<application name>\<webmodulename>.

JSP attributes can be set in AAT under a Web module’s Assembly Property Extensions entry.

19.17 IBM EAR Extensions: Sharing Session Contexts

By default, it is not possible to share session context among multiple web apps. This flag allows to override the behavior.

Author Comment: need to try and expand on this one.

19.18 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.

19.19 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.

19.19.1 Where should common classes go?

There are basically 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 **only** by this Web module.

3. As loose classes within an EJB module.

   Although these classes are visible from within other modules that point on the EJB JAR via a manifest Class-Path entry, 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.

19.20 Summary

In this chapter, we have covered how to package an enterprise application using the Application Assembly Tool. We have also covered the various J2EE extensions provided by the WebSphere Application Server. In the next chapter, we get you to deploy the Webbank Application using the administrative console.

20

Deploying an application

In this chapter we introduce/provide/describe/discuss ...

In this chapter:
In this chapter, the following topics are discussed/described:
This chapter provides/describes/discusses/contains the following:

- ...
- ...
- ...
- Sample level 2 “n.n” chapter heading
  (created by **Special > Cross-Reference > Format: Head > Insert**)
- Sample next level 2 heading

Part 4

Managing WebSphere

Note to Author: Optionally, describe the book part here. If you are not using Part files, you need to restart the page numbering in the first chapter file of your book:

- In FrameMaker 5.5: **Open .book > select first chapter> File > Set Up File > Page Numbering: > Restart at 1 > Set**
  - Now delete the three Part files (p01, p02 and p03) from your book: **Open .book > select a file > File > Rearrange Files > Delete > Done**
- In FrameMaker 6.0: **Open .book > select first chapter> Format > Document > Numbering > Page “tab” > select First Page # radio button and set Page # to 1 > Set**
  - Now delete the three Part files (p01, p02 and p03) from your book: **Open .book > select a part file > Edit > Delete File from Book**

In this part we introduce/provide/describe/discuss...

21 Monitoring and tuning your runtime environment

In this chapter we describe the updated Performance Monitoring Infrastructure (PMI) found in IBM WebSphere Application Server V5.0 as well as introduce the support of the Performance Data Framework defined in the J2EE management specification.

This chapter will discuss the following topics:

- Performance Monitoring Infrastructure
- Performance monitoring servlet
- Tivoli Performance Viewer (formerly known as the Resource Analyzer)
- PMI Request Metrics
- Monitoring the IBM HTTP Server
- Tuning techniques for the WebSphere Application Server runtime environment
- Tivoli Performance Viewer Advisor

21.1 Gathering and displaying application server performance data

Table 21-1 shows the types of data that can be collected, the required actions to collect it, and how to view it.

Table 21-1 Performance data collection and viewing

Collection configuration	Methods for configuration	Viewed with
Enable PMI service. ► Required for all resource monitoring. ► Enables data collection for counters, statistical, and load data tailored to the resource type (as defined by instrumentation level).	Set at application server level ► Admin console: Performance Monitoring Service ► wsadmin	N/A
Configure instrumentation levels. ► Used for all resource monitoring (defaults exist). ► Performance data types collected are assigned impact ratings (high, medium, etc). The instrumentation level filters data collection by impact level. The default is “None” so you must set these to collect data.	Set at application server / resource category level ► Admin console: Performance Monitoring Service ► Tivoli Performance Viewer: File>Current Activity ► wsadmin	► Tivoli Performance Monitor ► Performance Monitoring Servlet
Enable / configure PMI request metrics and filters. ► Collects response time info for the major services used by a request as it moves through the application server.	Set at cell level, applies to every Web and EJB container in the cell. ► Admin console: Troubleshooting ► wsadmin	► Text file (System.out) ► Application Response Measurement (ARM 2.0) compliant agent

Collection configuration	Methods for configuration	Viewed with
Enable JVMPI facility ► Used when you want performance data about the JVM running the application server or node agent.	Set at application server or node agent level ► Admin console: Process Definition / JVM ► wsadmin	???

21.1.1 Viewing data

- Tivoli Performance Viewer
- Performance Monitoring Servlet
- RYO

21.2 Performance Monitoring Infrastructure

The *Performance Monitoring Infrastructure* (PMI) was introduced with WebSphere Application Server V3.5.5 and was included in the WebSphere Application Server V4.0 base product. PMI is a set of packages and libraries designed to assist with gathering, delivering, processing and displaying performance data in WebSphere Application Server runtime components.

PMI uses a client/server architecture. In PMI terms, a server is any application or runtime component that uses the PMI API to collect performance data. Servers can include application servers, Web servers, and Java applications. 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 PMI components and infrastructure have been extended and updated in WebSphere Application Server V5.0 to support the new management structure and to comply with the Performance Data Framework of the J2EE Management Specification.

PMI is composed of components for collecting performance data on the application server side and components for communicating between runtime components and between the clients and servers. The primary PMI components and related Management Beans (MBeans) are illustrated in Figure 21-1.

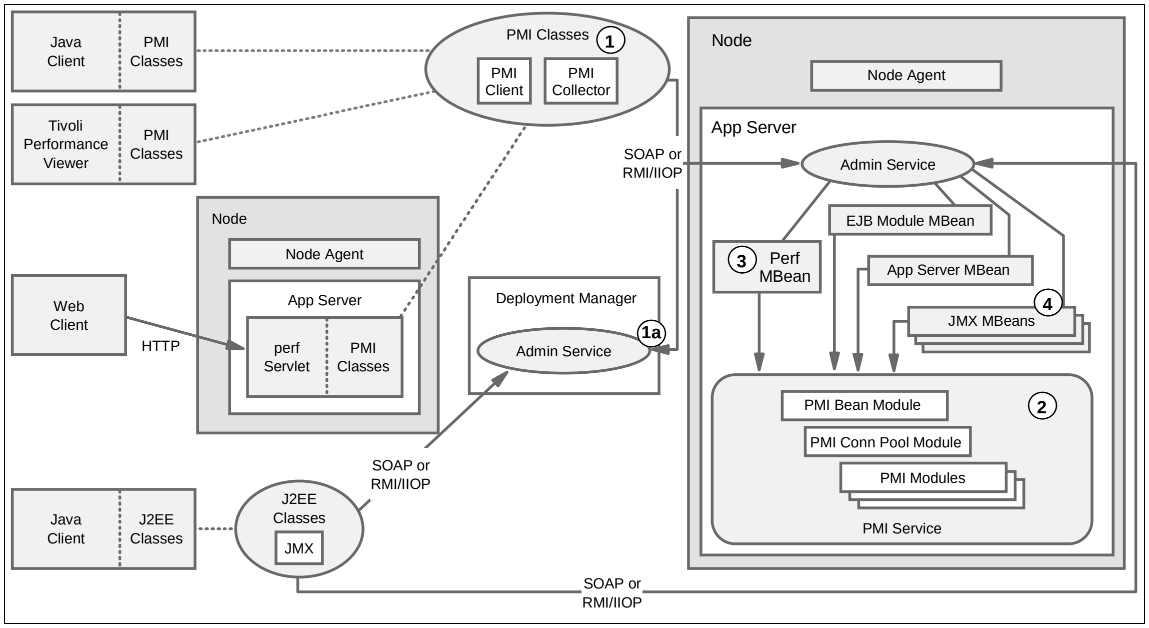


Figure 21-1 Performance Monitoring components

1. PMI Client API with its communications implementation, the PMI Collector. The PMI Client API initially contacts the admin service of the Deployment manager (1a) to get a list of nodes, servers and MBeans for the entire cell.
2. PMI Service consisting of PMI modules for collecting performance data and methods for instrumenting and retrieving the data from the runtime components. This service contacts the PMI modules upon application server startup, depending on the current configuration.
3. PerfMBean that is a JMX management bean used to extract performance data from the PMI modules to the PMI Collector of the PMI Client API.
4. Extensions of the standard JMX MBeans (used for managing components and settings) to support management of performance data. This enables a JMX based client to retrieve performance data. See “Java Management Extensions (JMX)” on page 913 for a description of the JMX framework.

The J2EE classes and PMI classes can use either the RMI over IIOP or SOAP protocol to communicate with the admin service. In a single server environment the classes connect to the admin service of each individual application server in order to collect performance data. In a network deployment environment the client can choose to connect to the deployment manager first to retrieve a list of nodes, servers and MBeans in the cell. Performance data retrieval is subsequently performed in the same way for the two environments.

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 of how the PMI and PMI client interface is organized and works together with Tivoli Performance Viewer.

21.2.1 Performance data classification

Performance data is classified into the following five types in compliance with the J2EE management specification:

- **Count statistic interface:** Specifies standard count measurements. It consists of a single numeric value and is used to represent data such as counts and sizes. Examples of count statistics include number of times beans were created, number of calls retrieving an object from the pool and used memory in the JVM runtime.
- **Boundary statistic interface:** Specifies standard measurements of the upper and lower limits of the value of an attribute. This classification is currently not being used by PMI.
- **Range statistic interface:** Specifies standard measurements of the lowest and highest values an attribute has held as well as its current value. These values can be used for obtaining number of concurrent invocations to a call method, average percent of the pool that is in use or the number of requests that are concurrently processed by the ORB.
- **Bounded range statistic interface:** Extends the Range Statistic and Boundary Statistic interfaces and provides standard measurements of a range that has fixed limits. Examples of uses range number of free connections in the pool, total memory in JVM runtime and average number of threads in pool.
- **Time statistic interface:** Specifies standard timing measurements for a given operation. Examples of time statistic are average response time in milliseconds on the bean methods (home, remote, local) and average connection time in milliseconds until a connection is granted.

Note: The Performance data classifications have been redefined for the WebSphere 5.0 release in order to comply with the J2EE Management Specification. The former classifications (Numeric, Statistical, Load and Group) as well as the new classifications will be available from the PMI Client API for compatibility reasons.

21.2.2 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 cell. This is where the node agent resides in a Deployment Manager environment.
- **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 Tivoli Performance Viewer these are enterprise java beans, database connection pools, J2C connectors, JVM runtime, object request broker, relational resource adapter, servlet session manager, thread pools, transaction manager, and Web applications.
- **Submodule.** A submodule represents a fine granularity resource category under the model. Submodules themselves can contain submodules. An example of a submodule is the ORB thread pool which is a a fine granularity resource category under the thread pool module.
- **Counter**. A counter is a data type used to hold performance information for analysis. Examples of counters include the number of active enterprise beans and the time spent responding to a servlet request,

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.

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 21-2 shows the counter *Avg Method RT* assigned to both the enterprise beans module, and the methods of the Container1.Bean1 instance. Figure 21-2 also shows a hierarchy of data collections that are organized for reporting to the Tivoli Performance Viewer.

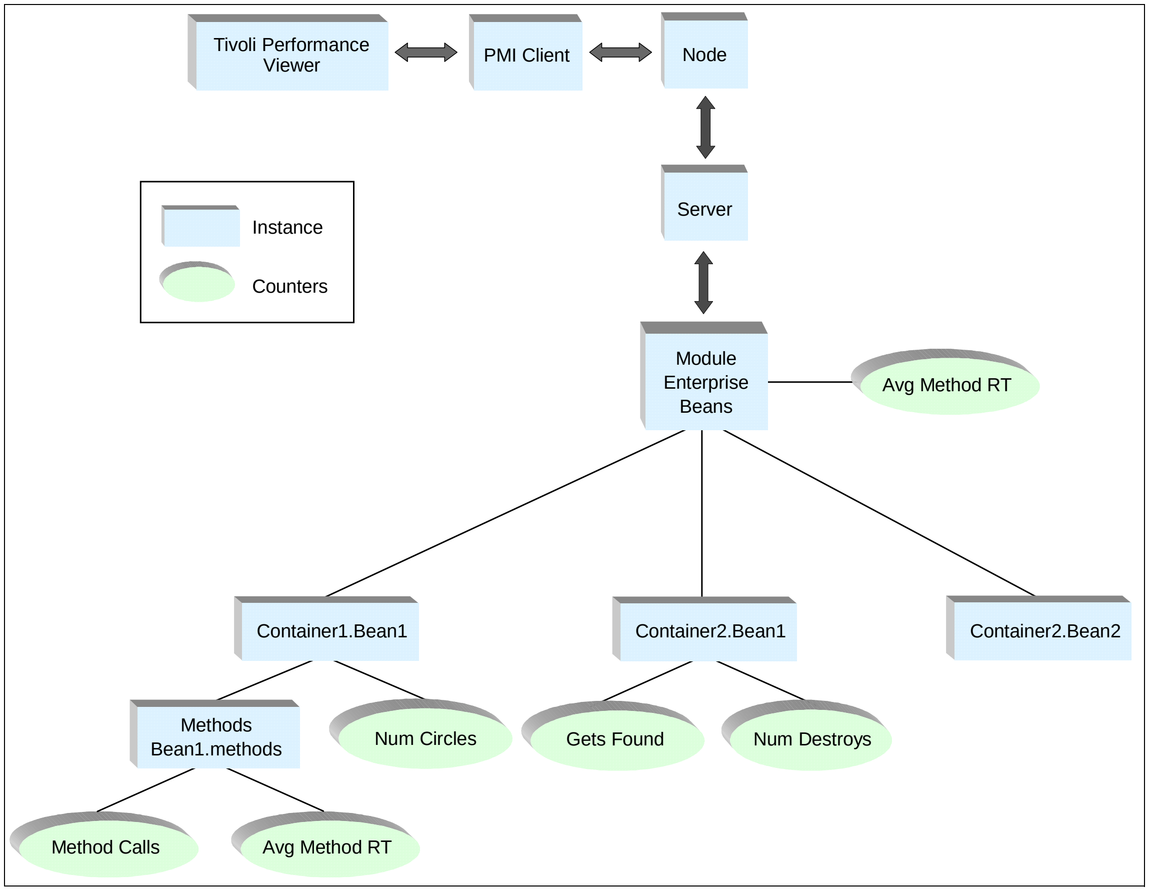


Figure 21-2 Example performance group hierarchy

A subset of counters is available based on the instrumentation level chosen for the particular module or instance. See 21.3.4, “Understanding instrumentation levels” on page 852. Counters are enabled at the module level and can be enabled or disabled for elements within the module. For example, in the figure, if the Module Enterprise Beans module is enabled, its Avg Method RT counter is enabled by default. However, you can then disable the Avg Method RT counter for the Methods Beans1.methods and the aggregate response time reported for the whole Enterprise Beans module will no longer include Methods Bean1.methods data.

21.2.3 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). Note that the modules denoted with an asterisk (*) are new to WebSphere 5.0:

- 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. In addition, all performance data that was previously collected in the JVMPI module are now being collected here. Examples are number of garbage collection calls, number of times a thread waits for a lock and total number of objects allocated in the heap. See "Using JVMPI facility" on page 872 for a description of the JVMPI facility.

- ORB*

  Reports usage information about the Object Request Broker that enables remote clients to instantiate and lookup objects in the application server JVM.

Examples are lookup time for a object reference before method dispatch, total number of requests sent to the ORB and the time it takes for a registered portable interceptor to run.

- 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 time taken for writing session data to the persistent store.

- 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.

- Web services gateway*

  Reports usage information from the Web services gateway facility. Example of performance data collected is number of synchronous and asynchronous requests and responses.

- System data*

  Reports load information for a node. This module is only resident on the node agent, hence cannot be accessed in a single server environment. Examples include CPU utilization and free memory available.

- Workload management*

  Reports information on enterprise bean workload management. Examples include number of WLM clients serviced, server response time and number of concurrent requests.

- Dynamic cache*

Reports usage information from the dynamic cache service. Examples include number of client requests, cache misses and cache hits on disk.

Important: In order to have the system data module available on a AIX 4.3.3 node, the bos.perf.perfstat 4.3.3.0 base fileset available on the AIX 4.3.3 installation disc needs to be installed and applied with patches APAR IY24983 and APAR IY27782 available on
`http://techsupport.services.ibm.com/rs6k/fixdb.html`
AIX 5 ships with these patches already applied.

All the performance data counters are listed in the WebSphere 5.0 InfoCenter article “Performance data organization”.

Applications that use the PMI facility are:

- Tivoli Performance Viewer, discussed in 21.3, “Using Tivoli Performance Viewer” on page 848.
- Performance Servlet, discussed in 21.4, “Using performance monitoring servlet” on page 867.

21.2.4 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 5.0 InfoCenter. The “Developing your own monitoring applications” section discusses the PMI client package as well as the JMX MBeans used for writing PMI clients.

21.3 Using Tivoli Performance Viewer

The Tivoli Performance Viewer is a stand-alone runtime performance monitor for WebSphere 5.0. It provides a Graphical User Interface (GUI) console that is available on Windows and UNIX platforms. Tivoli Performance Viewer 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.

The tool was formerly known as Resource Analyzer, but was renamed for the WebSphere 5.0 release.

21.3.1 About Tivoli Performance Viewer

The Tivoli Performance Viewer retrieves performance data by periodically polling the PMI Service of the application server that is being monitored. As all application servers host a PMI Service, one connection is created from the Tivoli Performance Viewer to every application server in a cell. In addition, the node agent in a deployment manager environment also hosts a PMI Service for monitoring the performance data of the node agent component. The performance data requested by Tivoli Performance Viewer is provided by the PMI Service component by means of the Perf MBean (stateless session management bean) residing in the application servers MBeanServer. The Perf MBean fetches the performance data from the PMI Service component using a PMI Collaborator component.

You can regulate the impact incurred from data collection by using the Tivoli Performance Viewer or the WebSphere Administrative Console (see 21.3.4, “Understanding instrumentation levels” on page 852). The Tivoli Performance Viewer’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 21-3 for an infrastructure overview.

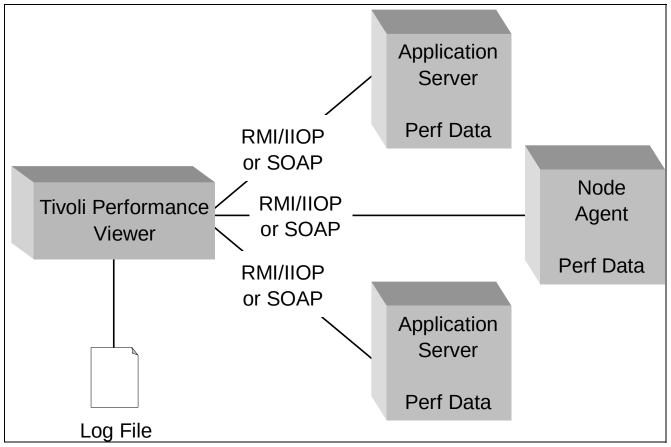


Figure 21-3 Tivoli Performance Viewer infrastructure overview

21.3.2 What can Tivoli Performance Viewer do?

The Tivoli Performance Viewer 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 21.2, “Performance Monitoring Infrastructure” on page 841 for more details about performance data organization.

Tivoli Performance Viewer functionality

Depending on which aspects of performance are being measured, you can use the Tivoli Performance Viewer 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 Tivoli Performance Viewer 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.

21.3.3 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 WebSphere 5.0 InfoCenter article “Performance data organization”. There is a separate table for each resource category. As an example, the ORB counters are shown Table 21-2.

Table 21-2 Counter information for ORB service

Name	**Description**	**Ver-sion**	**Granu larity**	**Type**	**Level**
reference LookupTime	The time (in milliseconds) to look up an object reference before method dispatch can be carried out	5.0	ORB	Time Statistic	Medium
numRequest	The total number of requests sent to the ORB	5.0	ORB	Count Statistic	Low
concurrent Requests	The number of requests that are concurrently processed by the ORB	5.0	ORB	Range Statistic	High
processing Time	The time (in milliseconds) it takes a registered portable interceptor to run	5.0	per interc eptor	Time Statistic	Medium

- Name: Name of the counter as it appears in the Counter Selection window.
- Description: Brief description of the counter content.
- Version: The version of WebSphere Application Server when the counter was introduced into the PMI framework
- Granularity: The unit to which data collection is applied for that counter.
- Type: The performance data classification as described in “Performance data classification” on page 843
- Level: The instrumentation level for which this counter is included. We will cover instrumentation levels in “Understanding instrumentation levels” on page 852.

21.3.4 Understanding instrumentation levels

The resources in a WebSphere cell 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 can also 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:

- Tivoli Performance Viewer's main menu, clicking **File** -> **Current Activity**
- WebSphere Administrative Console on the application server Performance Monitoring Service properties pane.
- **`wsadmin`** command interface

See "Enabling PMI Service and setting the instrumentation level" on page 855 for a detailed description.

21.3.5 Using Tivoli Performance Viewer 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.

Before starting the Tivoli Performance Viewer the PMI Service has to be enabled for the application servers you would like to monitor. See "Enabling PMI Service and setting the instrumentation level" on page 855 for a description on how to enable the PMI Service.

Starting Tivoli Performance Viewer

On Windows platforms do one of the following:

- Click **Start** -> **Programs** -> **IBM WebSphere** -> **Application Server v5.0** -> **Tivoli Performance Viewer.**
- Use the command line by typing:

```
<WAS_HOME>\bin\tperfviewer.bat
```

On UNIX platforms do the following:

- Type into a shell:

```
<WAS_HOME>/bin/tperfviewer.sh
```

By default Tivoli Performance Viewer tries to access the node agent admin service on the local machine (localhost, port 2809 using the RMI protocol). If no node agent is running you get a dialog to try and enter new connection properties. Clicking **Cancel** from this dialog start Tivoli Performance Viewer in a disconnected state and can be used to replay a previously recorded session (see “Record a load scenario” on page 863). If you want to access a remote machine running the deployment manager, node agent or a application server, the command syntax is:

```
tperfviewer.bat|sh [host_name [port_number [conn_type [version]]]]
```

Where:

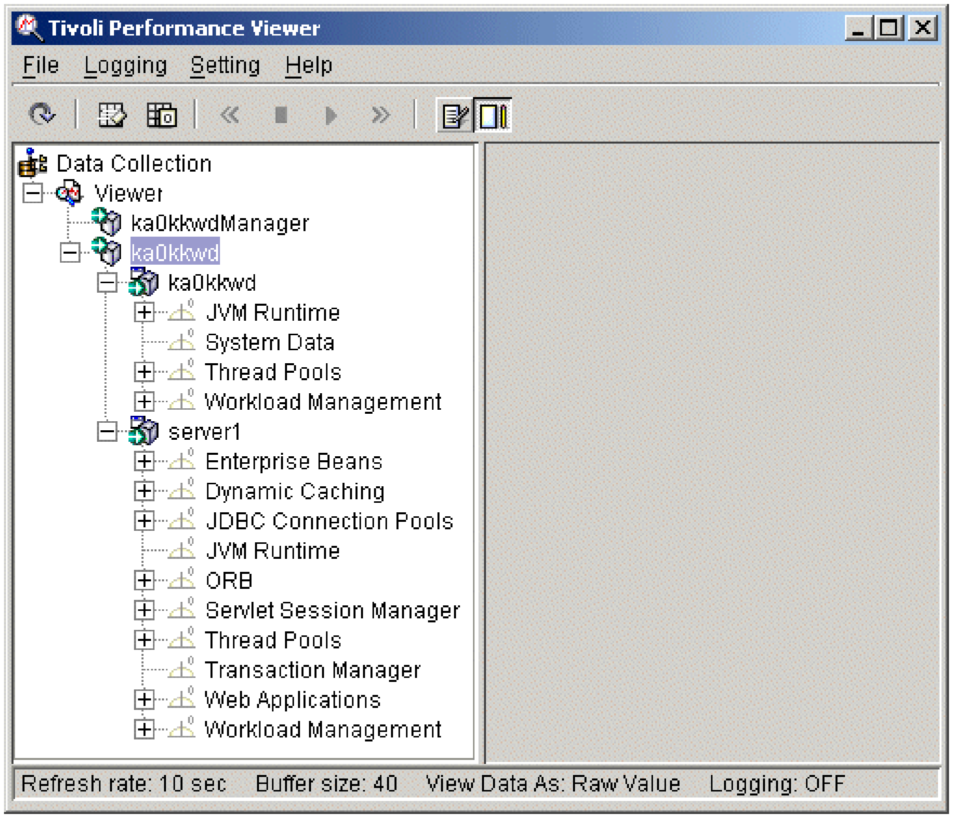
`host_name`	Host name or IP address of the remote system (default: localhost).
`port_number`	Port number of the remote system (default: 2809).
`conn_type`	Type of connection. Either RMI or SOAP (default: RMI).
`version`	The version of WebSphere Application Server establishing a connection to. Currently only the keyword WAS50 is valid (default: WAS50).

For example, if you want to access a remote UNIX machine running the node agent from your Windows system, type in the following:

```
tperfviewer.bat mywas.mydomain.com 2809 RMI
```

If the default port 2809 and RMI protocol is used, they can be left out.

After starting the Tivoli Performance Viewer successfully, you will get the main window shown in Figure 21-4. Tivoli Performance Viewer window consists of the menu bar at the top, the toolbar just below the menu 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.



```
Data Collection
  Viewer
    ka0kkwdManager
    ka0kkwd
      ka0kkwd
        JVM Runtime
        System Data
        Thread Pools
        Workload Management
      server1
        Enterprise Beans
        Dynamic Caching
        JDBC Connection Pools
        JVM Runtime
        ORB
        Servlet Session Manager
        Thread Pools
        Transaction Manager
        Web Applications
        Workload Management
```

Refresh rate: 10 sec   Buffer size: 40   View Data As: Raw Value   Logging: OFF

Figure 21-4 Tivoli Performance Viewer

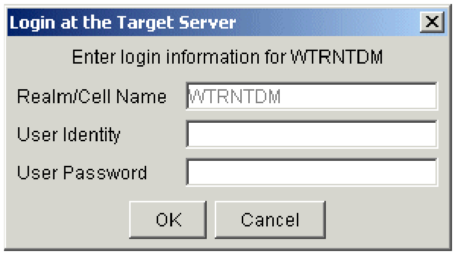
Stop the Tivoli Performance Viewer by selecting **File** -> **Exit** from the main menu. This will bring up a window if the performance monitoring feature was not enabled before starting the Tivoli Performance Viewer. The window asks whether you want to save the monitoring settings in the admin server or not. Choosing **Yes** will save the performance monitoring settings within the application server; choosing **No** will discard them.

Tip: Due to memory requirements, it is recommended that you run the Tivoli Performance Viewer on a separate machine from your deployment managers, node agents and application servers.

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 21-5, 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.



Login at the Target Server

Enter login information for WTRNTDM

Realm/Cell Name	WTRNTDM
User Identity	
User Password	

Figure 21-5 Tivoli Performance Viewer login challenge

Enabling PMI Service and setting the instrumentation level

In order to monitor a resource with Tivoli Performance Viewer or any PMI or JMX client, the PMI service of the application server associated with that resource has to be enabled. The PMI service can be enabled from the Performance Monitoring Service configuration pane in the administration console or using the **`wsadmin`** command interface.

To specify the instrumentation level for resource categories, use the Performance Monitoring Service configuration pane from the administration console or the Performance Monitoring Settings window from the Tivoli Performance Viewer.

You can also use the **`wsadmin`** command to set the instrumentation level, as described in “Using the wsadmin command to configure the PMI service” on page 857.

Using administrative console to configure the PMI service

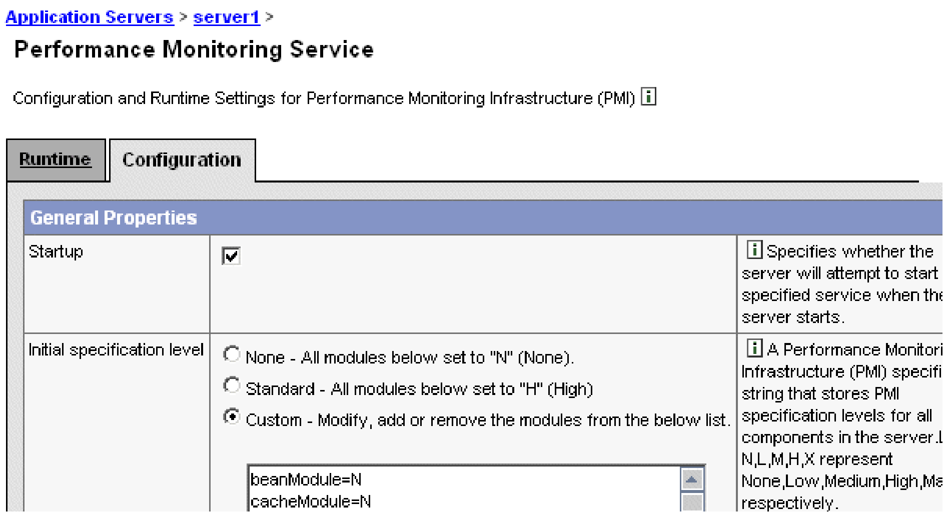
In order to enable the PMI service and set instrumentation levels from the administrative console, open the Performance Monitoring Service properties configuration pane by using the following steps:

1. Expand the **Servers** folder from the navigation tree
2. Click **Application Servers** from the **Servers** folder
3. Click on the name of your application server (for example server1) from the list of application servers in the workspace.
4. Click the **Performance Monitoring Service** entry in the additional properties pane of the workspace.

The Performance Monitoring Service properties configuration pane opens in the workspace. To enable the PMI Service of this application server, mark the checkbox labelled **Startup**. In order to configure the instrumentation level used at application server startup time the initial specification level can be altered from this pane. The configuration needs to be saved and the application server restarted for these changes to take effect.

After saving and restarting the application server, the Performance Monitoring Service properties pane shows two pages, the Configuration page just used to set the initial settings and a Runtime page used to alter the instrumentation level at runtime without having to restart the application server.

The Performance Monitoring Service properties pane is shown in Figure 21-6 .


Application Servers > server1 >

Performance Monitoring Service

Configuration and Runtime Settings for Performance Monitoring Infrastructure (PMI)

Runtime | Configuration

General Properties

Startup	✓	Specifies whether the server will attempt to start specified service when the server starts.

Initial specification level	None - All modules below set to "N" (None). [✓ Custom - Modify, add or remove the modules from the below list.] Standard - All modules below set to "H" (High)
	beanModule=N
	cacheModule=N
		A Performance Monitori Infrastructure (PMI) specifi string that stores PMI specification levels for all components in the server.I N,L,M,H,X represent None,Low,Medium,High,Ma respectively.

Figure 21-6 Performance Monitoring Service properties pane

See “Performance Monitoring Service configuration” on page 858 for an introduction to setting the instrumentation level.

Using Tivoli Performance Viewer to set instrumentation level

To open the Performance Monitoring Settings window, choose the **Data Collection** icon on the Resource Selection panel of the Performance Viewer. This selection provides two options on the Counter Selection panel. Choose the **Current Activity** option to view and change monitoring settings. Alternatively, use **File** -> **Current Activity** to view the monitoring settings. Selecting **Custom** and clicking **Specify...** for any application server opens the Performance Monitoring Settings window.

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 for the application server associated with that resource.

Using the wsadmin command to configure the PMI service

In order to configure the PMI Service of a specific application server, a reference to the PMI Service configuration object of that application server is needed. All PMI Service configuration objects can be listed using the following **`wsadmin`** command:

```
wsadmin> $AdminConfig list PMIService
```

To enable performance data monitoring, use the following **`wsadmin`** command with your specific PMI Service configuration id:

```
wsadmin> $AdminConfig modify \
   (cells/ka0kkwdNetwork/nodes/ka0kkwd/servers/server1: \
   server.xml#PMIService_1) {{enable true}}
```

The configuration needs to be saved before restarting the application server, Use this **`wsadmin`** command to save the configuration:

```
wsadmin> $AdminConfig save
```

To restart the application server use these **`wsadmin`** commands (in a single server environment, don't specify the node in the startServer command):

```
wsadmin> $AdminControl stopServer server1
wsadmin> $AdminControl startServer {server1} {ka0kkwd}
```

To disable performance data collection, use the following **`wsadmin`** command (save the configuration and restart the application server for the change to take effect):

```
wsadmin> $AdminConfig modify \
   (cells/ka0kkwdNetwork/nodes/ka0kkwd/servers/server1: \
   server.xml#PMIService_1) {{enable false}}
```

To set the overall instrumentation level to low for the server1 application server use the following **wsadmin** commands:

```
wsadmin> set perfObjRef [$AdminControl makeObjectName \
   [$AdminControl completeObjectName type=Perf,process=server1,*]]
wsadmin> set params [java::new {java.lang.Object[]} 2]
wsadmin> $params set 0 [java::new java.lang.String pmi=l]
wsadmin> $params set 1 [java::new java.lang.Boolean true]
wsadmin> set sigs  [java::new {java.lang.String[]} 2]
wsadmin> $sigs set 0 java.lang.String
wsadmin> $sigs set 1 java.lang.Boolean
wsadmin> $AdminControl invoke_jmx $perfObjRef setInstrumentationLevel \
   $params $sigs
```

The **setInstrumentationLevel** method call doesn't produce any output. You can use the Tivoli Performance Viewer as a quick way to check that the commands actually had an effect on the instrumentation level setting.

The string passed in as the first parameter to the **setInstrumentationLevel** method (**pmi=l**) specifies the instrumentation level setting and can hold the overall instrumentation level or a compound of instrumentation levels for each module and/or submodule like this:

```
java::new java.lang.String \
   beanModule=H:connectionPoolModule=H:jvmRuntimeModule=L:orbPerfModule=H
```

For a description of valid modules and values, see the next section Performance Monitoring Service configuration.

Author Comment: This part needs to be described, but depending on the final administrative console layout this should go in a different section (in case this changes from beta4 toGA)

Performance Monitoring Service configuration

The Performance Monitoring Service properties pane in the administrative console and the Performance Monitoring Settings window in Tivoli Performance Viewer both offer to set the instrumentation levels to none or standard in the same ways, but using the custom option to set levels, there is a big difference in their use.

Author Comment: This above statement is hopefully only for beta 4. The help for the Performance Monitoring Service properties pane describes setting levels as easy as ‘selecting the PMI module and monitoring level, click Add’. As a result the pane layout could very well change for GA.

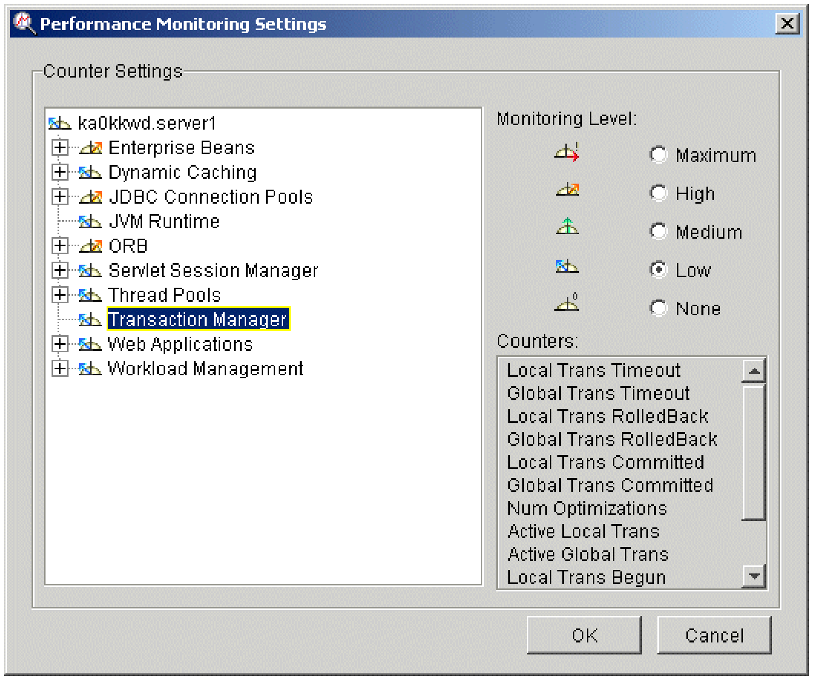
Where as Tivoli Performance Viewers Performance Monitoring Settings window offers to view the module tree structure graphically and even lists specific applications, methods and counters, the properties pane in the administrative console has a text pad for adding or changing modules and levels by hand. Settings are syntax verified in the administrative console, so that the configuration file will be syntactically right, but no semantics check is performed.

To set the instrumentation level for a resource category in the Tivoli Performance Monitoring Settings window:

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 21-7, for example, shows Transaction Manager set to Low.
3. Click **OK** to close the window
4. Click **Apply** in the data monitoring pane from the main window to apply the settings.

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 right (in the Counters box) when you choose the monitoring level. Figure 21-7 shows the counters available for Transaction Manager with the instrumentation level set to **Low**.



ka0kkwd.server1
- Enterprise Beans
- Dynamic Caching
- JDBC Connection Pools
- JVM Runtime
- ORB
- Servlet Session Manager
- Thread Pools
- Transaction Manager
- Web Applications
- Workload Management

Monitoring Level:

( ) Maximum
( ) High
( ) Medium
[✓ Low]
( ) None

Counters:

Local Trans Timeout
Global Trans Timeout
Local Trans RolledBack
Global Trans RolledBack
Local Trans Committed
Global Trans Committed
Num Optimizations
Active Local Trans
Active Global Trans
Local Trans Begun

Figure 21-7 Performance Monitoring Settings for JVM Runtime

Setting a particular instrumentation level for one resource category applies the same instrumentation level to all of its 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 node agent. You have to disable them explicitly.

Setting instrumentation level for enterprise bean methods

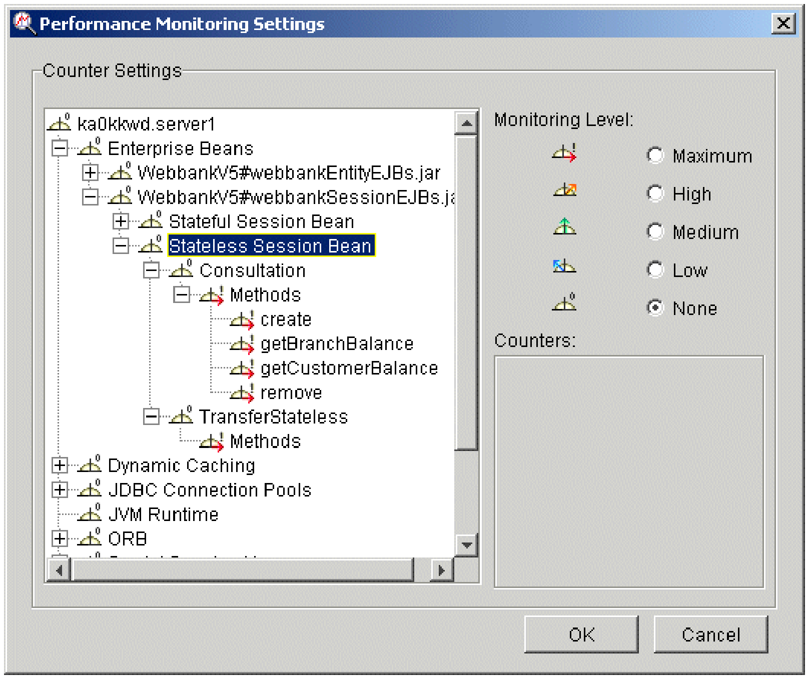
To avoid the overhead of monitoring individual remote methods, individual methods in enterprise beans will not be displayed in the performance pane unless the methods monitoring level is set to maximum. To display individual methods and specify their instrumentation levels, do the following:

1. Open the Performance Monitoring Settings window.
2. Set the instrumentation level for the methods category to Maximum.

3. Click **OK** to close the Performance Monitoring Settings window.
4. Click **Apply** to make the change take effect. Then re-open the Performance Monitoring Settings window. Individual methods are now displayed as shown in Figure 21-8.
5. 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 21-8, for an example, the Consultation enterprise bean has been called at least once, but the TransferStateless bean has not been called yet.



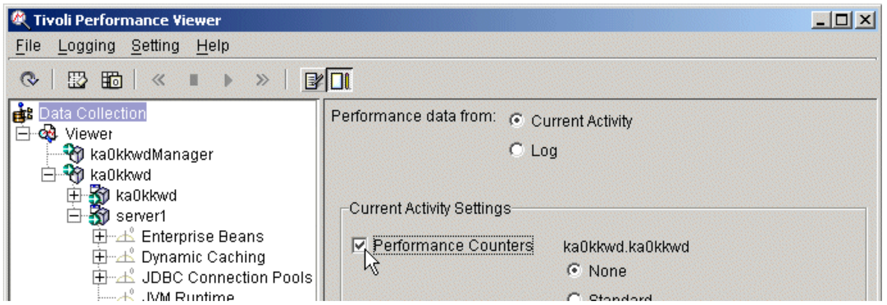
```
ka0kkwd.server1
  Enterprise Beans
    WebbankV5#webbankEntityEJBs.jar
    WebbankV5#webbankSessionEJBs.ja
      Stateful Session Bean
      Stateless Session Bean
        Consultation
          Methods
            create
            getBranchBalance
            getCustomerBalance
            remove
        TransferStateless
          Methods
  Dynamic Caching
  JDBC Connection Pools
  JVM Runtime
  ORB
```

Monitoring Level: Maximum [ ] High [ ] Medium [ ] Low [ ] [✓ None]

Figure 21-8 Enterprise bean methods available after the first method call

Starting and stopping data reporting

Starting and stopping performance data collecting can be useful for monitoring select periods of time. To enable or disable performance data collecting, toggle the checkbox **Performance Counters** in the data monitoring pane and click **Apply**, as illustrated in Figure 21-9



Performance data from: (● Current Activity) (Log)

Current Activity Settings

[✓] Performance Counters ka0kkwd.ka0kkwd

(● None) (Standard)

Figure 21-9 Controlling performance data collecting

When new modules or instances (JSPs, servlets, enterprise beans) are loaded by WebSphere Application Server while Tivoli Performance Viewer 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 **File** -> **Refresh** from the main menu, or click the **Refresh** icon in the toolbar.
3. Set instrumentation level for this new module or instance as described in “Enabling PMI Service and setting the instrumentation level” on page 855.

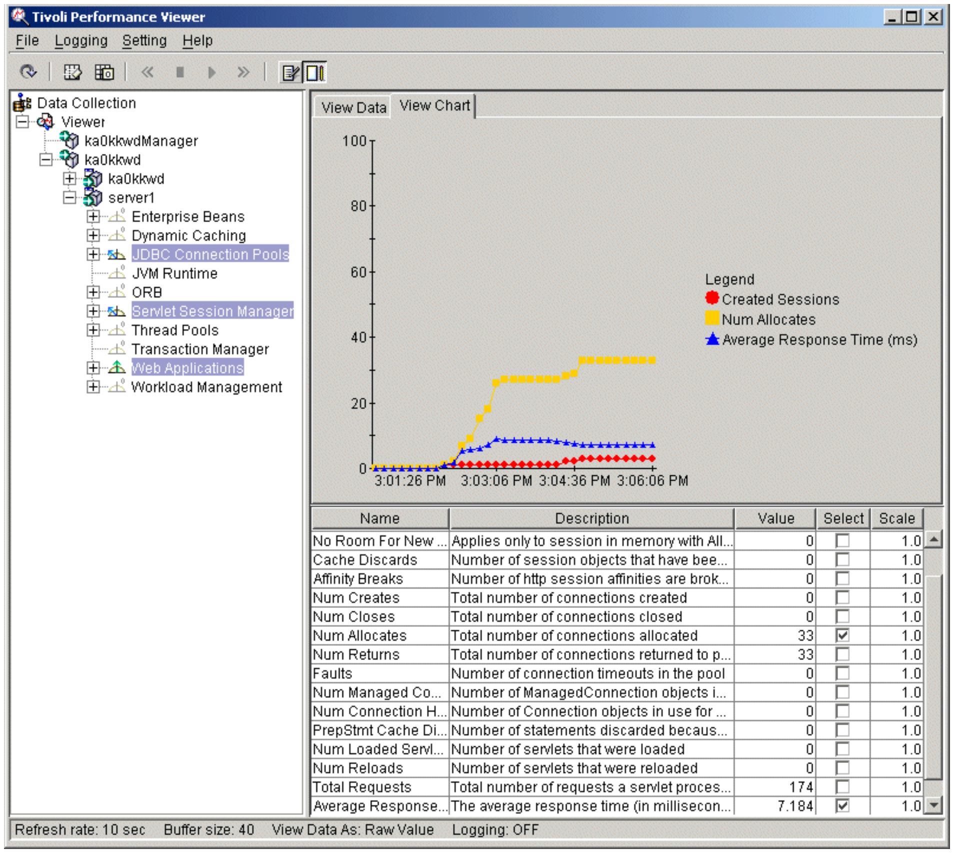
Tip: For load test scenarios it is wise to set up so-called preload tests. Preload tests make sure that all of your resources are loaded at least once. All caches inside WebSphere Application Server will be initialized. Another advantage is that you can configure Tivoli Performance Viewer in one step because all instances are available.

Showing correlated counters

To analyze the performance data captured by Tivoli Performance Viewer 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 JDBC Connection Pools), and *Average Response Time* (from Web Applications), then set up the Tivoli Performance Viewer as shown in Figure 21-10.



Name	Description	Value	Select	Scale
No Room For New ...	Applies only to session in memory with All...	0	☐	1.0
Cache Discards	Number of session objects that have bee...	0	☐	1.0
Affinity Breaks	Number of http session affinities are brok...	0	☐	1.0
Num Creates	Total number of connections created	0	☐	1.0
Num Closes	Total number of connections closed	0	☐	1.0
Num Allocates	Total number of connections allocated	33	☑	1.0
Num Returns	Total number of connections returned to p...	33	☐	1.0
Faults	Number of connection timeouts in the pool	0	☐	1.0
Num Managed Co...	Number of ManagedConnection objects i...	0	☐	1.0
Num Connection H...	Number of Connection objects in use for ...	0	☐	1.0
PrepStmt Cache Di...	Number of statements discarded becaus...	0	☐	1.0
Num Loaded Servl...	Number of servlets that were loaded	0	☐	1.0
Num Reloads	Number of servlets that were reloaded	0	☐	1.0
Total Requests	Total number of requests a servlet proces...	174	☐	1.0
Average Response...	The average response time (in millisecon...	7.184	☑	1.0

Refresh rate: 10 sec Buffer size: 40 View Data As: Raw Value Logging: OFF

Figure 21-10 Selecting multiple counters from different resource categories

Record a load scenario

All data being reported by the Tivoli Performance Viewer 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 Tivoli Performance Viewer main menu or click the **Logging on/off** icon in the toolbar.
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 *.perf (Tivoli Performance Viewer 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 or click the **Logging on/off** icon in the toolbar.

Replaying a load scenario

Both log file types can be replayed using the Tivoli Performance Viewer. Log files saved in XML format (*.xml) provide the option of analysis using third-party tools, but can get rather large. If size is of concern it is recommended to use the binary log format. To replay a log file, do the following:

1. Select **File** -> **Log** from the Tivoli Performance Viewer main menu.
2. Click the **Browse...** button to open the file browser.
3. In the Open window, locate the name of the file to replay and click the **Open** button.
4. Click the **Play** icon from the toolbar or select **Setting -> Log Replay -> Play** from the main menu

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 the **FF** button (Fast Forward) in the toolbar or by select **Setting -> Log Replay -> FF** from the main menu.

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, click the **Rewind** button in the toolbar or select **Setting -> Log Replay -> Rewind** from the main menu.

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 **Setting** -> **Clear Buffer** from the main menu. Alternatively, click the **Clear Buffer** icon in the toolbar.

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 **Setting** -> **Reset to Zero** from the main menu. Alternatively, click the **Reset to Zero** icon in the toolbar.

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 Tivoli Performance Viewer display settings:

- Specifying a different refresh rate

- Changing the data view
- Changing the buffer size

Specifying a different refresh rate

By default, the Tivoli Performance Viewer 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 **Setting** -> **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 **Setting** -> **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 Tivoli Performance Viewer 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 buffer 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 **Setting** -> **Set Buffer Size...** from the main menu.
2. In the Set Buffer Size window, specify the number of rows to display.
3. Click **OK**.

21.3.6 Getting online help

WebSphere Tivoli Performance Viewer has HTML-based online help. To see the help pages select **Help** -> **Help Topics** from the main menu. There are also a number of Tivoli Performance Viewer articles in the WebSphere InfoCenter, available at:

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

21.4 Using performance monitoring servlet

The performance servlet provides a way to use an HTTP request to query the performance data for an entire WebSphere Application Server administrative cell.

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 Tivoli Performance Viewer as well. Therefore, it is subject to the same restrictions on the availability of data as the Tivoli Performance Viewer.

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 cell, provided that the server you installed on can connect to all application servers in the cell.

21.4.1 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 21-1 shows the number of loaded servlets. Note that the path providing the context of the observation is not shown.

Example 21-1 Number of loaded servlets

```
<numLoadedServlets>
   <PerfNumericInfo time="1035469483328" uid="pmi1" val="5.0" />
</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 21-2 shows the response time of an object. Note that the path providing the context of the observation is not shown.

Example 21-2 Response time of an object

```
<responseTime>
```

```
   <PerfStatInfo mean="4.878878648233487" num="13020"
      sum_of_squares="4.2228191E7" time="1035470723062" total="63523.0"
      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 21-3 shows the number of concurrent requests. Note that the path providing the context of the observation is not shown.

Example 21-3 Number of concurrent requests

```
<poolSize>
   <PerfLoadInfo currentValue="10.0" integral="1.7639095E7"
      mean="9.787263143293107" time="1035470722859"
      timeSinceCreate="1802250.0" uid="pmi4" />
</poolSize>
```

21.4.2 Monitoring an application with the performance servlet

The first request initializes the performance servlet. During initialization the servlet connects to the deployment manger to retrieved a list of nodes, servers and MBeans located within the cell in which it is deployed. Subsequently all application servers in the cell are contacted in order to gather performance data. In a single server environment only the default application server is contacted during initialization. Because the collection of this data is expensive, the performance servlet holds this as a cached list for performance reasons.

The performance servlet tries to contact the deployment manager or application server on port 8879 using the SOAP protocol. If that fails port 8880 is tried instead. As the connection setup is only performed on the very first run of the performance servlet, it is very important that the deployment manager, node agent or application server is running and listening on either of these ports or that the performance servlet is configured to use appropriate connection settings. The performance servlet takes the parameters shown in Table 21-3 upon initialization. Note that either all or no parameters needs to be specified for the settings to take effect, regardless of their initial default value.

Table 21-3 Performance servlet initialization parameters

Parameter	**Description**
host=host_name	Specifies the hostname or IP address of the node running the deployment manager, node agent or application server to connect to (default: localhost).
port=connector_port	Specifies which TCP port number to use for the connection (default: 8879 and 8880).
connector=connector_protocol	Specifies which protocol to use for the connection, currently only SOAP or RMI are supported (default: SOAP).
refreshConfig=boolean	Forces a module hierarchy refresh. This parameter needs to be set (true) at initialization (default: false)

To show the performance data as XML do the following:

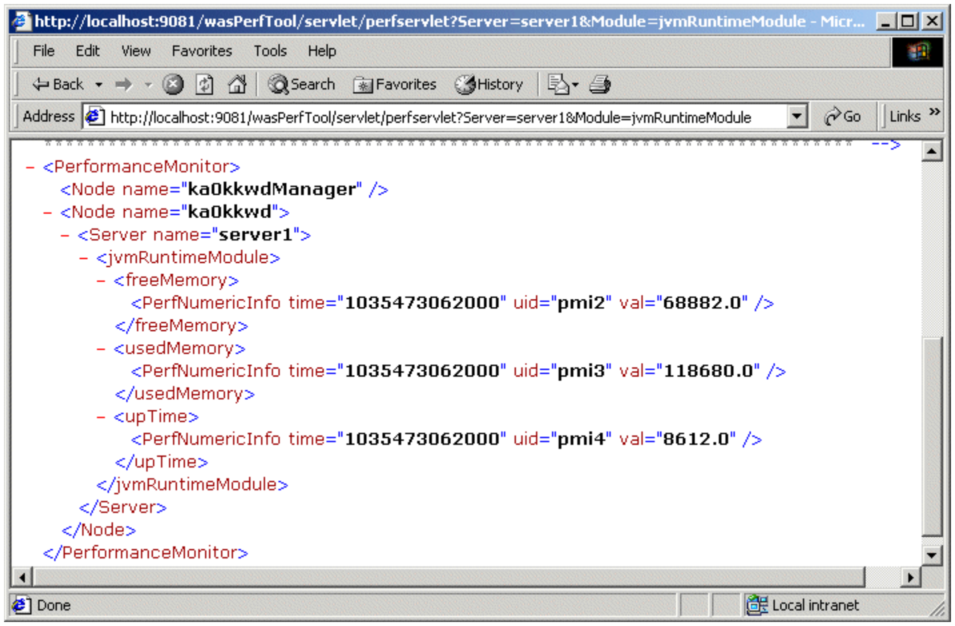
1. Make sure the deployment manager, node agent or default application server (server1) is running.
2. Make sure performance monitoring is enabled and instrumentation levels are set.
3. Open your browser using the following URL to invoke the servlet. The URL depends on the environment configuration:

   ```
   http://localhost/wasPerfTool/servlet/perfservlet
   ```

If you are in a non-default environment, use the following syntax for the URL in step 3 above:

```
http://localhost/wasPerfTool/servlet/perfservlet?host=myhost&port=8885&
   connector=SOAP&refreshConfig=true
```

Figure 21-11 shows the XML page generated by the performance monitoring servlet in Microsoft Internet Explorer 5.5, for server1 and JVM runtime instrumentation level set to Low.


http://localhost:9081/wasPerfTool/servlet/perfservlet?Server=server1&Module=jvmRuntimeModule - Micr...
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9081/wasPerfTool/servlet/perfservlet?Server=server1&Module=jvmRuntimeModule
Go Links
Done
Local intranet

```
- <PerformanceMonitor>
    <Node name="ka0kkwdManager" />
  - <Node name="ka0kkwd">
    - <Server name="server1">
      - <jvmRuntimeModule>
        - <freeMemory>
            <PerfNumericInfo time="1035473062000" uid="pmi2" val="68882.0" />
          </freeMemory>
        - <usedMemory>
            <PerfNumericInfo time="1035473062000" uid="pmi3" val="118680.0" />
          </usedMemory>
        - <upTime>
            <PerfNumericInfo time="1035473062000" uid="pmi4" val="8612.0" />
          </upTime>
        </jvmRuntimeModule>
      </Server>
    </Node>
  </PerformanceMonitor>
```

Figure 21-11 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 21-4.

Table 21-4 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
- cacheModule
- connectionPoolModule
- jvmRuntimeModule
- orbPerfModule
- servletSessionsModule
- systemModule
- threadPoolModule
- transactionModule
- webAppModule
- wlmModule

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=yourNodeName
http://localhost/wasPerfTool/servlet/perfservlet?Server=Test%20Server
http://localhost/wasPerfTool/servlet/perfservlet?Module=beanModule+
   jvmRuntimeModule
```

(%20 represents a space in a URL)

In our example in Figure 21-11 on page 871, we used the following URL to request performance data for only the jvmRuntimeModule module:

```
http://hostname/wasPerfTool/servlet/perfservlet?Server=server1&
   Module=jvmRuntimeModule
```

21.5 Using JVMPI facility

Author Comment: can you see these stats in the TPV or PMS?

The Java Virtual Machine Profiler Interface (JVMPI) is a facility of the JVM used 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 or node agent.

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.

All JVMPI performance data is collected by the JVM module, but JVMPI needs to be enabled for the module to update its counters.

The JVMPI facility is available on the Windows, AIX, and Solaris platforms.

21.5.1 Performance data provided by JVMPI

The JVM Profiler Interface provides internal runtime performance data about the following resources:

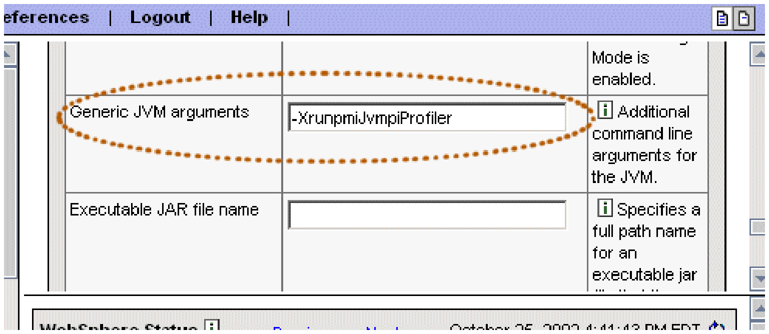
- Garbage collector
  - Number of garbage collection calls
  - Average time in milliseconds between garbage collection calls
  - Average duration in milliseconds of a garbage collection call
- Monitor
  - Number of times that a thread waits for a lock
  - Average time that a thread waits for a lock
- 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

21.5.2 Enabling JVMPI from the administrative console

To enable JVMPI reporting for each individual application server or node agent, do the following on the WebSphere Administrative Console:

1. Click **Servers** -> **Application Servers** or **System Administration** -> **Node Agents** in the console navigation tree (depending on the JVM you would like to profile).

2. Click the application server or node agent from the list of application servers or node agents in the workspace for which JVMPI needs to be enabled.
3. Click the **Process Definition** entry in the additional properties pane of the workspace.
4. Click the **Java Virtual Machine** entry in the additional properties pane of the workspace.
5. Type -`XrunpmiJvmpiProfiler` in the **Generic JVM arguments** field (add it before or after any existing arguments in case you have other arguments already).



		Mode is enabled.
Generic JVM arguments	-XrunpmiJvmpiProfiler	Additional command line arguments for the JVM.
Executable JAR file name		Specifies a full path name for an executable jar

Figure 21-12 Enabling JVMPI

6. Click **Apply** or **OK** to apply the changes.
7. Click **Save** to store the changes in the WebSphere configuration.
8. Start the application server or node agent, or restart the application server or node agent if it is currently running.

Make sure your have set the instrumentation level of the JVM module to MAX for the profiler counters to be updated. Also, refresh the Tivoli Performance Viewer if you are using it.

Important: node agents and application servers collects data in the JVMPI counters of the JVM module regardless of having the command line argument specified. Be cautioned, that collected data is not reliable until the -`XrunpmiJvmpiProfiler` argument is specified.

21.5.3 Enabling JVMPI with the command line interface

To enable JVMPI profiling using the `wsadmin` command interface, perform these steps:

1. Start `wsadmin`.

2. Enter the following command at the prompt:

   ```
   wsadmin> $AdminConfig modify \
      (cells/ka0kkwdNetwork/nodes/ka0kkwd/servers/server1:server.xml#\
      JavaVirtualMachine_1) {{genericJvmArguments -XrunpmiJvmpiProfiler}}
   ```

3. Save the configuration and start the application server or node agent, or restart the application server or node agent if it was already running for the change to take effect.

21.5.4 Disabling JVMPI profiling

From the administrative console

Follow the steps described in 21.5.2, “Enabling JVMPI from the administrative console” on page 873 and remove the JVM command line argument **`-XrunpmiJvmpiProfiler`**.

From the command line interface

Use the **`wsadmin`** command shown in 21.5.3, “Enabling JVMPI with the command line interface” on page 874, but change the **`genericJvmArguments`** to read:

```
{{genericJvmArguments {}}}
```

21.6 PMI request metrics

PMI request metrics is a facility introduced in WebSphere 5 for providing response time information for the major services used by a request as it moves through application servers. The request metrics gives information on the incoming request, such as its type, URI, invoked bean method, client origin etc. and provides the response time to the request as it is processed by major WebSphere services. Metrics are produced at the Web and EJB container level and are measured from the request enters the container to a response is provided.

Author Comment: something seems to be missing in the statement above ..

The request metrics component is designed to be very lightweight in processing and to have the smallest resource footprint possible. This is very important as the facility is globally enabled and extends to every Web and EJB container in the cell. To minimize process and resource usage, a filter can be applied to the metrics component in order to allow only a subset of incoming requests to be monitored. This filter mechanism does not only reduce process usage but also minimizes the amount of metrics data produced.

The request metrics record can be written to a text file (System.out) or sent to an Application Response Measurement (ARM 2.0) compliant agent - or both. Every Web and EJB container stores the timestamp of all requests entering that specific container and writes a record with response time and other metrics to the System.out JVM log or to an available ARM agent as soon as a response is provided. As requests in a WebSphere environment often fan out to different processes on several physical nodes, request metrics can be scattered across different log files. The metrics can be correlated together to resemble a detailed sequence diagram of request response times.

Request metrics are useful for getting information on overall system performance and hence give WebSphere administrators and developers a feel for the experience a user might have with the system at a given time. They can also be used to give an indication of what component is causing problems to the overall system performance and needs to be examined more closely. Problems being application flaws, bottlenecks, leaks, communications etc. The indication of problems with a specific component can also be helpful in tuning efforts as to direct the attention to the correct node and process.

As the request metrics are part of the Performance Monitoring Infrastructure (PMI) framework they are named PMI request metrics. In this section the terms “request metrics” and ”PMI request metrics” will be used interchangeably, both referring to the request metrics of the PMI framework.

21.6.1 Enabling and configuring PMI request metrics

As mentioned, the request metrics are enabled and configured globally for all application servers in the entire WebSphere cell in the deployment manager environment. For a single server environment, the metrics are enabled and configured for all application servers residing on the local node.

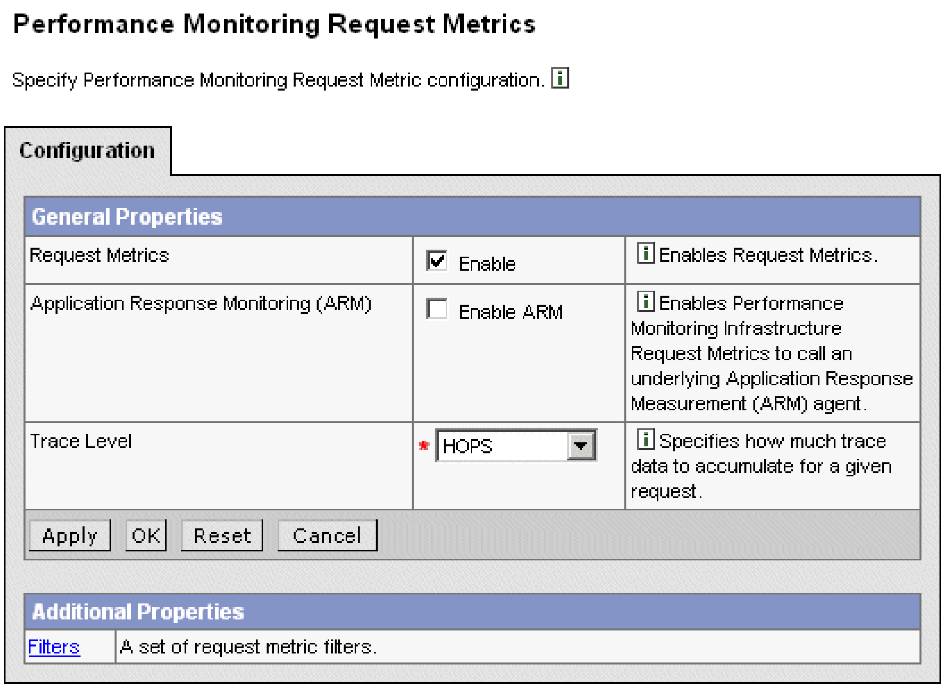
The PMI request metrics component can be enabled, disabled and configured at runtime, without having to restart each individual application server in the cell.

Go to the PMI request metrics configuration pane, by clicking **Troubleshooting** -> **PMI Request Metrics** from the console navigation tree of the administrative console. From this pane there are three general properties as shown in Example 21-13, namely:

- **Request Metrics**. Used for enabling and disabling the request metrics extension to the Web and EJB containers.

- **Application Response Measurement (ARM)**. Activates sending request metrics to an ARM 2.0 compliant agent. For more information on the ARM facility and to see a usage example, see the WebSphere 5.0 InfoCenter.
- **Trace Level**. Specifies level of detail on the request metrics data. A setting of NONE produces no metrics data. HOPS shows the EJB method invoked, requested URI or JDBC statement executed along with the response time. Settings of PERF_DEBUG and DEBUG shows additional detail, but are more performance intensive.

Author Comment: As of this writing only Trace levels of NONE and HOPS are implemented - no additional data is produced for PERF_DEBUG or DEBUG. Statement above needs some work ..


Performance Monitoring Request Metrics

Specify Performance Monitoring Request Metric configuration.

Configuration

General Properties

Request Metrics	✓ Enable	Enables Request Metrics.
Application Response Monitoring (ARM)	☐ Enable ARM	Enables Performance Monitoring Infrastructure Request Metrics to call an underlying Application Response Measurement (ARM) agent.
Trace Level	* HOPS	Specifies how much trace data to accumulate for a given request.

Additional Properties

Filters	A set of request metric filters.

Figure 21-13 PMI request metrics configuration pane

As soon as request metrics are enabled and a trace level greater than NONE is specified and saved to the WebSphere configuration, trace records are written to the System.out JVM log for all application servers for any incoming Web or EJB request.

See the next section for a detailed description of the trace record format.

Important: Only remote EJB calls entering the RMI activity service are monitored by the PMI request metrics component. Hence a servlet calling a local EJB does not produce any metrics data for the bean.

21.6.2 Request metrics trace record format

The request metric trace record written to the System.out JVM log, has the following format:

```
PMRM0003I:
   parent:ver=n,ip=n.n.n.n,time=nnnnnnnnnn,pid=nnnn,reqid=nnnnnn,event=nnnn
   - current:ver=n,ip=n.n.n.n,time=nnnnnnnnnn,pid=nnnn,reqid=nnnnnn,event=nnnn
   type=TTT detail=some_detail_information elapsed=nnnn
```

The fields of the trace record are described in Table 21-5, note that **`PMRM0003I`** is the message identifier for trace records as found in the System.out log.
Table 21-6 lists the entries of the correlator fields (identical for both parent and current correlator).

Table 21-5 PMI request metric trace record format

Field	Description
parent	Identifier to uniquely represent the upstream request (parent correlator)
current	Identifier to uniquely represent the current request (current correlator)
type	Type of request (valid types: URI, EJB or JDBC)
detail	Detailed request information showing the full URI of the request, the fully qualified EJB method invoked or the JDBC SQL statement executed.
elapsed	The total elapsed time for the operation in milliseconds. The time includes time for all child operations performed (metrics records having their parent correlator field equal to this records current correlator).

The type and detail fields are described as follows:

- **Universal Resource Identifier (URI):** The trace record was generated by a Web component. The URI is the name of the URI used to invoke the request.
- **Enterprise bean (EJB):** The fully qualified package and method name of the enterprise bean.
- **Java Database Connectivity (JDBC):** The values select, update, insert or delete for prepared statements. For non-prepared statements, the full statement can appear.

Table 21-6 PMI request metric correlator fields

Field	Description
ver	The version of the correlator. For convenience, it is duplicated in both the parent and current correlators.
ip	The IP address of the node of the application server that generated the correlator.
pid	The process ID of the application server that generated the correlator.
time	The start time of the application server process that generated the correlator.
reqid	An ID assigned to the request by Request Metrics, unique to the application server process.
event	An event ID assigned to differentiate the actual trace events.

The trace record format is composed of two correlators: a parent correlator and current correlator. The parent correlator represents the upstream request and the current correlator represents the current operation. If the parent and current correlators are the same, then the record represents an operation that occurred as it entered WebSphere Application Server.

To correlate trace records for a particular request, collect records with a message ID of **PMRM0003I** from the appropriate server logs. Records are correlated by matching current correlators to parent correlators. The logical tree can be created by connecting the current correlators of parent trace records to the parent correlators of child records. This tree shows the progression of the request across the server cluster.

Examples of URI and EJB request metrics records are shown in Example 21-4 and Example 21-5.

Example 21-4 URI request metrics

```
[10/29/02 9:46:31:578 EST] 7bca033b PmiRmArmWrapp I PMRM0003I:
parent:ver=1,ip=9.24.104.131,time=1035902046562,pid=3492,reqid=8193,event=0
- current:ver=1,ip=9.24.104.131,time=1035902046562,pid=3492,reqid=8193,event=1
type=URI detail=/webbank/Navigation.html elapsed=16
```

Example 21-5 EJB request metrics

```
[10/29/02 9:00:16:812 EST] 6d23ff00 PmiRmArmWrapp I PMRM0003I:
parent:ver=1,ip=9.24.104.131,time=1035899941844,pid=3524,reqid=4096,event=0
- current:ver=1,ip=9.24.104.131,time=1035899941844,pid=3524,reqid=4096,event=1
type=EJB detail=com.ibm.ws.wlm.server.WLMTemplateImpl.push elapsed=15
```

21.6.3 Filters

Filters are important for producing only the output necessary for a given monitoring task. A named URI, EJB or client IP address can be specified and only the needed metrics data is produced. The performance load added to the containers by the request metrics component is reduced by adding more restrictive filters.

For HTTP requests arriving at a Web container it is possible to filter on the URI and client IP address. For incoming requests to the EJB container, it is possible to filter on the bean method name and the client ip address. All the filters are listed and described here:

- **Client IP address filters**. Requests are filtered based on a known IP address. You can specify a mask for an IP address using the asterisk (*). If used, the asterisk must always be the last character of the mask, for example 127.0.0.*, 127.0.*, 127*.
- **URI filters**. Requests are filtered, based on the URI of the incoming HTTP request. The rules for pattern matching are the same as for matching client IP address filters.
- **Enterprise bean method name filters**. Requests are filtered based on the full name of the enterprise bean method. As with IP address and URI filters, you can use the asterisk (*) to provide a mask. The asterisk must always be the last character of a filter pattern.
- **Filter combinations**. If both URI or EJB and Client IP address filters are active, then request metrics requires a match for both filter types. If neither is active, all requests are considered a match.

It is important to understand that filters are applied to requests as they enter the WebSphere application server environment from the client (at the Web or EJB container level). Depending on defined filters the request is either marked to having metrics generated or not marked to mean that no request metrics records should be produced for the duration of the request.

As an example think of two WebSphere nodes, one having a Web container with servlets and the other having a EJB container with beans. A remote client sends a HTTP request for a servlet to the Web container. If request metrics are enabled and a filter with the client IP address is defined request metrics will be generated for the servlet request. If the servlet were to invoke a bean method on the remote node, a request metrics record on the remote node would also be generated as the initial request was originating from a client for which a client IP address filter exists.

21.7 Monitoring the IBM HTTP Server

Author Comment: The IHS 1.3.26 hooks to the Windows performance monitor is currently broken, but should be fixed for GA.

The Windows version of IBM HTTP Server includes Windows performance monitor hooks. This allows you to use the Windows NT performance monitor or the Windows 2000 system monitor to observe the current state of an active IBM HTTP Server, along with all kinds of other system resources.

21.7.1 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 or uncomment the lines shown in Example 21-6 from the **`httpd.conf`** file and restart the IBM HTTP Server.

Example 21-6 Enable extended performance data in httpd.conf

```
LoadModule status_module modules/ApacheModuleStatus.dll
ExtendedStatus On
```

For more information about modifying the HTTP server configuration please refer to the IBM HTTP Server InfoCenter:

`http://www.ibm.com/software/webservers/httpservers/library.html`

21.7.2 Starting the Windows 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:

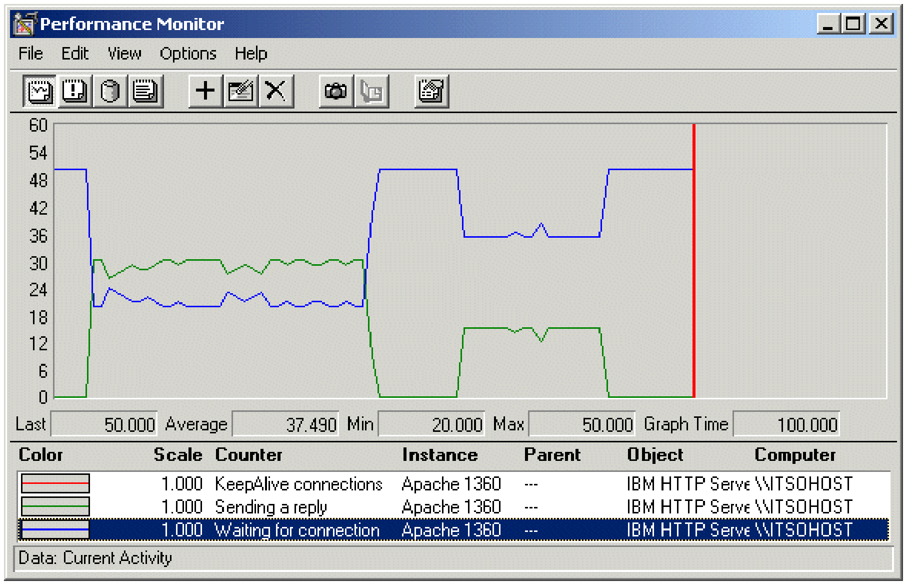
1. Click **Start** -> **Settings** -> **Control Panel** to open the Control Panel window.
2. Double-click **Administrative Tools** to open the Administrative Tools window.
3. Double-click **Performance** to open the Performance window.
4. Select the **Tree** tab on the top-left and click **System Monitor** to activate it.

21.7.3 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 21-14 shows an example of a monitoring session.



Last 50.000 Average 37.490 Min 20.000 Max 50.000 Graph Time 100.000

Color	Scale	Counter	Instance	Parent	Object	Computer
🟥	1.000	KeepAlive connections	Apache 1360	---	IBM HTTP Serve	\\ITSOHOST
🟩	1.000	Sending a reply	Apache 1360	---	IBM HTTP Serve	\\ITSOHOST
🟦	1.000	Waiting for connection	Apache 1360	---	IBM HTTP Serve	\\ITSOHOST

Data: Current Activity

Figure 21-14 Windows NT Performance Monitor

21.7.4 IBM HTTP Server status page

The IBM HTTP Server server-status page is available on all supported IBM HTTP Server platforms. It shows performance data on a Web page in HTML format.

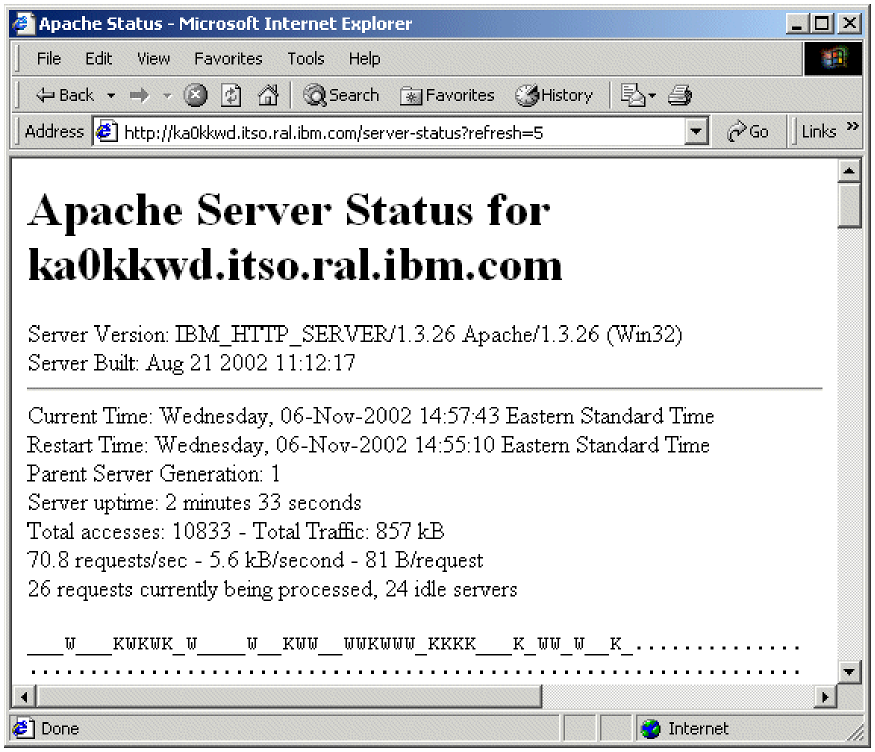
Perform the following steps to activate the server-status page:

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 21-15, you can see (along with additional information) the number of requests currently being processed, and the number of idle servers.


Apache Status - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://ka0kkwd.itso.ral.ibm.com/server-status?refresh=5
Go Links
Done
Internet

Apache Server Status for ka0kkwd.itso.ral.ibm.com

Server Version: IBM_HTTP_SERVER/1.3.26 Apache/1.3.26 (Win32)
Server Built: Aug 21 2002 11:12:17

Current Time: Wednesday, 06-Nov-2002 14:57:43 Eastern Standard Time
Restart Time: Wednesday, 06-Nov-2002 14:55:10 Eastern Standard Time
Parent Server Generation: 1
Server uptime: 2 minutes 33 seconds
Total accesses: 10833 - Total Traffic: 857 kB
70.8 requests/sec - 5.6 kB/second - 81 B/request
26 requests currently being processed, 24 idle servers

```
___W___KWKWK_W____W__KWW__WWKWWW_KKKK___K_WW_W__K_..............
................................................................
```

Figure 21-15 IBM HTTP Server status page

21.8 Tuning your environment

As there are so many components in WebSphere Application Server that have an impact on performance and since the tuning task is highly dependent on the applications running in the environment, this tuning section is only going to give an overview of the discipline of tuning and give simple guidelines for getting your runtime to perform at its best.

There are a number of things to take into consideration when starting tuning the WebSphere environment. Tuning is about utilizing resources to the best of usage and their fullest potential to have requests processed as fast as possible. If you have a 4THz CPU and only 1% of its processing resource is being utilized, it likely would have been a better investment to have opted for a 400GHz CPU and some additional RAM or a network upgrade. It could also be that you didn’t have the clients to utilize your environment to its fullest potential, or some queue or timeout setting was preventing every other component in the system from performing at their fullest, simply queueing up requests at a point creating a bottleneck.

This section is going to help locate bottlenecks, give a practical introduction to monitoring for tuning purposes and finally give recommendations on settings for major WebSphere environment properties. It covers WebSphere Application Server parameters tuning, application tuning will not be covered. For help on application tuning, see the WebSphere Developer Domain (`http://www7b.boulder.ibm.com/wsdd`) it has a zone for coding and tuning best practices and a zone for development specific environments.

A number of in depth articles and books exist on the tuning subject, they will be referenced at the end of the tuning section and are also listed in the “Related publications” on page 1087.

21.8.1 Queuing

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.

As an example, think of your application as built of servlets, EJBs and a connection manager. Each application part is able to process a certain amount of requests in a given time frame. A client request enters the Web server and travel through WebSphere components in order to provide a response to the client. We can illustrate individual WebSphere components, containing the specific application parts, as interconnected pipes that form a large tube as shown in Figure 21-16

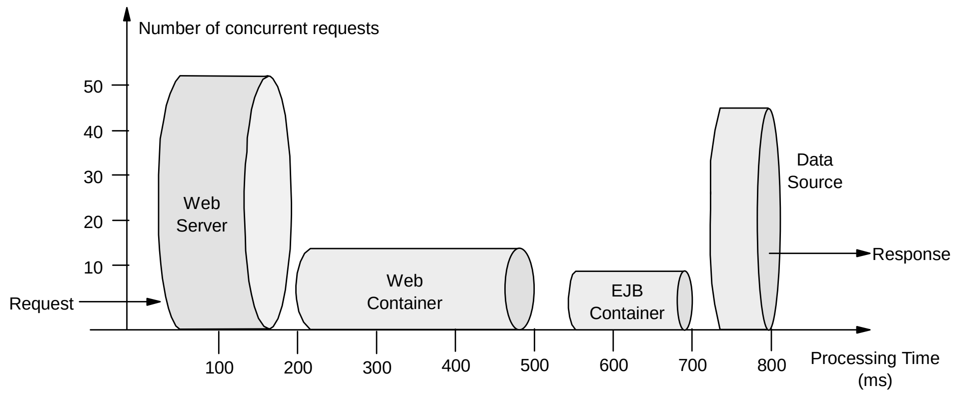


Figure 21-16 Queuing network

The width of the pipes (equals height in the illustration) represent the number of requests a specific application part can process at any given time. The length represents the processing time the component takes to provide a response to the request. In order to find a component that might be a bottleneck to the others (not processing as many request in a given second as the others), it is useful to calculate a operations per second (tps) ratio for each component. Ratio calculations for some fictional application parts are shown in Example 21-7

Example 21-7 Operations per second ratio calculations

The Web Server can process 50 requests in 100 ms = $\frac{50req}{0.1s} = 500tps$

The Web container parts can process 18 requests in 300 ms = $\frac{18req}{0.3s} = 60tps$

The EJB container parts can process 9 requests in 150 ms = $\frac{9req}{0.15s} = 60tps$

The datasource can process 40 requests in 50 ms = $\frac{40req}{0.05s} = 800tps$

The example shows that the application parts resident in the Web container and EJB container processes requests at the same speed. Nothing would be gained from increasing the processing speed of the servlets or other parts of the Web container because the EJB container parts would slow things down to the speed of 60tps. In fact building up a queue of requests in front of the EJB container.

The example also shows the importance of having queues in the system. Looking at the operations ratio of the Web and EJB container parts, the components are able to process the same number of requests over time, but the Web container could produce double the amount of requests that the EJB container parts could start process at any given time. It is crucial that in order to keep the EJB container parts busy and able to keep the processing speed of 60tps the remaining 9 requests be queued as the EJB container would be able to start processing these after 150ms, actually 150ms before the Web container parts could produce new requests. On the other hand it would not be beneficial to have too many requests queue up between the Web server and Web container as the Web server processing takes a short time and is able to produce a larger number of requests at a given time. In the latter case, the requests should be rejected at the Web server and queued up in the network in front of the Web server (in order to save WebSphere resources).

The first rule of tuning is to minimize the number of requests in WebSphere Application Server queues. In general, requests should wait in the network (in front of the Web server), rather than waiting in WebSphere Application Server. This configuration allows only those requests that are ready to be processed to enter the queuing network. It also adds stability, because no component is overloaded. The Edge Server can be used to direct waiting users to other servers in a WebSphere Application Server cluster.

It should be noted that is common for applications to have more requests processed in the Web Server and Web container component parts (static content such as HTML and gifs as well as servlets and jsps) than by EJBs and back-end systems. As a result the queue sizes would be progressively smaller moving deeper into the WebSphere components. This is one of the reasons queue sizes should not solely depend on the operation ratios.

As the above example shows, it is very important to configure queue sizes for the WebSphere Application Server environment to perform at its best, but what is the right queue size and how is it configured?

21.8.2 Determining queue sizes

A simple way to determine the right queue size for any component is to perform a number of load runs against the application server environment while having very large queues (this allows maximum concurrency through the system). For example, start the first experiment with a queue size of 100 at each of the servers in the queuing network: Web server, Web container and data source (see “Configuring the queues” on page 888). A test that represents an average user interaction should be used to load the WebSphere environment continuously from a number of clients in order to measure overall throughput. Run a set of experiments to determine when the system capabilities are fully stressed (the saturation point). Conduct these tests after most of the bottlenecks have been removed from the application. The typical goal of these tests is to drive CPUs to near 100% utilization. It is recommended to use a stress tool for generating the application load. Such a tool could be the IBM Web Performance Tools (WPT), formerly named aktools, freely downloadable from `http://www.alphaworks.ibm.com` or a test suite such as LoadRunner from Mercury Interactive, `http://www.mercuryinteractive.com`

Begin a series of experiments to plot the throughput curve, increasing the concurrent user load after each experiment. For example, perform experiments with 1 user, 2 users, 5, 10, 25, 50, 100, 150 and 200 users. After each run, record the throughput (requests per second) and response times (seconds per request).

The curve resulting from the baseline experiments should resemble the typical throughput curve. An example throughput curve is shown in Figure 21-17

Note that concurrent users is not a well defined term in the literature. Here it is used to mean the number of concurrently active users that sends a request, waits for the response and immediately resends a new request upon response reception - without thinktime.

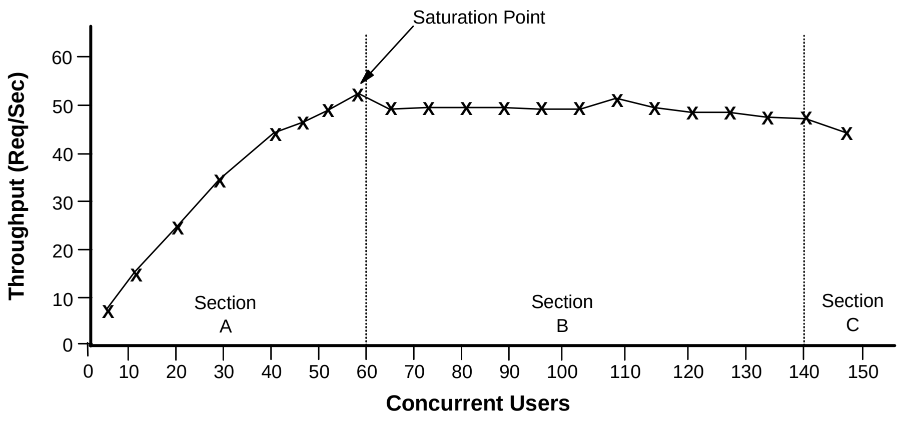


Figure 21-17 Throughput curve

The throughput of WebSphere Application Server is a function of the number of concurrent requests present in the total system. Section A, the light load zone, shows that as the number of concurrent user requests increases, the throughput increases almost linearly with the number of requests. This reflects that, at light loads, concurrent requests face very little congestion within the WebSphere Application Server system queues. At some point, congestion starts to develop and throughput increases at a much lower rate until it reaches a saturation point that represents the maximum throughput value, as determined by some bottleneck in the WebSphere Application Server system. The most manageable type of bottleneck occurs when the CPUs of the WebSphere Application Server machines become fully utilized. This is desirable because a CPU bottleneck can be fixed by adding additional or more powerful CPUs.

In the heavy load zone or Section B, as the concurrent client load increases, throughput remains relatively constant. However, the response time increases proportionally to the user load. That is, if the user load is doubled in the heavy load zone, the response time doubles. At some point, represented by Section C, the buckle zone, one of the system components becomes exhausted. At this point, throughput starts to degrade. For example, the system might enter the buckle zone when the network connections at the Web server exhaust the limits of the network adapter or if the requests exceed operating system limits for file handles.

It is desirable to reach the saturation point by driving CPU utilization close to 100% as this gives an indication that a bottleneck is not caused by e.g.. application memory usage. For example, the application might be creating Java objects causing excessive garbage collection bottlenecks in Java.

The number of concurrent users at the throughput saturation point represents the maximum concurrency of the application. For example, if the application saturated WebSphere Application Server at 50 users, 48 users might give best combination of throughput and response time. This value is called the Max Application Concurrency value. Max Application Concurrency becomes the preferred value for adjusting the WebSphere Application Server system queues. Remember, it is desirable for most users to wait in the network; therefore, queue sizes should increase when moving downstream farther from the client. For example, given a Max Application Concurrency value of 48, start with system queues at the following values: Web server 75, Web container 50, data source 45. Perform a set of additional experiments adjusting these values slightly higher and lower to find the best settings.

21.8.3 Configuring the queues

Within WebSphere Application Server, the queues are represented as pooled resources, for example, thread pool, database connection pool. The pool setting determines the maximum concurrency level of the resource. This section describes how the different queues are represented in WebSphere and their settings.

As the queues are most easily tuned from the inside out of the WebSphere Application Server environment this section describes the queue properties in this order (looking at the components from right to left in Figure 21-16 on page 885).

Data sources

Connection pool size

When accessing any database, the initial database connection is an expensive operation. WebSphere Application Server supports JDBC 2.0 Standard Extension APIs to provide support for connection pooling and connection reuse. The connection pool is used for direct JDBC calls within the application, as well as for enterprise beans using the database.

The connection pool size is set from the administrative console using these steps.

1. Click **Resources** -> **JDBC** in the console navigation tree.
2. Click the name of provider from the list of JDBC providers in the workspace.
3. Click the **Datasources** entry in the additional properties pane of the workspace.
4. Click the name of datasource from the list of Datasources in the workspace.
5. Click the **Connection Pool** entry in the additional properties pane of the workspace.
6. Use the **Min connections** and **Max connections** fields to configure the pool size.
7. Save the configuration and restart the affected application servers for the changes to take effect.

Tivoli Performance Viewer can help find the optimal size for the connection pool. Use a standard workload that represents a typical number of incoming client requests, use a fixed number of iterations, and use a standard set of configuration settings. Watch the **Pool Size**, **Percent Used** and **Concurrent Waiters** counters of the Datasource submodule of the JDBC Connection Pools module. The optimal value for the pool size is that which reduces the values for these monitored counters. If Percent Used is consistently low, consider decreasing the number of connections in the pool.

Lower settings for the connection pool size (10-30 connections) typically perform better than higher (more than 100) settings. On UNIX platforms, a separate DB2 process is created for each connection. These processes quickly affects performance on systems with low memory, causing errors.

Each Entity EJB transaction requires an additional connection to the database specifically to handle the transaction. Be sure to take this into account when calculating the number of data source connections.

Deadlock can occur if the application requires more than one concurrent connection per thread, and the database connection pool is not large enough for the number of threads. Suppose each of the application threads requires two concurrent database connections and the number of threads is equal to the maximum connection pool size. Deadlock can occur when both of the following are true:

- Each thread has its first database connection, and all are in use.
- Each thread is waiting for a second database connection, and none would become available since all threads are blocked.

To prevent the deadlock in this case, the value set for the database connection pool must be at least one higher of the number of waiting threads in order to have at least one thread complete its second database connection.

The connection pool settings are directly related to the number of connections that the database server is configured to support. If the maximum number of connections in the pool is raised, and the corresponding settings in the database are not raised, the application fails and SQL exception errors are displayed in the stderr.log file.

Prepared statement cache size

The WebSphere Application Server data source optimizes the processing of prepared statements to help make SQL statements process faster. It is important to configure the cache size of the datasource to gain optimal statement execution efficiency. A prepared statement is a precompiled SQL statement that is stored in a prepared statement object. This object is used to efficiently execute the given SQL statement multiple times. If the JDBC driver specified in the data source supports precompilation, the creation of the prepared statement will send the statement to the database for precompilation. Some drivers might not support precompilation and the prepared statement might not be sent until the prepared statement is executed.

If the cache is not large enough, useful entries will be discarded to make room for new entries. In general, the more prepared statements your application has, the larger the cache should be.

The cache size is set from the administrative console using these steps.

1. Click **Resources** -> **JDBC** in the console navigation tree.
2. Click the name of provider from the list of JDBC providers in the workspace.
3. Click the **Datasources** entry in the additional properties pane of the workspace.
4. Click the name of datasource from the list of datasources in the workspace.
5. Use the **Statement Cache Size** field to configure the total cache size.

6. Save the configuration and restart the affected application servers for the change to take effect.

Tivoli Performance Viewer can help tune this setting to minimize cache discards. Use a standard workload that represents a typical number of incoming client requests, use a fixed number of iterations, and use a standard set of configuration settings. Watch the **PrepStmt Cache Discard** counter of the JDBC Connection Pools module. The optimal value for the statement cache size is the setting used to get either a value of zero or the lowest value for PrepStmt Cache Discards.

As with the connection pool size, the statement cache size setting requires resources at the database server. Specifying too large a cache could have an impact on database server performance. It is highly recommended to consult your database administrator for determining the best setting for the prepared statement cache size.

Note: The statement cache size setting defines the maximum number of prepared statements cached per connection, this is different from WebSphere 4.0 where this was specified per container.

EJB container

Method invocations to enterprise beans are only queued for requests coming from remote clients going through the RMI activity service. An example of such a client is an EJB client running in a separate Java Virtual Machine (another address space) from the enterprise bean. In contrast, no queuing occurs if the EJB client (either a servlet or another enterprise bean) is installed in the same JVM that the EJB method runs on and the same thread of execution as the EJB client.

Remote enterprise beans communicate by using the RMI/IIOP protocol. Method invocations initiated over RMI/IIOP are processed by a server-side ORB. The thread pool acts as a queue for incoming requests. However, if a remote method request is issued and there are no more available threads in the thread pool, a new thread is created. After the method request completes the thread is destroyed. Therefore, when the ORB is used to process remote method requests, the EJB container is an open queue, due to the use of unbounded threads.

The ORB thread pool size is configured from the administrative console using these steps.

1. Click **Servers** -> **Application Servers** in the console navigation tree.
2. Click the name of application server from the list of application servers in the workspace.

3. Click the **ORB Service** entry in the additional properties pane of the workspace.
4. Click **Thread Pool** in the additional properties pane of the workspace.
5. Use the **Maximum Size** field to configure the maximum pool size. Note that this only affects the number of threads held in the pool (the actual number of ORB threads can be higher).
6. Save the configuration and restart the affected application server for the change to take effect.

Tivoli Performance Viewer can help tune the ORB thread pool size settings. Use a standard workload that represents a typical number of incoming client requests, use a fixed number of iterations, and use a standard set of configuration settings. Watch the **Percent Maxed** counter of the Object Request Broker submodule of the Thread Pools module. If the value of this counter is consistently in the double-digits, then the ORB could be a bottleneck and the number of threads in the pool should be increased.

The degree to which the ORB thread pool value needs to be increased is a function of the number of simultaneous servlets (that is, clients) calling enterprise beans and the duration of each method call. If the method calls are longer or the applications spend a lot of time in the ORB, consider making the ORB thread pool size equal to the Web container size. If the servlet makes only short-lived or quick calls to the ORB, servlets can potentially reuse the same ORB thread. In this case, the ORB thread pool can be small, perhaps even one-half of the thread pool size setting of the Web container.

Web container

To route servlet requests from the Web server to the Web containers, a transport queue between the Web server plug-in and each Web container is established. The number of client requests accepted by the container is determined by the Web container thread pool. Connection reuse is another factor that influences number of concurrent threads that is processes by the Web container.

Thread pool

The Web container maintains a thread pool to process inbound HTTP(S) requests for resources in the container (e.g.. servlets/JSPs). This is a closed queue as the thread pool is bounded by the maximum thread size.

The Web container thread pool size is configured from the administrative console using these steps.

1. Click **Servers** -> **Application Servers** in the console navigation tree.
2. Click the name of application server from the list of application servers in the workspace.

3. Click the **Web container** entry in the additional properties pane of the workspace.
4. Click the **Thread Pool** entry in the additional properties pane of the workspace.
5. Use the **Maximum Size** field to configure the maximum pool size. Note that in contrast to the ORB, the Web container only uses threads from the pool - hence a closed queue.
6. Save the configuration and restart the affected application server for the change to take effect.

Tivoli Performance Viewer can help tune the Web container thread pool size settings. Use a standard workload that represents a typical number of incoming client requests, use a fixed number of iterations, and use a standard set of configuration settings. Watch the **Percent Maxed** and **Active Threads** counters of the Web container submodule of the Thread Pools module. If the value of the percent maxed counter is consistently in the double-digits, then the Web container could be a bottleneck and the number of threads should be increased. On the other hand if the number of active threads are significantly lower than the number of threads in the pool, consider lowering the thread pool size for a performance gain.

Note: For Linux systems, the recommended value for the Web container maximum thread pool size is 25.

MaxKeepAliveConnections

This parameter describes the maximum number of concurrent connections to the Web container that are allowed to be kept alive, that is, to be processed in multiple requests. The Web server plug-in keeps connections open to the application server as long as it can. However, if the value of this property is too small, performance is negatively impacted because the plug-in has to open a new connection for each request instead of sending multiple requests through one connection.

The maximum number of keep alive connections allowed to the Web container is configured from the administrative console using these steps.

1. Click **Servers** -> **Application Servers** in the console navigation tree.
2. Click the name of application server from the list of application servers in the workspace.
3. Click the **Web Container** entry in the additional properties pane of the workspace.
4. Click **HTTP Transports** in the additional properties pane of the workspace.

5. Click the host link for which to configure the max keep alive connections setting in the host column of the HTTP Transports pane in the workspace.
6. Click **Custom properties** in the additional properties pane of the workspace.
7. Click the **New** button.
8. Enter **MaxKeepAliveConnections** in the Name field and an integer value in the Value field (the recommended starting value is 90% of the maximum threads in the Web container thread pool).
9. Click **OK**, save the configuration and restart the affected application server for the change to take effect.

The value should be at least 90% of the maximum number of threads in the Web container thread pool. If it is 100% of the maximum number of threads in the Web container thread pool, all the threads could be consumed by keep-alive connections leaving no threads available to process new connections.

The **`netstat`** command utility can help tune the maximum keep alive connections setting. Use a standard workload that represents a typical number of incoming client requests, use a fixed number of iterations, and use a standard set of configuration settings. Watch the number of connections in the TIME_WAIT state to the application server port. If the count of TIME_WAITs is consistently in the double-digits it might improve performance to raise the maximum keep alive connections or maximum keep alive requests properties (described in the MaxKeepAliveRequests section). Commands for retrieving the count of TIME_WAITs is shown in Example 21-8. Substitute the port number with the port of the specific application server you want to monitor. Be aware that having both the Web server and application server installed on the same machine would result in a double count of every connection (as the TIME_WAIT state is listed from both the client side and server side by netstat).

Example 21-8

```
On the Windows platform the chain of commands would be:
   netstat -na | find /i "time_wait" | find /c "9080"

On the Unix platform the chain of commands would look like:
   netstat -na | grep -i time_wait | grep -c 9080
```

The application server might not accept a new connection under a heavy load if there are too many sockets in TIME_WAIT state.

If all client requests are going through the Web server plug-in and there are many TIME_WAIT state sockets for the Web container port, the application server is closing connections prematurely, which decreases performance. The application server will close the connection from the plug-in, or from any client, for any of the following reasons:

- The client request was an HTTP 1.0 request.
- The maximum number of concurrent keep-alives was reached.
- The maximum number of requests for a connection was reached.
- A time out occurred while waiting to read the next request or to read the remainder of the current request.

MaxKeepAliveRequests

The maximum number of requests allowed on a single keep-alive connection. This parameter can help prevent denial of service attacks when a client tries to hold on to a keep-alive connection. The Web server plug-in keeps connections open to the application server as long as it can, providing optimum performance.

The maximum number of requests allowed is configured from the administrative console using these steps.

1. Click **Servers** -> **Application Servers** in the console navigation tree.
2. Click the name of the application server from the list of application servers in the workspace.
3. Click the **Web Container** entry in the additional properties pane of the workspace.
4. Click **HTTP Transports** in the additional properties pane of the workspace.
5. Click the host link for which to configure the max keep alive requests setting in the host column of the HTTP Transports pane in the workspace.
6. Click **Custom properties** in the additional properties pane of the workspace.
7. Click the **New** button.
8. Enter **MaxKeepAliveRequests** in the Name field and an integer value in the Value field (the recommended starting value is 100).
9. Click **OK** to store the property.
10. Save the configuration and restart the affected application server for the change to take effect.

A good starting value for the maximum number of requests allowed is 100. If the application server requests are received from the plug-in only, increase this parameter's value. The `netstat` utility can be used to tune the value of maximum keep alive requests as described in the MaxKeepAliveConnections section. If the number of connections in the TIME_WAIT state is too high, consider raising the maximum keep alive requests setting,

Web server

In order to utilize WebSphere Application Server resources on request processing instead of request queuing, the client requests that arrive to the Web server when the WebSphere Application Server is saturated should be queued in the network. All Web servers have settings for configuring the maximum number of concurrent requests accepted. The settings for configuring the maximum number of concurrent clients accepted by the IBM HTTP server are described here. For configuring the maximum concurrent client requests accepted for other Web servers, check the documentation for that specific Web server implementation.

IBM HTTP Server

The maximum number of concurrent requests accepted by the IBM HTTP server is configured by a parameter in the **`httpd.conf`** file. The parameter sets the number of concurrent threads running at any one time within the IBM HTTP Server. Set this value to prevent bottlenecks, allowing just enough traffic through to the application server. The parameter keyword is different on different platforms. On the Windows platform, the configuration parameter keyword is **`ThreadsPerChild`**. On IBM HTTP servers running on the Unix platform the keyword is **`MaxClients`**.

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 21-9.
3. Save the changes and restart the IBM HTTP server.

Example 21-9 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
```

Determining the Web Server maximum concurrency threads setting

A Web server monitor (such as the Windows performance monitor or IBM HTTP Server server-status page described in “Monitoring the IBM HTTP Server” on page 881) can help tune the maximum concurrent thread processing setting for the Web server. Use a standard workload that represents a typical number of incoming client requests, use a fixed number of iterations, and use a standard set of configuration settings. Watch the number of Web server threads going to the Web container and the number of threads accessing static content locally on the Web server. Set the concurrent requests setting to the sum of the Web container thread pool size and the number of static content requests processed by the Web server. If no static content is served from the Web server, the number of requests should be about equal to the Web container thread pool size (depending on how many clients and Web servers are concurrently accessing the same Web container).

21.8.4 Cloning

The capabilities for cloning application servers can be a valuable asset in configuring highly scalable production environments. This is especially true when the application is experiencing bottlenecks that are preventing full CPU utilization of Symmetric Multiprocessing (SMP) servers. When adjusting the WebSphere Application Server system queues in clustered configurations, remember that when a server is added to a cluster, the server downstream receives twice the load. This is illustrated in Figure 21-18.

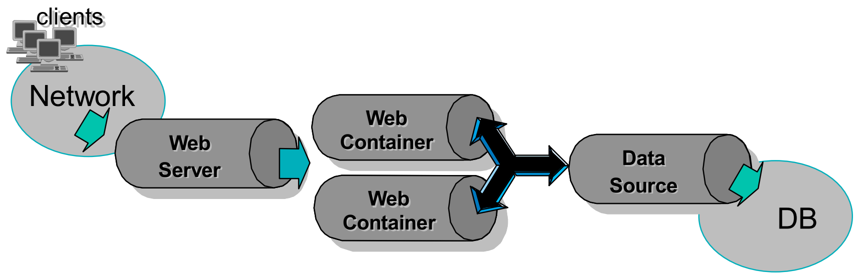


Figure 21-18 Clustering and queuing

Two Web container clones are located between a Web server and a data source. It is assumed the Web server, servlet engines and data source (but not the database) are all running on a single SMP server. Given these constraints, the following queue considerations need to be made:

- Web server queue settings can be doubled to ensure ample work is distributed to each Web container.
- Web container thread pools can be reduced to avoid saturating a system resource such as CPU or another resource that the servlets are using.
- The data source can be reduced to avoid saturating the database server.

Java heap parameters can be reduced for each instance of the application server. For versions of the JVM shipped with WebSphere Application Server, it is crucial that the heap from all JVMs remain in physical memory. Therefore, if a cluster of four JVMs are running on a system, enough physical memory must be available for all four heaps.

21.8.5 Other tuning considerations

The tuning tasks described in this chapter have concentrated on determining and setting WebSphere component queue sizes and optimizing connection throughput. There are a number of other areas that are important to understand for running an efficient WebSphere Application Server environment. Examples of these areas include:

- Java resource usage (CPU, memory etc.)
- Security optimization (socket efficiency, encryption/decryption etc.)
- Object Request Broker connection optimization (object passing, caching etc.)
- Failure avoidance (failover efficiency, cluster optimization etc.)
- Network layer (physical speed, packet sizes, timeouts etc.)

More on these subjects can be found in the WebSphere 5.0 InfoCenter (find a reference to the “WebSphere Application Server, Version 5.0 Tuning Guide” in the “Tuning Performance” section) and on the IBM WebSphere Application Server Whitepapers homepage at
`http://www-3.ibm.com/software/webservers/appserv/whitepapers.html`

Author Comment: More tuning references are needed here and in the bibliography

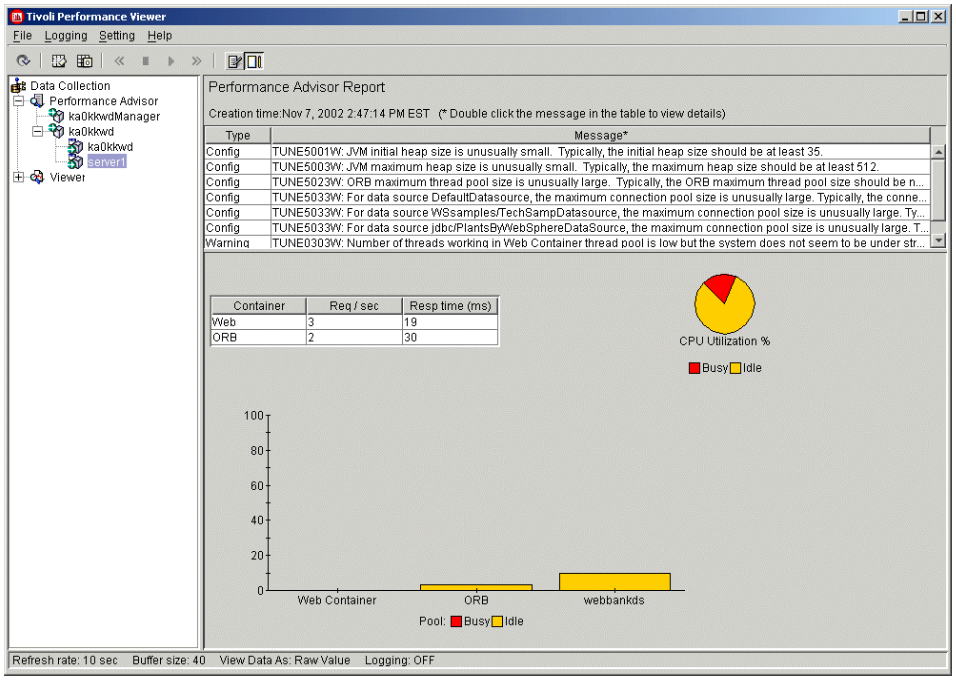
21.9 Tivoli Performance Advisor

An automated tool to perform the tuning analysis is currently available as a technical preview. The analyzer named Tivoli Performance Advisor is going to be incorporated into the Tivoli Performance Viewer and will be able to produce an advisor report with recommendations for tuning specific WebSphere Application Server settings. The queue settings described previously will be part of the recommendations in the advisor report.

The analysis is based on PMI performance data collected and analyzed in realtime or data collected in a previous stress test situation and recorded to a log file. This has the advantage that the advisor analysis task can be performed from a remote client or a standalone workstation without consuming valuable WebSphere Application Server resources.

> **Author Comment:** The advisor is currently invoked from the command line, in the <WAS>\bin directory type:
> `tperfviewer.bat|tperfviewer.sh <host> <port> <protocol> 50tp`

Choosing an application server from the Performance Advisor branch of the navigation tree in the Tivoli Performance Viewer, the advisor layout is as shown in Figure 21-19. The Workspace of the advisor shows the current Web and EJB container thread pools usage counters and related performance data as well as the datasource usage. Also the CPU utilization of the node running the application server is shown. Based on this data and current configuration settings for the WebSphere components, an advisor report is produced and shown in a table on the workspace.



Performance Advisor Report

Creation time:Nov 7, 2002 2:47:14 PM EST (* Double click the message in the table to view details)

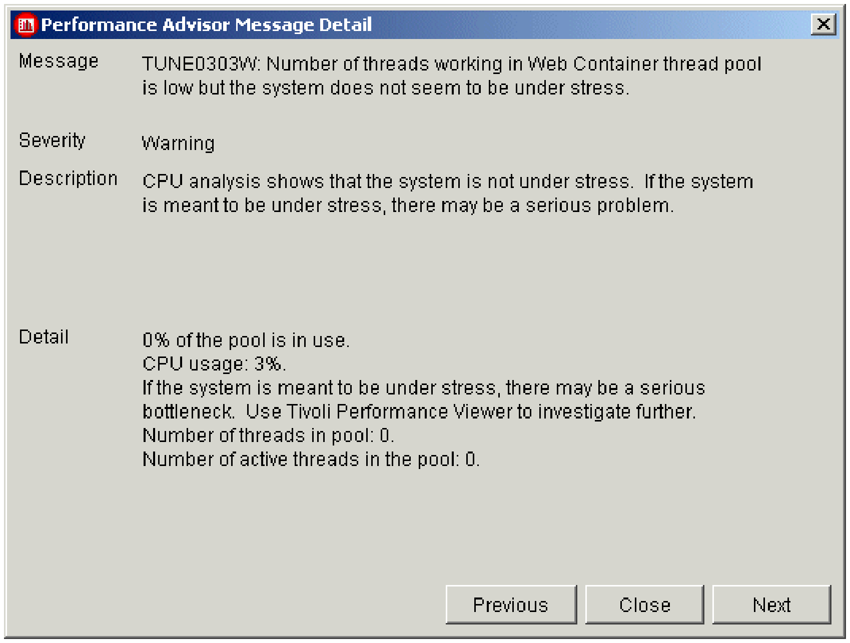
Type	Message*
Config	TUNE5001W: JVM initial heap size is unusually small. Typically, the initial heap size should be at least 35.
Config	TUNE5003W: JVM maximum heap size is unusually small. Typically, the maximum heap size should be at least 512.
Config	TUNE5023W: ORB maximum thread pool size is unusually large. Typically, the ORB maximum thread pool size should be n...
Config	TUNE5033W: For data source DefaultDatasource, the maximum connection pool size is unusually large. Typically, the conne...
Config	TUNE5033W: For data source WSsamples/TechSampDatasource, the maximum connection pool size is unusually large. Ty...
Config	TUNE5033W: For data source jdbc/PlantsByWebSphereDataSource, the maximum connection pool size is unusually large. T...
Warning	TUNE0303W: Number of threads working in Web Container thread pool is low but the system does not seem to be under str...

Container	Req / sec	Resp time (ms)
Web	3	19
ORB	2	30

Refresh rate: 10 sec Buffer size: 40 View Data As: Raw Value Logging: OFF

Figure 21-19 Tivoli Performance Advisor main view

In the idle state an advisor warning message is produced to indicate that performance data was collected and analyzed for a node that was not utilized. The full warning message is shown in Figure 21-20



Message TUNE0303W: Number of threads working in Web Container thread pool is low but the system does not seem to be under stress.

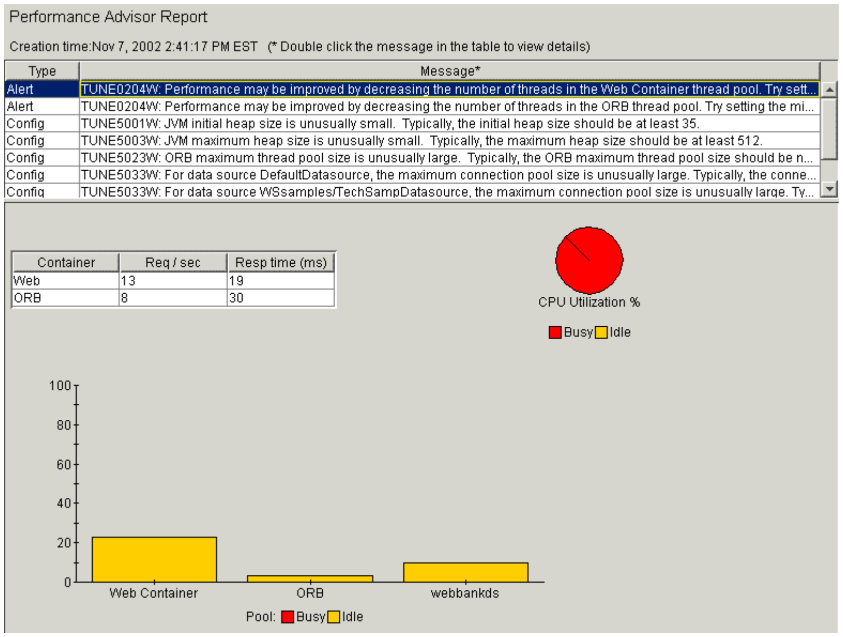
Severity Warning

Description CPU analysis shows that the system is not under stress. If the system is meant to be under stress, there may be a serious problem.

Detail
0% of the pool is in use.
CPU usage: 3%.
If the system is meant to be under stress, there may be a serious bottleneck. Use Tivoli Performance Viewer to investigate further.
Number of threads in pool: 0.
Number of active threads in the pool: 0.

Figure 21-20 Advisor poor performance warning

During a stress load run, the performance data collected and the analysis produced by the advisor might look as shown in Figure 21-21. The ruleset used to produce the advisor report is currently defined in very general terms. As stated previously the right configuration settings are based on the applications running in the WebSphere Application Server environment and the application usage. It might be possible in a future release of the advisor to change the ruleset by the advisor user to match the applications running in the specific WebSphere Application Server environment monitored.



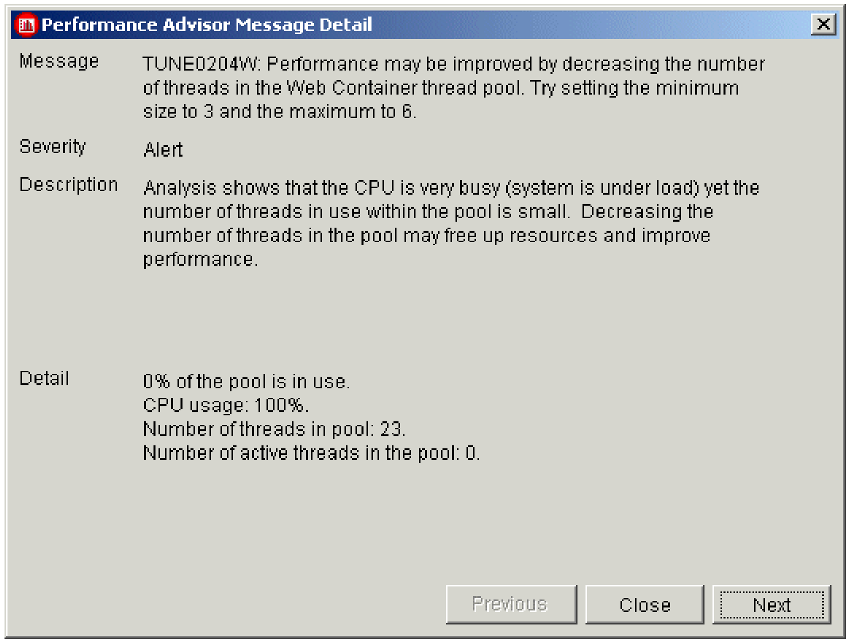
Performance Advisor Report

Creation time:Nov 7, 2002 2:41:17 PM EST (* Double click the message in the table to view details)

Type	Message*
Alert	TUNE0204W: Performance may be improved by decreasing the number of threads in the Web Container thread pool. Try sett...
Alert	TUNE0204W: Performance may be improved by decreasing the number of threads in the ORB thread pool. Try setting the mi...
Config	TUNE5001W: JVM initial heap size is unusually small. Typically, the initial heap size should be at least 35.
Config	TUNE5003W: JVM maximum heap size is unusually small. Typically, the maximum heap size should be at least 512.
Config	TUNE5023W: ORB maximum thread pool size is unusually large. Typically, the ORB maximum thread pool size should be n...
Config	TUNE5033W: For data source DefaultDatasource, the maximum connection pool size is unusually large. Typically, the conne...
Config	TUNE5033W: For data source WSsamples/TechSampDatasource, the maximum connection pool size is unusually large. Ty...

Container	Req / sec	Resp time (ms)
Web	13	19
ORB	8	30

Figure 21-21 Performance advisor runtime analysis



Performance Advisor Message Detail

Message TUNE0204W: Performance may be improved by decreasing the number of threads in the Web Container thread pool. Try setting the minimum size to 3 and the maximum to 6.

Severity Alert

Description Analysis shows that the CPU is very busy (system is under load) yet the number of threads in use within the pool is small. Decreasing the number of threads in the pool may free up resources and improve performance.

Detail 0% of the pool is in use.
CPU usage: 100%.
Number of threads in pool: 23.
Number of active threads in the pool: 0.

Figure 21-22 Performance advisor tuning recommendation

22

Troubleshooting

Problems within an e-business environment can take many forms: poor performance, application unavailability, unexpected results, etc. In order to fix the problem, you must first isolate the problem and understand what it is.

In this chapter, we will introduce you tools and techniques you can use to analyze and correct problems. Included in this chapter is information on:

- Console messages, logs, and traces
- Log Analyzer
- Thread Analyzer and the Java stack trace
- Collector Tool
- First Failure Data Capture (FFDC) logs
- System core dump analysis
- AST(Applicatoin Server Toolkit) and JRas framework.

WonYoung: If you copied or used anything other than redbooks to build this material, please list each paper, book, etc. at the end. We need to give proper credit if due.

22.1 Introduction/Concept/Methodology/History

Table 22-1

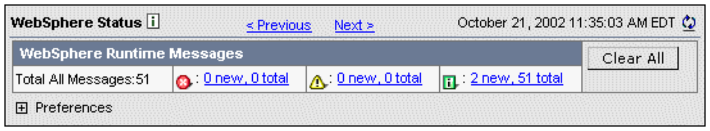
Symptom	Tool or technique
Operational errors	Administrative console messages
Java stack	Log Analyzer

WonYoung .. what would be nice here is a table with possible symptoms or problems on the left, and tools to use on the right. This would be a guide into the following sections.

22.2 Administrative console messages

Runtime status messages are displayed at the WebSphere Status area at the bottom of the administrative console, providing information regarding runtime events and configuration problems.

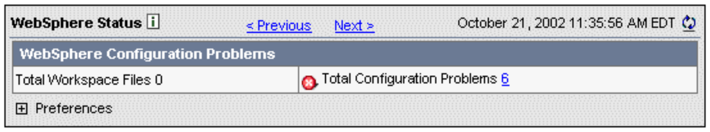
The area consists of two frames: WebSphere Runtime Messages, seen in Figure 22-1 and WebSphere Configuration Problems seen in Figure 22-2. You can toggle between the two with the **Previous** and **Next** links.


WebSphere Status < Previous Next > October 21, 2002 11:35:03 AM EDT

WebSphere Runtime Messages

Total All Messages:51 : 0 new, 0 total : 0 new, 0 total : 2 new, 51 total

Figure 22-1 WebSphere Status: Runtime Messages



WebSphere Status | < Previous | Next > | October 21, 2002 11:35:56 AM EDT

WebSphere Configuration Problems

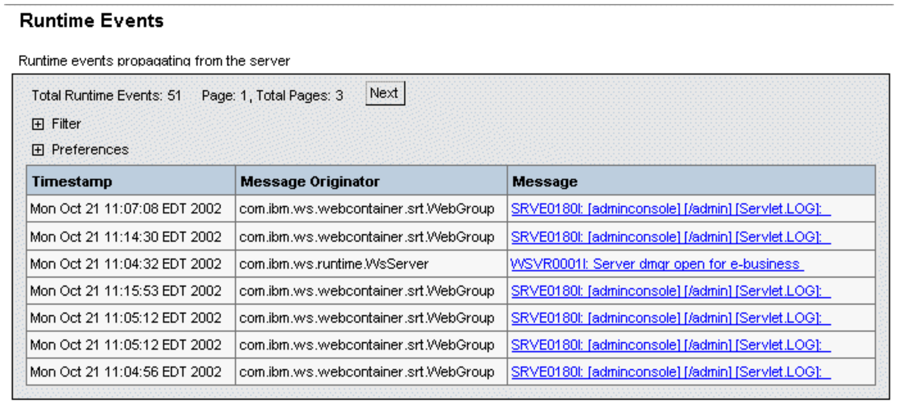
Total Workspace Files 0	Total Configuration Problems 6

⊞ Preferences

Figure 22-2 WebSphere Status: Configuration Problems

The messages on these screens are automatically refreshed at preset intervals. These intervals can be adjusted in the Preferences settings.

To view the messages, select the icon for the type of messages you want to view. Error messages are represented by the **x** icon, warning messages by the **!** icon, and informational messages by the **i** icon.



Runtime Events

Runtime events propagating from the server

Total Runtime Events: 51 Page: 1, Total Pages: 3 Next

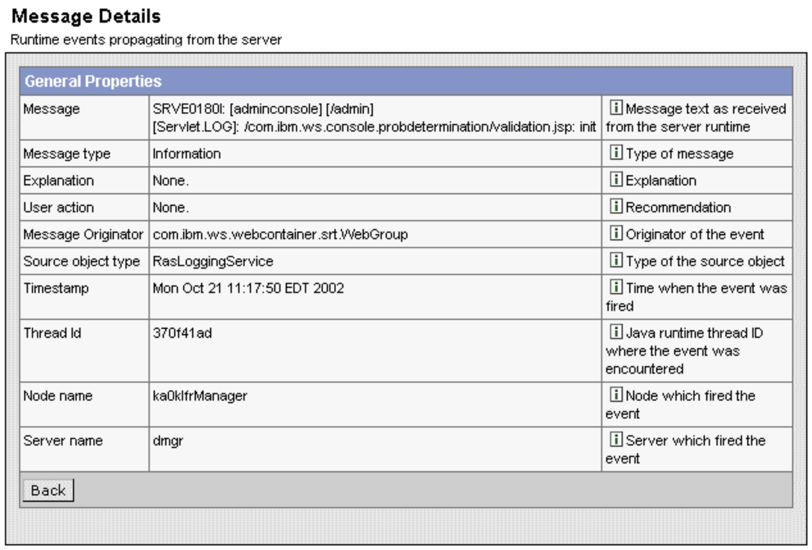
⊞ Filter

⊞ Preferences

Timestamp	Message Originator	Message
Mon Oct 21 11:07:08 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:
Mon Oct 21 11:14:30 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:
Mon Oct 21 11:04:32 EDT 2002	com.ibm.ws.runtime.WsServer	WSVR0001I: Server dmgr open for e-business
Mon Oct 21 11:15:53 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:
Mon Oct 21 11:05:12 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:
Mon Oct 21 11:05:12 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:
Mon Oct 21 11:04:56 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:

Figure 22-3 Runtime Events winsow

When the list of messages appears, you can view the details of the message by selecting the link in the Message column.



Message Details

Runtime events propagating from the server

General Properties		
Message	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]: /com.ibm.ws.console.probdetermination/validation.jsp: init	Message text as received from the server runtime
Message type	Information	Type of message
Explanation	None.	Explanation
User action	None.	Recommendation
Message Originator	com.ibm.ws.webcontainer.srt.WebGroup	Originator of the event
Source object type	RasLoggingService	Type of the source object
Timestamp	Mon Oct 21 11:17:50 EDT 2002	Time when the event was fired
Thread Id	370f41ad	Java runtime thread ID where the event was encountered
Node name	ka0klfrManager	Node which fired the event
Server name	dmgr	Server which fired the event

Figure 22-4 Message Details

WonYoung: What causes these messages to go away? Does the log wrap or can you clear them somehow? Are they archived and stored for processing by some other tool? I'm not sure at this time. I'll check it.

22.3 Log files

WebSphere Application Server can write system messages to several general purpose logs. These include :

- **JVM logs** are created by redirecting the `System.out` and `System.err` streams of the JVM. By default, these files are stored as `installation_root/logs/server_name/SystemOut.log` and `SystemErr.log`.
- **Process (native) logs** are created by redirecting the `stdout` and `stderr` streams of the process's native module(.dlls, .so, UNIX libraries, and other JNI native modules), including the JVM native code itself. These logs can contain information relating to problems in native code or diagnostic information written by the JVM. By default, these files are stored as `installation_root/logs/applicationServerName/native_stderr.log` and `native_stdout.log`.

- **Service logs** is a special log, by default named `activity log`, written in a binary format. You cannot view the log directly using a text editor. A special tools, Log Analyzer or Showlog tool, provides additional diagnostic capabilities.

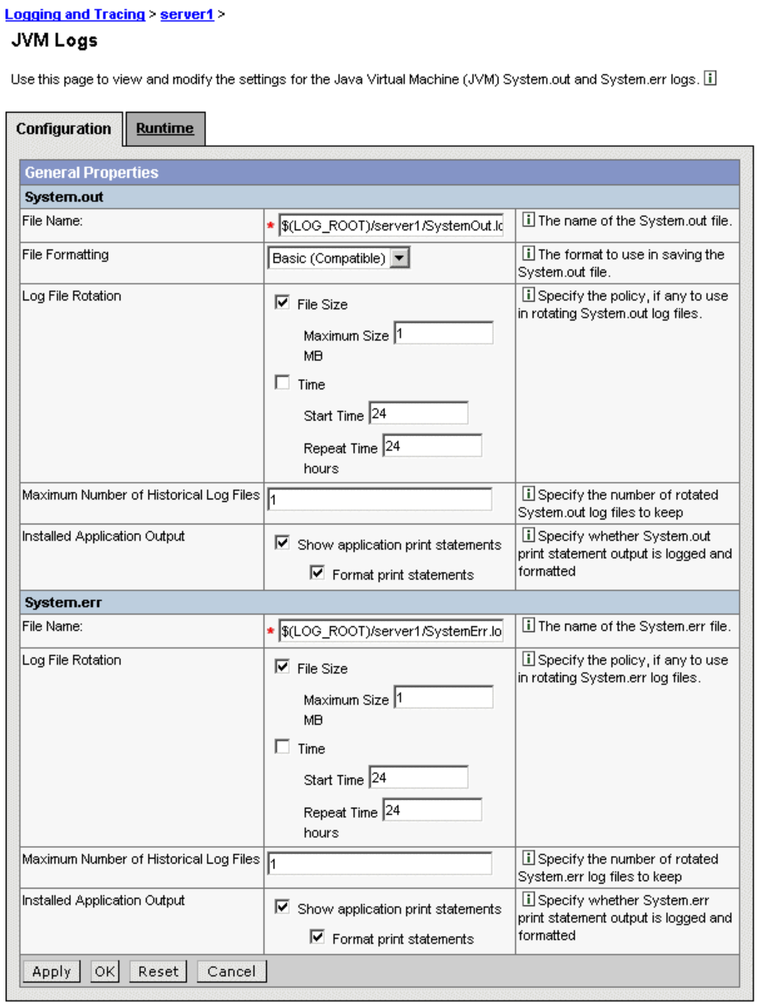
22.3.1 JVM (standard) logs

The JVM (standard) logs are created by redirecting the `System.out` and `System.err` streams. WebSphere Application Server writes formatted messages to the `System.out` stream. In addition, applications and other code can write to these streams using the `print()` and `println()` methods defined by the streams. Some JDK built-ins such as the `printStackTrace()` method on the `Throwable` class can also write to these streams. Typically, the `System.out` log is used to monitor the health of the running application server. The `System.err` log contains exception stack trace information that is useful when performing problem analysis.

Configuring the JVM logs

To view and modify the settings for the JVM `System.out` and `System.err` logs, use the administrative console as following steps:

1. Start the administrative console.
2. Click **Troubleshooting** -> **Logs and Trace** in the navigation tree.
3. Click the **server** which you want to view and click the **JVM Logs**
4. Select the **Configuration** tab.
5. Scroll through the panel to display the attributes as shown in Figure 22-5.
6. Change the appropriate configuration attributes and click **Apply**.
7. Save your configuration changes.


Logging and Tracing > server1 >

JVM Logs

Use this page to view and modify the settings for the Java Virtual Machine (JVM) System.out and System.err logs.

Configuration | Runtime

General Properties

System.out

File Name:	* $(LOG_ROOT)/server1/SystemOut.lo	The name of the System.out file.
File Formatting	Basic (Compatible)	The format to use in saving the System.out file.
Log File Rotation	☑ File Size Maximum Size 1 MB ☐ Time Start Time 24 Repeat Time 24 hours	Specify the policy, if any to use in rotating System.out log files.
Maximum Number of Historical Log Files	1	Specify the number of rotated System.out log files to keep
Installed Application Output	☑ Show application print statements ☑ Format print statements	Specify whether System.out print statement output is logged and formatted

System.err

File Name:	* $(LOG_ROOT)/server1/SystemErr.lo	The name of the System.err file.
Log File Rotation	☑ File Size Maximum Size 1 MB ☐ Time Start Time 24 Repeat Time 24 hours	Specify the policy, if any to use in rotating System.err log files.
Maximum Number of Historical Log Files	1	Specify the number of rotated System.err log files to keep
Installed Application Output	☑ Show application print statements ☑ Format print statements	Specify whether System.err print statement output is logged and formatted

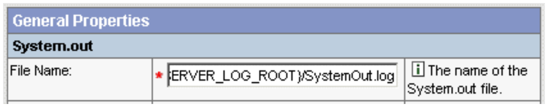
Apply | OK | Reset | Cancel

Figure 22-5 Configuring the JVM logs setting

- **File Name:**

  Specifies the name of the `System.out` or `System.err` file. The file name specified on the Configuration tab must have one of the following values:

- **filename**: The name of a file in the file system.You can use a fully qualified file name. If the file name is not fully qualified, it is considered to be relative to the current working directory for the server.

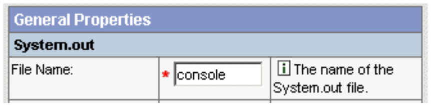


General Properties

System.out

File Name:	* ERVER_LOG_ROOT}/SystemOut.log	The name of the System.out file.

- **console**: This is a special file name used to redirect the stream to the corresponding process stream. If this value is specified for `System.out`, the file is redirected to `stdout`. If this value is specified for `System.err`, the file is redirected to `stderr`.

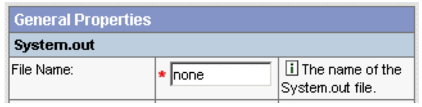


General Properties

System.out

File Name:	* console	The name of the System.out file.

- **none**. Discards all data written to the stream. Specifying `none` is equivalent to redirecting the stream to `dev/null` on a Unix system.



General Properties

System.out

File Name:	* none	The name of the System.out file.

WonYoung: It isn't clear to me how to specify console. Do you just type "console" in the field instead of a file name? Yes. It would be clear if you recapture the screen and have console specified for the system.out, and a file specified for system.err. Ok, is it clear now?

- **File formatting:**

  Specifies the format to use in saving the `System.out` file, Basic (Compatible) or Advanced

  Basic format used in earlier versions of WebSphere Application Server.
  `<timestamp><threadID><shortName><eventType>[class][method]<message>`

  Advanced format extends the basic format by adding information about an event, when possible.
  `<timestamp><threadID><eventType><UOW><source=longName>[class][method]<Organization><Product><Component><message>`

WonYoung: Is there any reason to pick one over the other? (for example, do you need a particular format so the logs can be used in a tool? Or do you just read these as text files so it is just a preference thing?
JVM logs are <stderr> and <stdout> stream log. Most of case, this logs are written by user applications System.out.println() and System.err.println() method. In my opinion, Advanced format has UOW item, so it can be used to debug or trace their application logic in distributed configuration among several clone or machines.

- **Log file rotation:**

  A self-managing log file will write messages to a file until some criteria, either size or time, is reached. At the specified time or when the file reaches the specified size,the current file is closed and renamed to a name consisting of the current name plus a timestamp. The stream then reopens a new file reusing the original name and continues writing.

  – File size

    If this option is selected, the file automatically performs self-maintenance by rolling over the file when it reaches the specified maximum size.

  – Maximum size

    This attribute specifies the maximum size in megabytes to which the file is allowed to grow.

  – Time

    Selecting this attribute allows the log file to manage itself based on the age of the file. If this option is selected, the file will roll itself over after the specified time period.

  – Start Time

    Specifies the hour of the day, from 1 to 24, from which the periodic rollover algorithm commences. The periodic rollover algorithm uses this hour to load the algorithm at application server startup. Once started, the rollover algorithm runs without adjustment until the application server is stopped.

  – Rollover period

    Specifies the number of hours after which the log file will be rolled over to a new filename. Valid values are from 1 to 24.

    Note that if both File Size and Time are selected, the file is rolled over based on the criteria that is met first.

- Maximum Number of Historical Log Files

  Specifies the number of rolled over files to keep.

- Installed Application Output

  Specifies whether the `System.out` or `System.err` print statements issued from the application code are logged and formatted.

  – Show application print statements

    Causes application messages written to this stream using the `print` and `println` stream methods to be shown. This will have no effect on system messages written to the stream by the WebSphere Application Server.

  – Format print statements

    Causes application messages written to this stream using the `print` and `println` stream methods to be formatted like WebSphere system messages.

The JVM logs are written as plain text files, so you can open and view the files directly using your own editor . Another way to view the JVM logs is using the administative console at the **Runtime** tab as shown in Figure 22-5 on page 908. This supports viewing the JVM logs from a remote machine.

JVM log message formats

Analyzing logs and understanding the meaning is a significant step in the problem determininination process. Messages logged by application server conponents and associated IBM products start with a unique message indentifier that indicates the component or application that issued the message.

The following sample illustrates the basic format of a log or trace entry.

Example 22-1 JVM logs(Basic format)

```
[10/16/02 13:37:35:516 EDT] 50e907e8 WebContainer  E SRVE0146E: Failed to Start
Transport on host , port 9090. The most likely cause is that the port is
already in use. Please ensure that no other applications are using this port
and restart the server. com.ibm.ws.webcontainer.exception.TransportException:
Failed to start transport http: java.net.BindException: Address in use: bind
```

A description of each field is shown in Table 22-2.

Table 22-2 Log entry format description

Field	Example	Description
Timestamp	`[10/16/02 13:37:35:516 EDT]`	The timestamp in fully qualified date, time and TimeZone format
Thread ID	`50e907e8`	The thread ID or the hash code of the thread issuing this message

Field	Example	Description
Component	`WebContainer`	The short name of component issuing this message
Event Type	`E`	The type of the message or trace evnet. Possible values include: **A** Audit **I** Informational **W** Warning **E** Error **F** Fatal **O** System.out by the user application or internal components **R** System.err by the user application or internal components **u** A special type used by the message logging component of the WebSphere Application Server run time **Z** A placeholder to indicate the type was not recognized
Message ID	`SRVE0146E`	The identifier of the message.
Message	`Failed to Start Transport on host , port 9090. The most likely cause is that the port is already in use. Please ensure that no other applications are using this port and restart the server.`	The text of the message and message arguments

The message identifier(Message ID) can be either 8 or 9 characters in length and has the form:

CCCC1234X

- **CCCC** is a four character alphabetic component or application identifier.
- **1234** is a four character numberic identifier used to identify the specific message for that component.
- **X** is and optional alphabetic severity indicator.(**I**=Informational, **W**=Warning, **E**=Error)

To view the message IDs or the meaning of the messages generated by WebSphere Application Server components, select the **Quick Reference** view on the InfoCenter and expand the topic **Messages**.

22.3.2 Process (native) logs

The `stdout` and `stderr` streams by native modules (.dlls, .so, UNIX libraries, and other JNI modules) are redirected to these log files at application server startup. By default, these files are stored as `installation_root/logs/applicationServerName/native_stderr.log` and `native_stdout.log`.

To view or change settings, click **Servers -> *server_name* -> Process Definition -> Process Logs**.

Note: This is a change from previous versions of WebSphere Application Server, which by default had one log file for both JVM standard output and native standard output, and one log file for both JVM standard error and native error output.

22.3.3 IBM service (activity) log

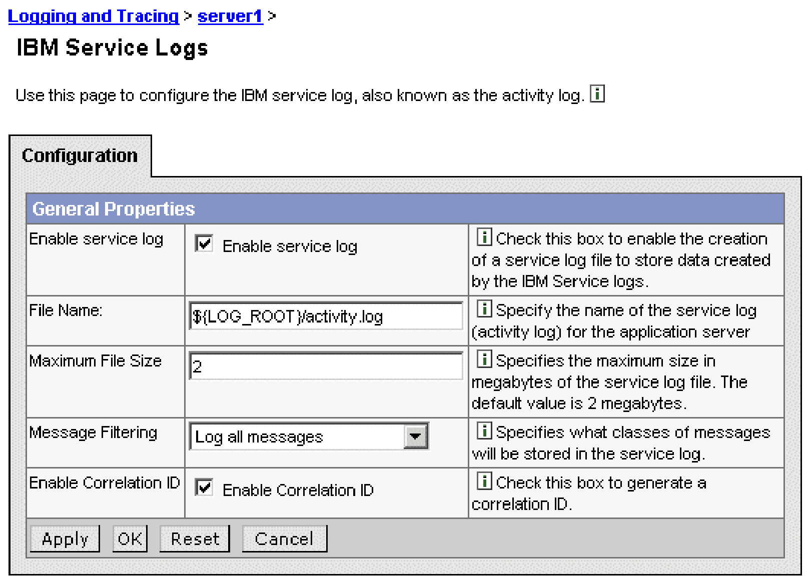
The IBM service log is a special log written in a binary format that captures events that show a history of WebSphere Application Server's activities, also known as the *activity log*.

By default, the IBM service log is shared among all server processes for a node. The configuration values for the IBM service log are inherited by each server process from the node configuration. You can configure a separate IBM service log for each server process by overriding the configuration values at the server level. Follow these steps to view or change the IBM service log settings:

1. Start the administrative console.
2. Click **Troubleshooting -> Logging and Tracing -> *server_name* -> IBM Service Logs**.
3. Select the Enable box to enable the service log, clear the check box to disable the log as shown in Figure 22-6.
4. Set the name for the service log. The default name is `activity.log`. If the name is changed, the run time requires write access to the new file, and the file must use the `.log` extension.
5. Set the maximum file size. Specifies the number of megabytes to which the file can grow. When the file reaches this size, it wraps, replacing the oldest data with the newest data.
6. Set the message filter level to the desired state.
7. Check **Enable Correlation ID** check box to generate a correlation ID. This option 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.

8. Save the configuration.
9. Restart the server to apply the configuration changes.


Logging and Tracing > server1 >

IBM Service Logs

Use this page to configure the IBM service log, also known as the activity log.

Configuration

General Properties

Enable service log	✓ Enable service log	Check this box to enable the creation of a service log file to store data created by the IBM Service logs.
File Name:	${LOG_ROOT}/activity.log	Specify the name of the service log (activity log) for the application server
Maximum File Size	2	Specifies the maximum size in megabytes of the service log file. The default value is 2 megabytes.
Message Filtering	Log all messages	Specifies what classes of messages will be stored in the service log.
Enable Correlation ID	✓ Enable Correlation ID	Check this box to generate a correlation ID.

Figure 22-6 IBM Service log

To view the service log (by default, the name is `activity.log`), use the `showlog` command in the `<WAS_HOME>/bin` directory. This `showlog` command dumps the binary log file to standard out or a file.

```
Usage: showlog binaryFilename [outputFilename]
```

`outputFilename` is optional. If no filename is given, showlog dumps the service log file to standard out.

```
Example: showlog ../logs/activity.log
```

Another powerful way to view the service log file is to use Log Analyzer (described later in).

22.4 Traces

Tracing can be useful if you have problems with particular components of WebSphere Application Server, clients and other processes, and the log files don’t provide you with enough information to determine what the problem exactly is.

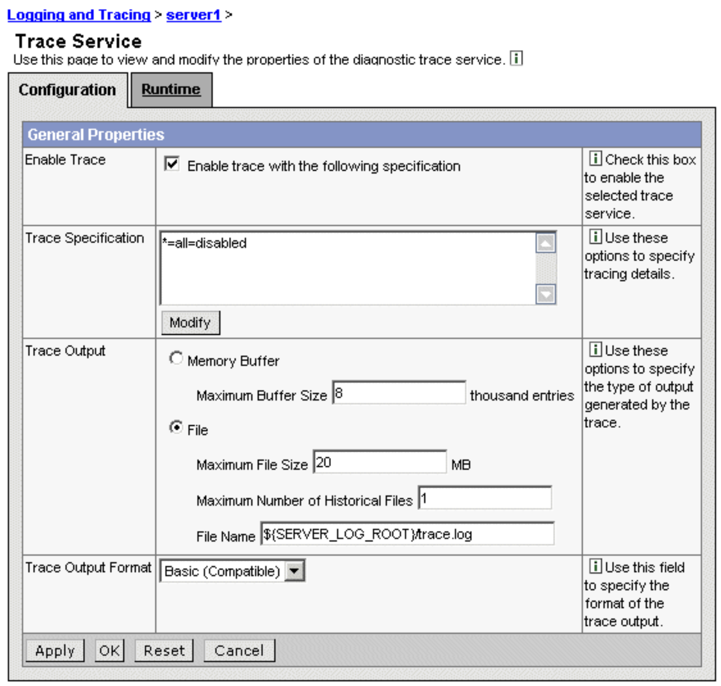
22.4.1 Diagnostic trace service

By default, the trace for all WebSphere Application Server components is disabled.

Enabling trace at server startup

The trace configuration settings are read at server startup time and used to configure the trace service. To configure or change the diagnostic trace settings, do the following steps:

1. Start the administrative console.
2. Click **Troubleshooting** -> **Logging and Tracing** in the console navigation tree.
3. Click ***server*** -> **Diagnostic Trace**.
4. Click **Configuration** tab as shown in Figure 22-7.
5. Select **Enable Trace** check box to enable trace.
6. Set the proper trace strings (described in “Trace string specification” on page 916).
7. Select whether to direct trace output to either a file or an in-memory circular buffer.
8. (Optional) If the in-memory buffer is selected, set the size of the buffer, specified in thousands of entries(described in , “Managing the application server trace service” on page 919).
9. (Optional) If a file is selected, set the maximum size in megabytes to which the file should be allowed to grow. When the file reaches the size, the existing file will be closed, renamed and a new file with the original name reopened.
10. Select the desired format for the generated trace.
11. Save the changed configuration.
12. Start (or restart) the server.


Logging and Tracing > server1 >

Trace Service

Use this page to view and modify the properties of the diagnostic trace service.

General Properties

Enable Trace	[✓] Enable trace with the following specification	Check this box to enable the selected trace service.
Trace Specification	*=all=disabled	Use these options to specify tracing details.
Trace Output	○ Memory Buffer Maximum Buffer Size 8 thousand entries ● File Maximum File Size 20 MB Maximum Number of Historical Files 1 File Name ${SERVER_LOG_ROOT}/trace.log	Use these options to specify the type of output generated by the trace.
Trace Output Format	Basic (Compatible)	Use this field to specify the format of the trace output.

Figure 22-7 Trace service configuration

Note: You can also enable a trace for a running server. See “Enabling trace on a running server” on page 918.

Trace string specification

Trace strings must conform to a specific grammar for processing by the trace service:

```
COMPONENT_TRACE_STRING[:COMPONENT_TRACE_STRING]*
```

where

- `COMPONENT_TRACE_STRING = COMPONENT_NAME=LEVEL=STATE[,LEVEL=STATE]*`
- `LEVEL = all | entryExit | debug | event`
- `STATE = enabled | disabled`
- `COMPONENT_NAME = COMPONENT | GROUP`

The `COMPONENT_NAME` is the name of a component or group registered with the trace service. Typically, WebSphere Application Server components register using a fully qualified Java classname, for example `com.ibm.servlet.engine.ServletEngine`. In addition, you can use a wildcard character of asterisk (*) to terminate a component name and indicate multiple classes or packages. For example, use a component name of `com.ibm.servlet.*` to specify all components whose names begin with `com.ibm.servlet`.

Examples of legal trace strings include:

```
com.ibm.ejs.ras.ManagerAdmin=debug=enabled
com.ibm.ejs.ras.ManagerAdmin=all=enabled,event=disabled
com.ibm.ejs.ras.*=all=enabled
com.ibm.ejs.ras.*=all=enabled:com.ibm.ws.ras=debug=enabled,entryexit=enable
d
```

Trace strings cannot contain blanks.

Trace strings are processed from left to right. Specifying a trace string like

```
abc.*=all=enabled,event=disabled
```

first enables all trace for all components whose names start with abc, then disables event tracing for those same components. This means that the trace string

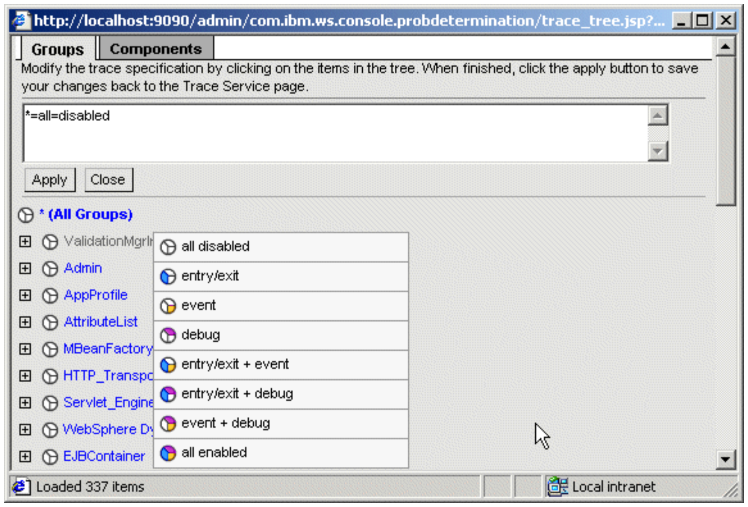
```
abc.*=all=enabled,event=disabled
```

is equivalent to

```
abc.*=debug=enabled,entryexit=enabled
```

Graphical trace interface

An alternative to entering the trace string directly is to use the graphical trace interface to generate the trace strings that specifies the components, packages, or groups to trace. Click the **Modify** button on the Configuration tab (see Figure 22-7) to start the graphical trace interface. Then, you can generate trace strings as shown in Figure 22-8.


http://localhost:9090/admin/com.ibm.ws.console.probdetermination/trace_tree.jsp?...
Groups
Components

Modify the trace specification by clicking on the items in the tree. When finished, click the apply button to save your changes back to the Trace Service page.

```
*=all=disabled
```

Figure 22-8 Graphical trace interface

Trace output format

You can specify one of three levels for trace output:

- **Basic** (Compatible) preserves only basic trace information. Select this option to minimize the amount of space taken up by the trace output.
- **Advanced** preserved more specific trace information. Select this option to see detailed trace information for use in troubleshooting and problem determination.
- **Log Analyzer** preserved trace information in a format that is compatible with the Log Analyzer tool. Select this option if you want to use the trace output as input to the Log Analyzer tool.

Enabling trace on a running server

You can also trace a server that is already active:

1. Start the administrative console.
2. Click **Troubleshooting** -> **Logging and Tracing** in the console navigation tree.
3. Click ***server* -> Diagnostic Trace**
4. Select the **Runtime** tab.
5. (Optional) Select the **Save Trace** check box if you want to write your changes back to the server configuration.

6. Change the existing trace state by changing the trace specification to the desired state. You can click the **Modify** button to use the graphical trace interface.
7. (Optional) Configure the trace output if a change from the existing one is desired.
8. Click **Apply**.

Managing the application server trace service

On an application server, trace output can be directed either to a file or to an in-memory circular buffer. If trace output is directed to the in-memory circular buffer, it must be dumped to a file before it can be viewed, as illustrated in Figure 22-9.

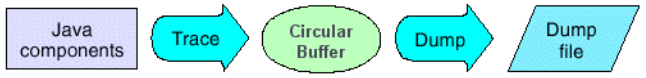


Figure 22-9 In-memory circular buffer

You can manage the trace service for a server process while the server is stopped or while it is running. Steps for this task:

1. Start the administrative console.
2. Click **Troubleshooting** -> **Logging and Tracing** in the console navigation tree, then click ***server*** -> **Diagnostic Trace**
3. If the server is running, select the **Runtime** tab. If the server is stopped, select the **Configuration** tab.
4. (Optional) For a running server, check the Save trace check box to write your changes back to the server configuration.
5. If Save trace is not selected, the changes you make will apply only for the life of the server process that is currently running.
6. Perform the desired operation:
   a. Enter the file name and click **Dump** to dump the in-memory circular buffer.
   b. To change the trace destination from a file to the in-memory circular buffer or to a different file, or to change from the in memory circular buffer to a file, select the appropriate radio buttons, then click **Apply**.
   c. To change the format in which trace output is generated, select the appropriate value from the drop-down list.

Looking at trace output

Example 22-2 is the trace output example(basic format).

Example 22-2 Trace output(Basic format)

```
[11/11/02 16:40:27:625 EST] 1c5907d1 JspServlet    d jspUri: /jvmlogs.jsp, 
lastCheck (last time checked): 0, cachedJSPLastModTimestamp: 0, firstTime: 
true, reloadEnabled: true, reloadInterval: 3000, debugEnabled: false
[11/11/02 16:40:27:625 EST] 1c5907d1 JspServlet    d Inside main compile/reload 
block.  firstTime: true, reload interval exceeded: true
[11/11/02 16:40:27:625 EST] 1c5907d1 JspServlet    d  creating jsw.outDir
[11/11/02 16:40:27:625 EST] 1c5907d1 JspServlet    d  jsw.outDir: 
C:\WebSphere\AppServer\temp\ka0klfr1\server1\DefaultApplication\DefaultWebAppli
cation.war\
...
[11/11/02 16:40:27:656 EST] 1c5907d1 JspServlet    d  loaded class: 
org.apache.jsp._jvmlogs
[11/11/02 16:40:27:672 EST] 1c5907d1 HttpJspBase   > init
[11/11/02 16:40:27:672 EST] 1c5907d1 WebGroup      I SRVE0180I: [Default Web 
Application] [/] [Servlet.LOG]: /jvmlogs.jsp: init
[11/11/02 16:40:27:672 EST] 1c5907d1 HttpJspBase   < init
```

The EventType is a one character field in lower case that indicates the type of the trace event. Possible values include:

```
> method entry
< method exit
e event
d debug
m dump
u unconditional
Z type was not recognized
```

Author Comment: In my experience, "enabling trace" is only needed when we have to open PMR. That means we, most customer or local IBM tech. support engineers, do not understand the output trace file nor the meaning of the trace. This trace file is mostly used to debug and trace the WebSphere Engine Product not the user application. Therefore, detailed information is not needed. It is enough for them to turn on and off the trace and how to send it to the IBM lab.

22.4.2 Web server logs and traces

Plug-in logs and traces

The Web server plug-in configuration file controls what content is transferred from the Web server to an application server. This file must be regenerated when server, cluster, HTTP transport, or virtual host alias configurations are changed. The generated `plugin-cfg.xml` file is placed in the `config` directory of the WebSphere installation. If your web server is located on a remote machine, you must manually move this file to that machine. The following is an example `plugin-cfg.xml` file.

Example 22-3 <WAS_HOME>/config/plugin-cfg.xml

```
<?xml version="1.0"?>
<Config>
    <Log LogLevel="Trace" Name="...\AppServer\logs\http_plugin.log"/>
    ...
</Config>
```

The plug-in log file is created by the WebSphere plug-in running in the Web server process. The plug-in log contains error and informational messages generated from the Web server plug-in. Different levels of information can be placed in this log via the plug-in configuration file, `plugin-cfg.xml`. The default setting is `Error` . Possible values are `Trace`, `Warn` and `Error`.

You can not change the log level on the administrative console . You have to change the log level manually by your own text editor. However, note that when you “Update web server plugin configuration“ on the administrative console, the log level will be updated again as the default value “Error”.

If the Web server is configured on a different machine from the application server, then the `http_plugin.log` and the `plugin-cfg.xml` files reside on the machine where the Web server is installed.

IBM HTTP Server logs

IBM HTTP Server(IHS) is based on the Apache HTTP Server. IBM has enhanced the Apache-powered HTTP server, for example, IBM has added SSL for secure transactions and offers full support when part of WebSphere Application Server bundle. There are two main log files, `CostomLog` and `ErrorLog`, which are specified by the configuration file, `<IHS_HOME>/conf/httpd.conf`, as shown in Example 22-4.

Example 22-4 IHS configuration file, httpd.conf

```
...
# ErrorLog: The location of the error log file. If this does not start
# with /, ServerRoot is prepended to it.

ErrorLog /usr/IBMHttpServer/logs/error_log

# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.

LogLevel warn
```

```
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# The location of the access logfile (Common Logfile Format).
# If this does not start with /, ServerRoot is prepended to it.

CustomLog /usr/IBMHttpServer/logs/access_log common
...
```

Enabling trace on the IHS

To trace and diagnose some problem of the web server, you can change the `LogLevel` for the error log file. The possible values are denoted in the file.

To ignore unnecessary log records in the access log file

Like all Web Servers, it records all HTTP access traffic in a log file, which on a busy site can easily grow to 1GB in a week. If it is not necessary to record some kinds of access entries, for instance, static contents or image files(*.gif,*.jpg), set it up as following.

Example 22-5 Ignoring image entries in the access_log

```
...
SetEnvIf Request_URI \.gif$ ignore=gif
SetEnvIf Request_URI \.jpg$ ignore=jpg

#CustomLog /usr/IBMHttpServer/logs/access_log common
CustomLog /usr/IBMHttpServer/logs/access_log common env!=ignore
...
```

To trace the elapsed times

The default `LogFormat` is `common` type for the access log file which does not have service elapsed time. The elapsed time is very useful information to understand and diagnose application performance problem. Enable the elapsed time option, %T, at the `LogFormat` entry.

Example 22-6 Elapsed time option %T

```
...
#LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

```
LogFormat "%h %l %u %t \"%r\" %>s %b %T" common
...
```

The unit of elapsed time is second.

22.5 Log Analyzer

The Log Analyzer is a GUI tool that permits the user to view any logs generated with loganalyzer TraceFormat, such as the IBM service log file and other traces using this format, as discussed in “Trace output format” on page 918. It can take one or more service logs 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 22-10.

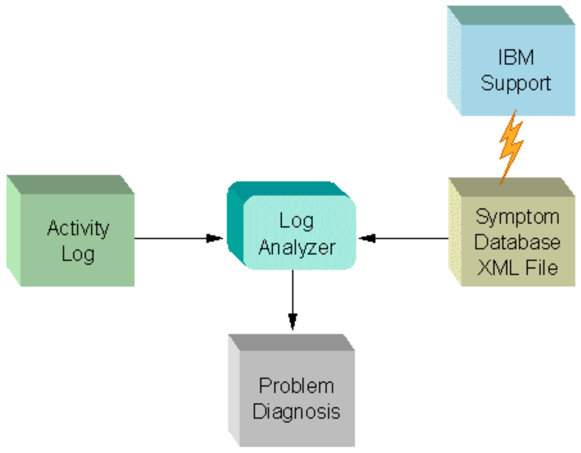


Figure 22-10 Log Analyzer concept

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 929 for details.

22.5.1 Starting Log Analyzer

To start using the Log Analyzer:

1. Select Start -> Programs -> IBM WebSphere -> Application Server 5.0 -> **Log Analyzer** from the Windows start menu, as shown in Figure 22-11.

   The Log Analyzer can also be started from the <WAS_HOME>/bin directory using the **`waslogbr.bat/sh`** command.

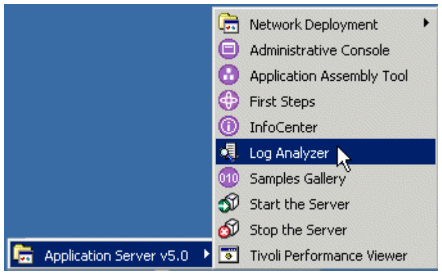
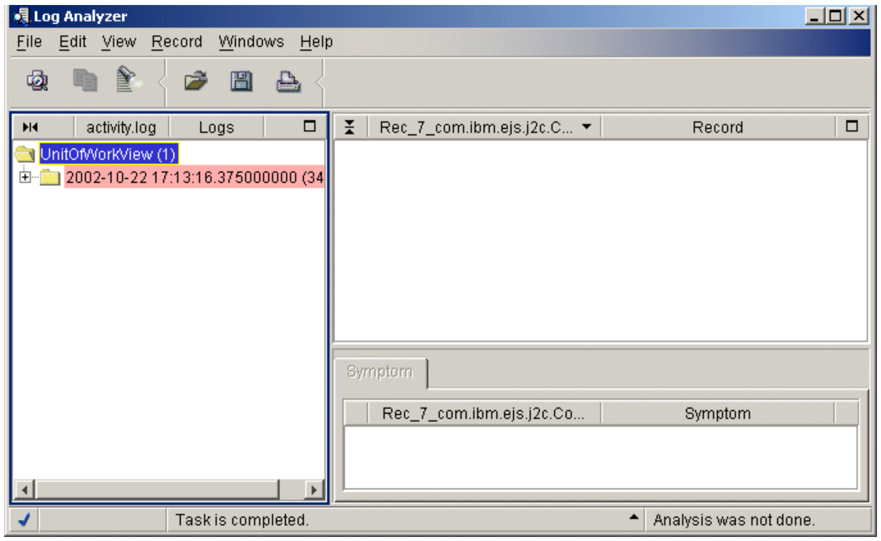


Figure 22-11 Starting Log Analyzer

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 22-12. As discussed in Section 22.3.3, “IBM service (activity) log” on page 913, all application servers and the administrative server write log records to this file.

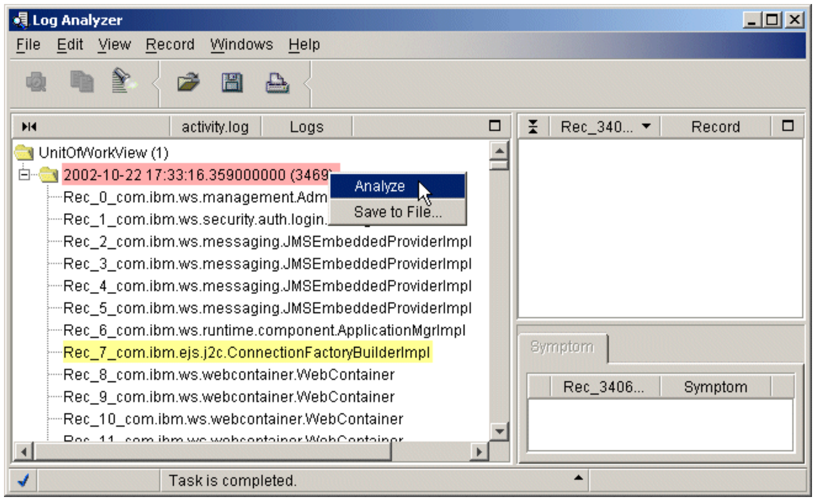

Log Analyzer
File Edit View Record Windows Help
activity.log
Logs
Rec_7_com.ibm.ejs.j2c.C...
Record
Symptom
Rec_7_com.ibm.ejs.j2c.Co...
Symptom

```
UnitOfWorkView (1)
  2002-10-22 17:13:16.375000000 (34
```

```
Task is completed.                    Analysis was not done.
```

Figure 22-12 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 22-13.

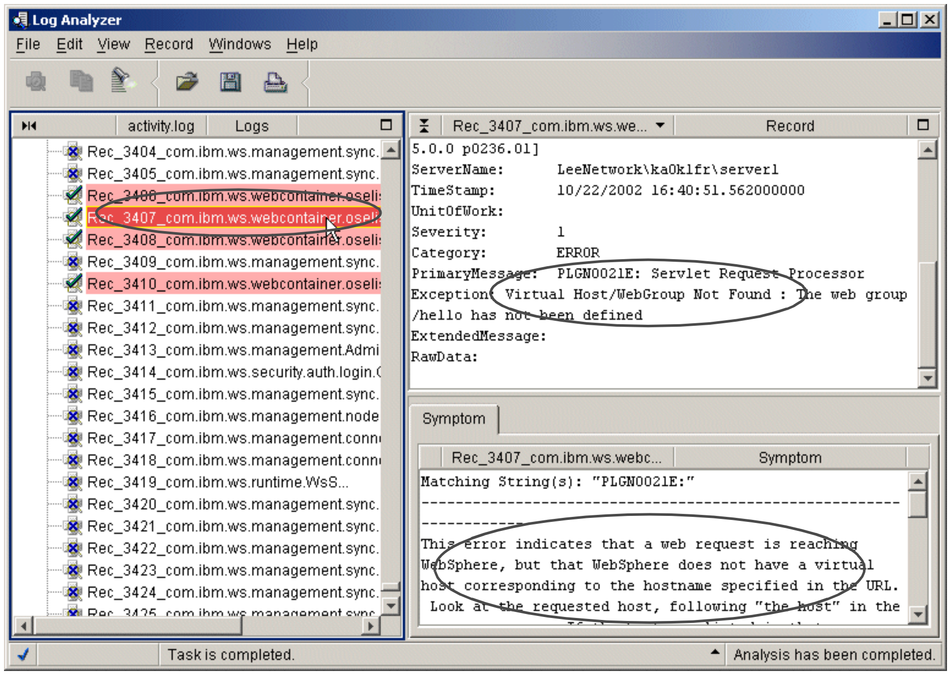


```
UnitOfWorkView (1)
  2002-10-22 17:33:16.359000000 (3469
    Rec_0_com.ibm.ws.management.Adm
    Rec_1_com.ibm.ws.security.auth.login.
    Rec_2_com.ibm.ws.messaging.JMSEmbeddedProviderImpl
    Rec_3_com.ibm.ws.messaging.JMSEmbeddedProviderImpl
    Rec_4_com.ibm.ws.messaging.JMSEmbeddedProviderImpl
    Rec_5_com.ibm.ws.messaging.JMSEmbeddedProviderImpl
    Rec_6_com.ibm.ws.runtime.component.ApplicationMgrImpl
    Rec_7_com.ibm.ejs.j2c.ConnectionFactoryBuilderImpl
    Rec_8_com.ibm.ws.webcontainer.WebContainer
    Rec_9_com.ibm.ws.webcontainer.WebContainer
    Rec_10_com.ibm.ws.webcontainer.WebContainer
    Rec_11_com.ibm.ws.webcontainer.WebContainer
```

```
Task is completed.
```

Figure 22-13 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 22-14 indicates that analysis information is available in the lower right pane.


Log Analyzer
File Edit View Record Windows Help
activity.log
Logs
Rec_3407_com.ibm.ws.we...
Record
Symptom
Rec_3407_com.ibm.ws.webc...
Symptom
Task is completed.
Analysis has been completed.

```
Rec_3404_com.ibm.ws.management.sync.
Rec_3405_com.ibm.ws.management.sync.
Rec_3406_com.ibm.ws.webcontainer.oseli
Rec_3407_com.ibm.ws.webcontainer.oseli
Rec_3408_com.ibm.ws.webcontainer.oseli
Rec_3409_com.ibm.ws.management.sync.
Rec_3410_com.ibm.ws.webcontainer.oseli
Rec_3411_com.ibm.ws.management.sync.
Rec_3412_com.ibm.ws.management.sync.
Rec_3413_com.ibm.ws.management.Admi
Rec_3414_com.ibm.ws.security.auth.login.(
Rec_3415_com.ibm.ws.management.sync.
Rec_3416_com.ibm.ws.management.node
Rec_3417_com.ibm.ws.management.conn
Rec_3418_com.ibm.ws.management.conn
Rec_3419_com.ibm.ws.runtime.WsS...
Rec_3420_com.ibm.ws.management.sync.
Rec_3421_com.ibm.ws.management.sync.
Rec_3422_com.ibm.ws.management.sync.
Rec_3423_com.ibm.ws.management.sync.
Rec_3424_com.ibm.ws.management.sync.
Rec_3425_com.ibm.ws.management.sync.
```

```
5.0.0 p0236.01]
ServerName:       LeeNetwork\ka0klfr\server1
TimeStamp:        10/22/2002 16:40:51.562000000
UnitOfWork:
Severity:         1
Category:         ERROR
PrimaryMessage:   PLGN0021E: Servlet Request Processor
Exception: Virtual Host/WebGroup Not Found : The web group
/hello has not been defined
ExtendedMessage:
RawData:
```

```
Matching String(s): "PLGN0021E:"
-------------------------------------------------------------
------------
This error indicates that a web request is reaching
WebSphere, but that WebSphere does not have a virtual
host corresponding to the hostname specified in the URL.
 Look at the requested host, following "the host" in the
```

Figure 22-14 Log Analyzer information

22.5.2 Main window

As shown in Figure 22-14, 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(left)

By default,the Log Analyzer's Logs pane displays log entries by unit of work(UOW). It lists all the UOW instances and its associated entries from the logs that you have opened. You may find the UOW grouping useful when you are trying to find related entries in the service or activity log or when you are diagnosing problems across multiple machines. The file name of the first log that you opened is shown in the pane's title bar. There is a root folder and under it, each UOW has a folder icon which you can expand to show all the entries for that UOW. All log entries without any UOW identification are grouped into a single folder in this tree view. 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:

```
2002-10-22 17:33:16.359000000 (3469) 1-1880:itsohost/WebbankServer01
```

Where:

- `2002-10-22 17:33:16.359000000` is the timestamp.
- `(3469)` 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_3407_com.ibm.ws.webcontainer.oselistener.OSEListenerDispatcher
```

Where:

- `Rec_3407` is the entry number.
- `com.ibm.ws.webcontainer.oselistener.OSEListenerDispatcher` is the class name.

Every log entry is assigned an entry number, `Rec_`*`nnnn`*, when a log is opened in the Log Analyzer. If more than one file is opened in the Log Analyzer (merged files), the `Rec_`*`nnnn`* 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_`*`nnnn`* 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. You may want to look at the log entry prior to this one when diagnosing problems.

WonYoung: what is a "re-raised" or "re-mapped" exception?
I deleted it, "the entry has some analysis information" is enough.
"and that it has a re-raised or re-mapped exception"

? 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(upper right)

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(lower right)

When the analyze action has been invoked, any information found in the symptoms database for the selected log entry 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.

22.5.3 Preferences

Merging logs on multiple application servers

The correlation ID can be used to correlate activity to a particular client request, or correlate activities on multiple application servers.

To merge different service or activity.log files from different machines where your transaction occurred:

1. Make sure that Enable Correlation ID box is checked as discussed in 22.3.3, “IBM service (activity) log” on page 913
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 symptom database included in the Log Analyzer package contains entries for common events and errors. New versions of the symptom database provide additional entries.

You can download the symptoms database using the Log Analyzer GUI. Select **File -> Update Database -> WebSphere Application Server Symptom Database** (for WebSphere Application Server) or **WebSphere Application Server Network Deployment Symptom Database** (for WebSphere Application Server Network Deployment) from the main menu, as shown in Figure 22-15.

Alternatively, you can download new versions of the database from the IBM FTP site. The URL for the FTP site is located in file: `<WAS_HOME>/properties/logbr/ivblogbr.properties`. Place the symptomdb.xml file in the `<WAS_HOME>/properties/logbr/symptoms/XXX` directory on your machine.

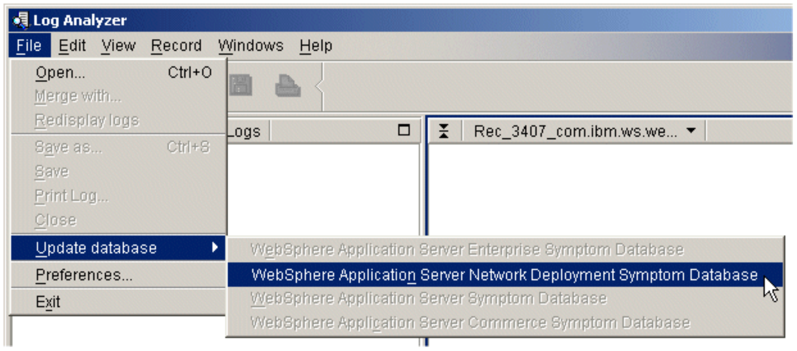

Log Analyzer
File Edit View Record Windows Help
Open... Ctrl+O
Merge with...
Redisplay logs
Save as... Ctrl+S
Save
Print Log...
Close
Update database
Preferences...
Exit
Logs
Rec_3407_com.ibm.ws.we...
WebSphere Application Server Enterprise Symptom Database
WebSphere Application Server Network Deployment Symptom Database
WebSphere Application Server Symptom Database
WebSphere Application Server Commerce Symptom Database

Figure 22-15 Updating the symptom database

If your organization uses a FTP or SOCKS proxy server, you can add a proxy definition to the Proxy Preferences page as described below:

1. Select **File** -> **Preferences** -> **Proxy**.
2. Select the appropriate proxy type.
3. Enter the host name and port number of the proxy server on the Proxy panel.

22.6 Thread Analyzer

In the past we have had the ability to determine the CPU or memory usage involved in processing applications but lacked the ability to look deeper into the application process. JVM problem determination, especially in the multi-threaded Web Application Server environment, has been difficult due to insufficient analysis tools. When performing problem determination it is often necessary to know exactly what is happening in the application server or in the application code.

In addition, application servers sometimes produce a Java stack trace, also called a Java thread dump or javacore file, as a result of errors. Tools were needed to assist in viewing this trace.

The Thread Analyzer has been included with WebSphere to assist you in analyzing Java stack trace files and in viewing the threads, both for hung or deadlock conditions, or to simply evaluate application processing. However, Thread Analyzer is simply a tool to help you view threads and stack traces. The analysis is still up to you so we will first take a look at the Java stack trace structure.

22.6.1 Understanding Java stack trace

When a JVM crashes with an internal error, for example a segmentation violation(`SIGSEGV`, signal # 11) , an illegal instruction(`SIGILL`, signal # 9) or an abort signal(`SIGABRT`, signal #6), it will call its own signal handler to dump the threads and monitors information.

If you are using the IBM JDK on the AIX, Linux or Windows platform, the Java stack trace will be dumped to the special `javacore` text file in addition to the system `core` dump file. The `javacore` filename will be in the following format:

```
javacore.<process_id>.<time>.txt
```

where `<time>` is the return value from the time subroutine at the time of the `core` dump.

The `javacore` file will be located in one of the following locations:

1. In the writable directory referred to by the environment variable `IBM_JAVACOREDIR`.
2. Otherwise, in the writable current working directory of the JVM process.
3. Otherwise, in the writable referred by the environment variable `TMPDIR`.
4. Otherwise, in the “`/tmp`” directory.

Normally, on AIX and Windows, the directory is `<WAS_HOME>` directory(not `<WAS_HOME>/bin`). Sometimes, the Java stack trace will be generated in the `<stderr>` log file or on your forground terminal console if your platform is Solaris or HP-UX.

WonYoung: normally <WAS_HOME>/bin ?
No. In previous versions of WebSphere, the directory is <WAS_HOME>/bin. But,in WebSphere V5, normally in <WAS_HOME> directory. I have checked this on Windows and AIX platform several times. Do I have to mension it among versions of WebSphere?

Generating the Java stack trace

The Java stack trace or thread dump can be generated explicitly in one of two ways:

- For Unix platforms: Send a specific signal explicitly to the JVM process, for example signal `SIGQUIT`(signal #3) or `SIGILL`(signal #4), `SIGTERM`(signal # 15), as looks like this:

  ```
  # kill -3 <unix_process_id>
  ```

In the case of sending signal #3 `SIGQUIT`, the WebSphere process does not stop running because it has it's own signal handler. The JVM will generate the Java stack trace and keep going.

> **Note:** Do not send signal #3 `SIGQUIT` frequently to the process on the heavily loaded systems because it could be temporarily blocked or, on rare occasions, crash your application server.

> **Note:** Do not use the command "`kill -9 <pid>`" which means sending a signal #9 `SIGKILL`. This command kills the process without any trace results.

For reference we have listed the signal numbers and their meanings in Appendix B, "Signal information table" on page 969.

- Or, use debugging tools or API calls, for instance the Thread Analyzer , described later in Section 22.6.2, "Running Thread Analyzer" on page 946.

If it was successful to generate a Java stack trace, it looks like this:

Example 22-7 Java stack trace

```
Fri Nov  8 10:41:53 2002
SIGQUIT received at 0x0 in <unknown>.
J2RE 1.3.1 IBM AIX build ca131-20020821
Current Thread Details
----------------------
    "Signal dispatcher" sys_thread_t:0x40BD2A28
         ----- Native Stack -----
        unavailable - iar 0x0 not in text area
-----------------------------------------------------------------------
Operating Environment
---------------------
Host                    : m10df51f.itso.ral.ibm.com:9.24.104.21
OS Level                : AIX 5.1.0.0
Processors -
        Architecture    : POWER_PC (impl: POWER_630, ver: PV_630)
        How Many        : 1
        Enabled         : 1
User Limits (in bytes except for NOFILE and NPROC) -
        RLIMIT_FSIZE    : 1073741312
        RLIMIT_DATA     : 2147483645
        RLIMIT_STACK    : 33554432
        RLIMIT_CORE     : 1073741312
        RLIMIT_NOFILE   : 32000
        NPROC(max)      : 262144
```

```
Page Space (in blocks) -
        /dev/hd6: size=131072, free=129917
Application Environment
-----------------------
Signal Handlers -
        SIGHUP          : intrDispatchMD (libhpi.a)
        ....
Environment Variables -
_=/usr/WebSphere/AppServer/java/bin/java
LANG=en_US
CONFIG_ROOT=/usr/WebSphere/AppServer/config
USER=root
WAS_HOME=/usr/WebSphere/AppServer
WAS_CELL=m10df51fNetwork
WAS_NODE=m10df51f
...
Loaded Libraries (sizes in bytes)
---------------------------------
/usr/WebSphere/AppServer/java/jre/bin/liborb.a
        filesize        : 9738
        text start      : 0xD32B9000
        text size       : 0x1849
        data start      : 0x43E4FE28
        data size       : 0x118
...
---------------------- Exception Information ---------------------------
No Exception

---------------------- System Properties -------------------------------
J2RE 1.3.1 IBM AIX build ca131-20020821
...
---------------------- XM component Dump Routine  ----------------------
Full thread dump Classic VM (J2RE 1.3.1 IBM AIX build ca131-20020821, native
threads):
    "Servlet.Engine.Transports : 3" (TID:0x32E47D10, sys_thread_t:0x44831928,
state:CW, native ID:0x
2532) prio=5
        at org.apache.jsp._deadlock._jspService(_deadlock.java(Compiled Code))
        at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(HttpJspB...)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        ....
        at com.ibm.ws.webcontainer.http.HttpConnection.handleRequest(HttpCo...)
        at com.ibm.ws.http.HttpConnection.readAndHandleRequest(HttpConnecti...)
        at com.ibm.ws.http.HttpConnection.run(HttpConnection.java(Compiled...))
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:546)
         ----- Native Stack -----
        at 0xD00549DC in _event_sleep
        at 0xD0054ECC in _event_wait
        at 0xD0060E08 in _cond_wait_local
        at 0xD006129C in _cond_wait
        at 0xD0061FC0 in pthread_cond_wait
        at 0xD3056164 in condvarWait
        at 0xD3055120 in sysMonitorWait
        at 0xD2F3A430 in lkMonitorEnter
        at 0xD3078154 in _jit_monitorEnterQuicker
        at 0xD307DB30 in JITSigSegvHandler
...
---------------------- LK component Dump Routine  ----------------------
Monitor pool info:
...
Monitor Pool Dump (flat & inflated object-monitors):
...
JVM System Monitor Dump (registered monitors):
...
Thread identifiers (as used in flat monitors):
...
Java Object Monitor Dump (flat & inflated object-monitors):
...
---------------------- END OF DUMP ------------------------------------
```

The `javacore` file can be grouped into three distinct sections, runtime environments, thread dumps, and monitors information.

- **Runtime environments**: current thread details, operating environment, application environment, loaded libraries, exception information, and system properties.
- **Thread dumps**: full thread dump.
- **Monitors information**: monitor pool info, monitor pool dump, JVM system monitor dump, thread identifiers, Java object monitor dump.

Understanding Thread status

Most important things to diagnose the Java stack trace is first to understand the thread status and the meaning. Look at the following thread entry which contains “`state:CW`”.

Example 22-8 Thread entry example

```
"Servlet.Engine.Transports : 3" (TID:0x32E47D10, sys_thread_t:0x44831928,
state:CW, native ID:0x
2532) prio=5
        at org.apache.jsp._deadlock._jspService(_deadlock.java(Compiled Code))
        at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(HttpJspB...)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        ....
```

The following are the thread status indicators and their meaning:

- **R**: Running or runnable thread.
- **CW**: Condition Wait. The thread is waiting on a condition variable. For example, `Thread.sleep()` and `Object.wait()` wait in `CW` state until notified. It is often denoted by “waiting to be notified” or “waiting on condition”.
- **MW:** Monitor Wait. The thread is waiting on a monitor lock. If more than one thread tries to enter a synchronized block concurrently, only one thread can enter. Others have to wait until the first thread release the monitor lock. In this case, the status of the other threads is `MW`. It is often denoted by “waiting to enter”, “waiting for monitor entry”, or “waiting to lock monitor”.

WonYoung: I don’t understand what you mean when you say denoted by .. where would you see this?
Ok, I added Java stack trace sample and thread entry. (“thread entry” is corrent word?)

- **S:** Suspended thread
- **MS:** Monitor Suspended. The thread suspended waiting on a monitor lock.

Monitors are data structures used for synchronization within the JVM. Each object has a monitor associated with it. A monitor can be thought of as a lock of an object. If another thread already owns the monitor associated with the lock object, the current thread waits until the object is unlocked, then tries again to gain ownership.

In most normal cases, you will see threads in `R` and `CW` status, and sometimes `MW`.

If you see a thread in `MS` status, you can suspect a JVM problem because most of the time a thread in `MW` status will appear in the `S` status when it is suspended.

Case 1 : Thread status MW (Monitor Wait)

Take a look at the code in Example 22-9.

Example 22-9 MWtest.jsp for threads in status R and MW

```
<%@ page session="false" contentType="text/html" %>
<%! public static Object lock = new Object(); %>
<html><body>MWtest.jsp for threads in status R and MW
<%
  synchronized(lock){
    for (long i=0;i<500000000L;i++);
  }
%>
```

```
</body></html>
```

In a situation where the jsp is requested continously and concurrently from different more than one clients, the state of the JVM stack could be as shown in Example 22-10.

Example 22-10 Java stack trace result of MWtest.jsp

```
...
---------------------- XM component Dump Routine  ----------------------
Full thread dump Classic VM (....):
    ......
    "Servlet.Engine.Transports : 6" (TID:0x131872B8, sys_thread_t:0x2418A9B8,
state:MW, native ID:0x844) prio=5
   at org.apache.jsp._MWtest._jspService(_MWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
    ...
    "Servlet.Engine.Transports : 4" (TID:0x13C15190, sys_thread_t:0x241876E0,
state:MW, native ID:0x9D8) prio=5
   at org.apache.jsp._MWtest._jspService(_MWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
    ...
    "Servlet.Engine.Transports : 3" (TID:0x14002138, sys_thread_t:0x2416F2D8,
state:R, native ID:0x920) prio=5
   at org.apache.jsp._MWtest._jspService(_MWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
    ...
    "Servlet.Engine.Transports : 0" (TID:0x137ACA68, sys_thread_t:0x23FC9718,
state:MW, native ID:0x948) prio=5
   at org.apache.jsp._MWtest._jspService(_MWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
    ...
---------------------- LK component Dump Routine  ----------------------
Monitor pool info:
  ...
Monitor Pool Dump (flat & inflated object-monitors):
  ...
  sys_mon_t:0x007B9408 infl_mon_t: 0x007B8F58:
    java.lang.Object@13186888/13186890: owner "Servlet.Engine.Transports : 3"
(0x2416F2D8), entry count 1
   Waiting to enter:
       "Servlet.Engine.Transports : 6" (0x2418A9B8)
       "Servlet.Engine.Transports : 4" (0x241876E0)
       "Servlet.Engine.Transports : 0" (0x23FC9718)
  ...
```

```
Java Object Monitor Dump (flat & inflated object-monitors):
    ...
    java.lang.Object@13186888/13186890
        locknflags 80001600 Monitor inflated infl_mon 0x007B8F58
---------------------- END OF DUMP -------------------------------------
```

You can see that only one thread, `Servlet.Engine.Transports:3`, is running in thread state `R`. This is because only one thread can enter the synchronized block associated with the lock object. The others are waiting to enter the block and so appear as having thread status `MW` (waiting on the monitor lock). In the `Monitor Pool Dump` section at the bottom, the owner of the monitor lock, `java.lang.Object@13186888/13186890`, is `Servlet.Engine.Transports:3`, and others are listed as the “Waiting to enter” (thread state `MW`).

If there are an abnormally large number of threads in `MW` state, this implies that you should consider the performance issues of using the synchronized code block.

WonYoung: I am not sure what the last sentence is saying .. is this implying an application design problem, or perhaps you need to add more application servers and spread out the applications among JVMs?
Yes. it is possible that the symtoms mean one of all the cases. If you think it is needed, please add the sentence.

Case 2 : Thread status CW (Condition Wait)

The code shown in Example 22-11, is used to show an example of the `CW` thread status (waiting on a condition variable).

Example 22-11 CWtest.jsp for threads in status R and CW

```
<%@ page session="false" contentType="text/html" %>
<%!
  public static Object lock = new Object();
  public static final int MAX_RUNNABLE_THREADS = 2;
  public static int running_thread_count = 0;
%>
<html><body>CWtest.jsp for threads in status R and CW
<%
  try{
    synchronized(lock){
      while( running_thread_count >= MAX_RUNNABLE_THREADS )
        try{ lock.wait(1000); }catch(Exception e){}
      ++running_thread_count;
    }
    //--------------------------------------------------
    for (long i=0;i<500000000L;i++);
    //--------------------------------------------------
  }
  finally{
    synchronized(lock){
      --running_thread_count;
      lock.notifyAll();
    }
  }
%>
</body></html>
```

This example code is similar to the previous one, but we use the `wait()` and `nofify/All()` thread APIs to control the maximum number of running threads at the same time.

The Java stack trace looks like this:

Example 22-12 Java stack trace result of CWtest.jsp

```
...
---------------------- XM component Dump Routine  ----------------------
Full thread dump Classic VM (....):
    "Servlet.Engine.Transports : 9" (TID:0x13329B60, sys_thread_t:0x2415EB78,
state:CW, native ID:0xAB8) prio=5
   at java.lang.Object.wait(Native Method)
   at org.apache.jsp._CWtest._jspService(_CWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
   ...
    "Servlet.Engine.Transports : 8" (TID:0x13329BC0, sys_thread_t:0x24129B30,
state:R, native ID:0xA40) prio=5
   at org.apache.jsp._CWtest._jspService(_CWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
   ...
    "Servlet.Engine.Transports : 2" (TID:0x13019B20, sys_thread_t:0x23FF8CC8,
state:CW, native ID:0x6A8) prio=5
   at java.lang.Object.wait(Native Method)
   at org.apache.jsp._CWtest._jspService(_CWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
   ...
    "Servlet.Engine.Transports : 1" (TID:0x13019EC8, sys_thread_t:0x2400EFC8,
state:CW, native ID:0x700) prio=5
   at java.lang.Object.wait(Native Method)
   at org.apache.jsp._CWtest._jspService(_CWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
   ...
    "Servlet.Engine.Transports : 0" (TID:0x10F043F0, sys_thread_t:0x23CDF5B8,
state:R, native ID:0x954) prio=5
   at org.apache.jsp._CWtest._jspService(_CWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
   ...
---------------------- LK component Dump Routine  ----------------------
   ...
Monitor Pool Dump (flat & inflated object-monitors):
   ...
  sys_mon_t:0x007B93E0 infl_mon_t: 0x007B8F38:
    java.lang.Object@132A9D10/132A9D18: <unowned>
   Waiting to be notified:
       "Servlet.Engine.Transports : 9" (0x2415EB78)
       "Servlet.Engine.Transports : 1" (0x2400EFC8)
       "Servlet.Engine.Transports : 2" (0x23FF8CC8)
Java Object Monitor Dump (flat & inflated object-monitors):
   ...
    java.lang.Object@132A9D10/132A9D18
        locknflags 80001500 Monitor inflated infl_mon 0x007B8F38
   ...
---------------------- END OF DUMP -------------------------------------
```

The only two threads, `Servlet.Engine.Transports:0` and `8`, are running in state `R` because the total number of runnable threads was restricted by the `MAX_RUNNABLE_THREADS` variable . The others , threads #1,2, and 9, are waiting to be notified in `CW` state at the `Object.wait()`. This information is also listed in the `Monitor Pool Dump` under the "`Waiting to be notified`" line for the monitor lock object, `java.lang.Object@132A9D10/132A9D18`.

When the threads in `CW` state are notified of a notify event, only one thread can access that object exclusively. Even when the previous thread has sent a notify event to the waiting threads, the waiting threads can't access the `synchronized` block until the notifying thread has left its `synchronized` block.

WonYoung: I dont understand the first sentence in the previous paragraph .. the word "somewhere" is throwing me off ..
The sentence, "from somewhere nofityAll()", removed

Be careful when you analyze threads in `CW` state because this can be a normal situation. All threads in a thread pool, for example, the worker-threads named

`Servlet.Engine.Transports` of the Web container, are always waiting in `CW` state to be notified of new requests, as shown in Example 22-13:

Example 22-13 WebSphere V5 Servlet engine thread is waiting for new request

```
"Servlet.Engine.Transports : 0" (TID:0x32246558, sys_thread_t:0x4482B9F8,
state:CW, native ID:0x1E2F) prio=5
        at java.lang.Object.wait(Native Method)
        at java.lang.Object.wait(Object.java(Compiled Code))
        at com.ibm.ws.util.BoundedBuffer.take(BoundedBuffer.java(CompiledCode))
        at com.ibm.ws.util.ThreadPool.getTask(ThreadPool.java(Compiled Code))
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:553)
         ----- Native Stack -----
        at 0xD00549DC in _event_sleep
        at 0xD0054ECC in _event_wait
        at 0xD0060E08 in _cond_wait_local
        at 0xD006129C in _cond_wait
        at 0xD0061FC0 in pthread_cond_wait
        at 0xD3056164 in condvarWait
        at 0xD3055120 in sysMonitorWait
        at 0xD2F39728 in lkMonitorWait
        at 0xD2EAB150 in JVM_MonitorWait
```

WonYoung: I am not sure the following sentence is needed. It doesn't seem to add anything and unless we plan to expand on that, I would say to delete it.
Ok.

“Note that the actual format of the Java stack trace of the waiting threads would be different in different version of WebSphere Application Server.“

On the other hand, let's take a case involving the JDBC connection pool. If there are lots of threads in state `CW` waiting to be notified of a free connection reference, this implies that all connections are held by other threads or some connections were not properly returned by the application to the connection pool. This is a case to pursue.

It is not easy to know which thread has to send a notify event to the waiting threads because the actual thread has gained just the control truely but for all that, it does not hold the actual monitor lock.

WonYoung: I am struggling with the previous sentence .. Are you saying that the thread that held the lock the others are waiting on has released it but not sent the notification? Partially yes. The thread that held the lock past does not held the lock now. What is the condition that threads can flow down from the first synchronized block? that is for example, `running_thread_count >= MAX_RUNNABLE_THREADS` in the above first sync block. If the first thread did not execute **“**`--running_thread_count`**”** at the last time, the others could not flow down at the first block forever. But this example is just user applicaiton condition logic story. Thread’s wait() and notify() can be understanded as the same. If some threads are at wait() and one thread has to notify to the related monitor lock. But the thread forgot to send notify-event. What happen? The others can not be awaken forever. Yap. Object.wait(time) is timebomb logic. but the “time” could be too late or the awaked thread have to still stay only in the while-loop block controled by some user applicaiton condition variable. So, the answer is that the first thread that held the lock previous has to notify to the monitor for awking the others waiting to be awaken. If possible, would you add more sentences?

This means that there is no direct relation between the suspected thread and the waiting threads. We can only assume that the actual thread is one of the running/runnable threads and even this might not be the case if the thread has entered a `CW` or `MW` state caused by another, unrelated monitor lock. The analysis process depends purely on your intuition, experience, and familiarity with the code.

Case 3 : deadlock

This last example is to illustrate deadlocked threads. If you call the JSP shown in Example 22-14 once there is no problem. But if you have multiple requests you will not get a response.

Example 22-14 Deadlocked threads

```
<%@ page session="false" contentType="text/html" %>
<%! public static Object monitor1 = new Object();
    public static Object monitor2 = new Object();
    public static boolean toggle = true;
%>
<html><body> Deadlock threads test
<%
  Object lock1 = (toggle)? monitor1:monitor2;
  toggle = !toggle;
  synchronized(lock1){
    for (long i=0;i<200000000L;i++);

    Object lock2 = (lock1==monitor1)? monitor2:monitor1;
    synchronized(lock2){
      for (long i=0;i<200000000L;i++);
    }
  }
%>
</body></html>
```

Look at the following Java stack trace in shown Example 22-15.

Example 22-15 Java stack trace of deadlocked threads

```
Full thread dump Classic VM (J2RE 1.3.1 IBM AIX build ca131-20020821, native
threads):
------------------------------------------------------------------------
    "Servlet.Engine.Transports : 3" (TID:0x32E47D10, sys_thread_t:0x44831928,
state:CW, native ID:0x
2532) prio=5
        at org.apache.jsp._deadlock._jspService(_deadlock.java(Compiled Code))
        at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(HttpJspB...)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        ...
         ----- Native Stack -----
        at 0xD00549DC in _event_sleep
        at 0xD0054ECC in _event_wait
        at 0xD0060E08 in _cond_wait_local
        at 0xD006129C in _cond_wait
        at 0xD0061FC0 in pthread_cond_wait
        at 0xD3056164 in condvarWait
        at 0xD3055120 in sysMonitorWait
        at 0xD2F3A430 in lkMonitorEnter
        at 0xD3078154 in _jit_monitorEnterQuicker
        at 0xD307DB30 in JITSigSegvHandler
------------------------------------------------------------------------
    "Servlet.Engine.Transports : 2" (TID:0x32E47D70, sys_thread_t:0x44831518,
state:CW, native ID:0x
2431) prio=5
        at org.apache.jsp._deadlock._jspService(_deadlock.java(Compiled Code))
        at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(HttpJspB...)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        ...
         ----- Native Stack -----
        at 0xD00549DC in _event_sleep
        at 0xD0054ECC in _event_wait
        at 0xD0060E08 in _cond_wait_local
        at 0xD006129C in _cond_wait
        at 0xD0061FC0 in pthread_cond_wait
        at 0xD3056164 in condvarWait
        at 0xD3055120 in sysMonitorWait
        at 0xD2F3A430 in lkMonitorEnter
        at 0xD3078154 in _jit_monitorEnterQuicker
        at 0xD307DB30 in JITSigSegvHandler
```

```
---------------------- LK component Dump Routine  ----------------------
Monitor Pool Dump (flat & inflated object-monitors):
  sys_mon_t:0x3020F558 infl_mon_t: 0x00000000:
    java.lang.Object@328781F0/328781F8: Flat locked by thread ident 0x27, entry
count 1
        Waiting to be notified:
            "Servlet.Engine.Transports : 2" (0x44831518)
  sys_mon_t:0x3020F5D8 infl_mon_t: 0x00000000:
    java.lang.Object@32878200/32878208: Flat locked by thread ident 0x26, entry
count 1
        Waiting to be notified:
            "Servlet.Engine.Transports : 3" (0x44831928)
  ...
Thread identifiers (as used in flat monitors):
    ident 0x27 "Servlet.Engine.Transports : 3" (0x44831928) ee 0x4483171C
    ident 0x26 "Servlet.Engine.Transports : 2" (0x44831518) ee 0x4483130C
  ...
```

First, there are two servlet worker threads, `Servlet.Engine.Transports : 3` and `2` in `CW` state. You can see that the identifiers of the threads are `0x27` and `0x26` respectively in the Thread identifiers section.

In the `Monitor Pool Dump` section we can see that the monitor lock objects that are locked by these threads are `Object@328781F0/328781F8` and `Object@32878200/32878208` respectively.

This is easier to look at if we put it in the following table.

Table 22-3 Threads relation table

thread name	thread ident	monitor locked by the thread	thread waiting to be notified
`Servlet.Engine.Transports : 3`	**0x27**	`Object@328781F0/328781F8`	`Servlet.Engine.Transports : 2`
`Servlet.Engine.Transports : 2`	**0x26**	`Object@32878200/32878208`	`Servlet.Engine.Transports : 3`

The key is in the last column. Each thread has to notify the other.

WonYoung: So it would be fair to say the each thread has to notify the other but the other thread is locked so it can't process (receive?) the notification?
Yes. But, actually it is not true that each thread has to notify the other. they are just waiting to be notified that means they are waiting for each monitor lock is released by some condition variable, especially, in this case it is controled by JVM not the user code notify/All() method. User code just try to gain the monitor lock, eg, sysnchronized block. Would you add sentence to my following sentence?
You mention "flat-locked". I don't think we have introduced this term.
Ok, it removed.

The two threads have their own lock monitors respectively and are waiting to be notified by the each other monitor. The two threads are at total deadlock.

Flat and Inflated monitors

It is normal for synchronized accesses to occur since only one thread at a time can access the object or region of code. Therefore, the overhead of setting up a monitor for the first thread entering the `synchronized` block is unnecessary. The first "locking thread" simply sets a flag on the object to say that the object is locked (by that thread) and proceeds onwards - this is a "`flat`" monitor. If a second thread needs to enter the `synchronized` block, then the full monitor operation takes place - the monitor is "`inflated`". We have thus postponed (and often avoided) the cost of setting up the monitor.

Threrefore, a `flat` monitor implies that only one thread is accessing the object or region of code. An `inflated` monitor implies that multiple threads are trying to gain access to the monitor.

Common system monitors

There are several JVM system monitors. Understanding the system monitors sometimes helps us to be able to understand the overall thread status in the Java virtual machine. The following table lists the JVM common registered monitors.

WonYoung: can we get descriptions for the others? I"m trying to find out, but I cann't. Is there any guy who explan the whole meaning? Probably, there are some guys in hursley lab.

Table 22-4 JVM common registered monitors

name	description
Evacuation Region lock	
Heap Promotion lock	
Integer lock access-lock	
Sleep lock	
Method trace lock	
UTF8 Cache lock	
Heap lock	Protects the Java heap during heap memory
Rewrite Code lock	Protects code when an optimization is attempted
Monitor Cache lock	Only one thread can have access to the monitor cache at a time this lock ensures the integrity of the monitor cache
JNI Pinning lock	Protects block copies of arrays to native method code
JNI Global Reference lock	Locks the global reference table which holds values that need to be explicitly freed, and will outlive the lifetime of the native method call
Class loader/loading lock	Ensures only one thread loads a class at a time
Binclass lock	Locks access to the loaded and resolved classes
Class linking lock	Protects a classes data when loading native libraries to resolve symbolic references
Monitor Registry lock	Only one thread can have access to the monitor registry at a time this lock ensures the integrity of that registry
Thread queue lock	Protects the queue of active threads
Verifier lock	
Name and type hash table lock	Protects the JVM hash tables of constants and their types
String intern lock	Locks the hashtable of defined strings that were loaded from the class constant pool
Zip lock	
Java stack lock	Protects the free stack segments list

name	description
Has finalization queue lock	Protects the lists of queue lock objects that have been garbage-collected, and deemed to need finalization. They are copied to the Finalize me queue.
Finalize me queue lock	Protects a list of objects that can be finalized at leisure

22.6.2 Running Thread Analyzer

It might be hard for you to read the text based Java stack trace file, which often contains hundreds or thousands of lines. The Thread Analyzer will assist you in reading and analyzing the trace.

Starting Thread Analyzer

To start Thread Analyzer, do the following:

```
C:\ThreadAnalyzer\bin>tagui.cmd

The WAS_HOME environment variable has not been set.
Set WAS_HOME to the base install path of WebSphere.
Example: set WAS_HOME=c:\WebSphere\AppServer

Press any key to continue . . .
C:\ThreadAnalyzer\bin>set WAS_HOME=c:\WebSphere\AppServer
C:\ThreadAnalyzer\bin>tagui.cmd
```

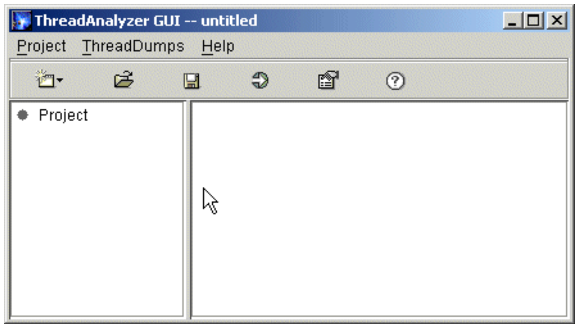


Figure 22-16 Starting Thread Analyzer

Obtaining thread dump to analyze

There are several ways to get a Java stack trace(or thread dump) into the Thread Analyzer.

From an existing Java stack trace file:

1. Select **ThreadDumps -> Open Existing File**.

2. Select and open a Java stack trace file, (normally named `javacore.???.txt`) to analyze.

From server output log file:

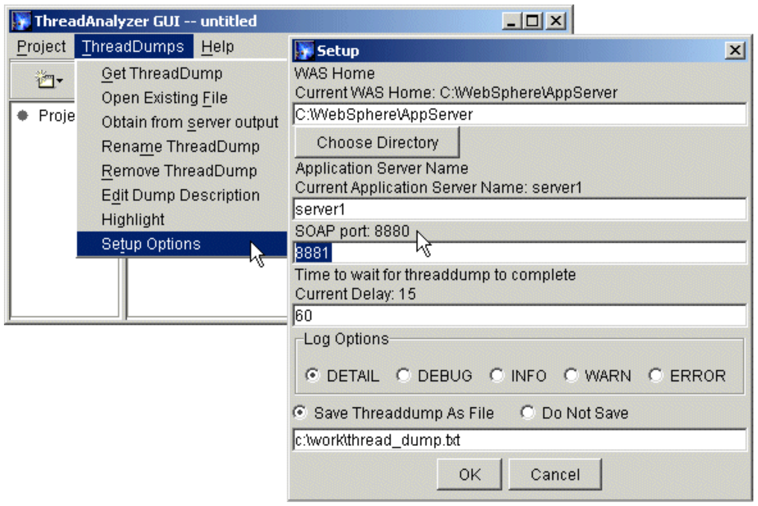
You can take a Java stack trace source from application server's log files, `<stderr>` or `<stdout>` files or any other snippets of files which contains a thread dump.

1. Select **ThreadDumps -> Obtain from server output**.
2. Click **Add files** on the new popup window to select files one or more times.
3. Then click **Process** button to extract the only Java stack trace parts from the files.

Getting ThreadDump in real time:

You can also generate a real-time thread dump while the application server is alive.

1. Before you get a thread dump from an applicaton server alive, you should set up the options by selecting **ThreadDumps -> Setup Options**, as shown in Figure 22-17.



Setup

WAS Home
Current WAS Home: C:\WebSphere\AppServer
C:\WebSphere\AppServer
Choose Directory
Application Server Name
Current Application Server Name: server1
server1
SOAP port: 8880
8881
Time to wait for threaddump to complete
Current Delay: 15
60
Log Options
[✓ DETAIL] DEBUG INFO WARN ERROR
[✓ Save Threaddump As File] Do Not Save
c:\work\thread_dump.txt
OK Cancel

Figure 22-17 Thread Analyzer - Setup Options

a. In the popup window, choose the WebSphere Application Server installation directory.

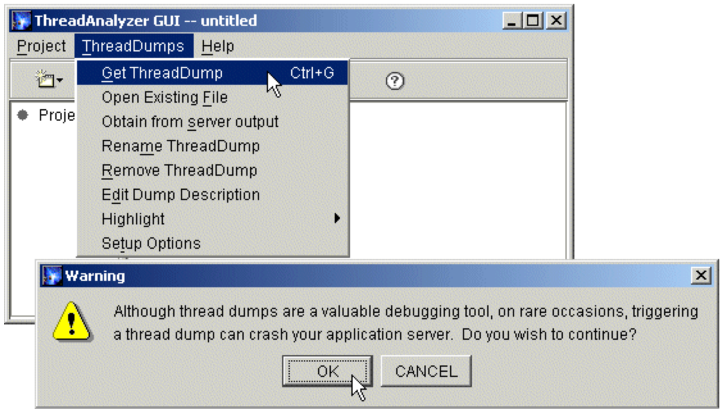
b. Type in the name of the application server you want to analyze.

c. Verify or select the SOAP port number. You should first check this port number from the administrative console of WebSphere Application Server (not from this Thread Analyzer tool) by selecting **Servers -> Application Servers** in the console navigation tree-> ***server*** -> **End Points** in Additional Properties -> `SOAP_CONNECTOR_ADDRESS`.

WonYoung: need to clarify .. do you select server>endpoints from th additional properties drop-down or is it Servers>application servers>server>endpoints>additional properties>soap connector address ...
Before:
You can check this from the administrative console by selecting **Servers -> Application Servers** in the console navigation tree, then select ***server*** -> **End Points** in Additional Properties --> `SOAP_CONNECTOR_ADDRESS`
After:
You should first check this port number from the administrative console of WebSphere Application Server (not from this Thread Analyzer tool) by selecting **Servers -> Application Servers** in the console navigation tree-> ***server*** -> **End Points** in Additional Properties --> `SOAP_CONNECTOR_ADDRESS`.

d. Change or take the default for the wait time, log options and stack trace saving options.

2. Now, you can generate a thread dump from a running application server by clicking **ThreadDumps -> Get ThreadDump,** or by simply clicking Ctl-G as shown in Figure 22-18.

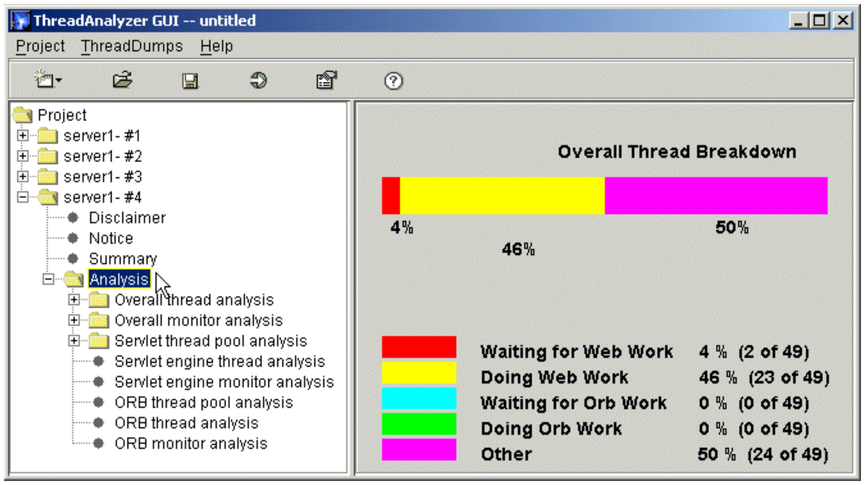


Although thread dumps are a valuable debugging tool, on rare occasions, triggering a thread dump can crash your application server. Do you wish to continue?

Figure 22-18 Thread Analyzer - Getting ThreadDump

Navigating Thread Analyzer

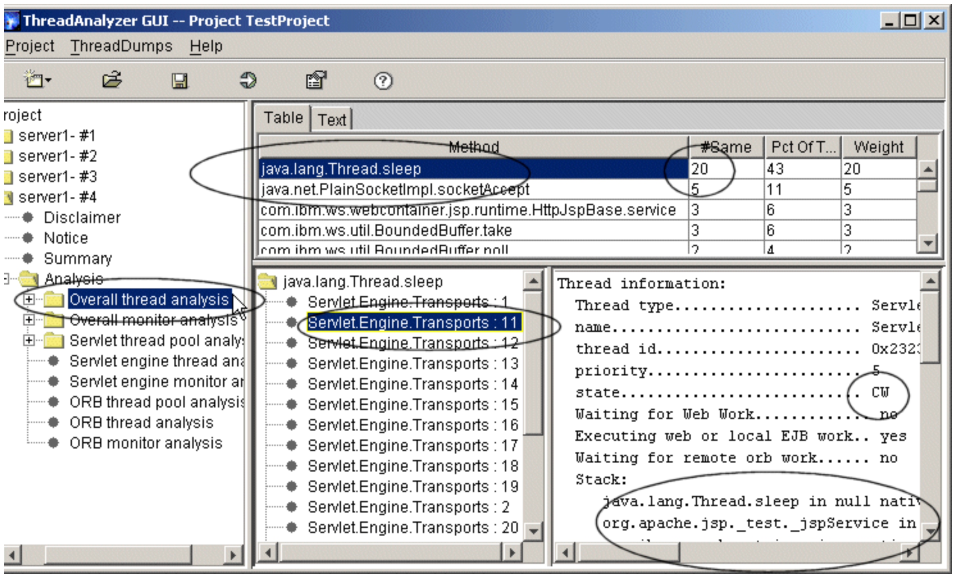
If you are successfull at generating or loading the Java stack trace or thread dump you should see something like in Figure 22-19.



🟥	Waiting for Web Work	4 % (2 of 49)
🟨	Doing Web Work	46 % (23 of 49)
🟦	Waiting for Orb Work	0 % (0 of 49)
🟩	Doing Orb Work	0 % (0 of 49)
🟪	Other	50 % (24 of 49)

Figure 22-19 Thread Analyzer - Result of ThreadDump

You can see that 23 servlet engine worker threads are doing something now and another two threads are waiting for a new request. For more detailed information about what the threads are doing, you can use the **overall thread analysis** for all threads state or **servlet thread pool analysis** for servlet threads in the left nevigation tree as shown in Figure 22-20.


ThreadAnalyzer GUI -- Project TestProject
Project ThreadDumps Help
roject
server1- #1
server1- #2
server1- #3
server1- #4
Disclaimer
Notice
Summary
Analysis
Overall thread analysis
Overall monitor analysis
Servlet thread pool analy
Servlet engine thread ana
Servlet engine monitor ar
ORB thread pool analysis
ORB thread analysis
ORB monitor analysis
Table
Text
java.lang.Thread.sleep
Servlet.Engine.Transports : 1
Servlet.Engine.Transports : 11
Servlet.Engine.Transports : 12
Servlet.Engine.Transports : 13
Servlet.Engine.Transports : 14
Servlet.Engine.Transports : 15
Servlet.Engine.Transports : 16
Servlet.Engine.Transports : 17
Servlet.Engine.Transports : 18
Servlet.Engine.Transports : 19
Servlet.Engine.Transports : 2
Servlet.Engine.Transports : 20

Method	#Same	Pct Of T...	Weight
java.lang.Thread.sleep	20	43	20
java.net.PlainSocketImpl.socketAccept	5	11	5
com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service	3	6	3
com.ibm.ws.util.BoundedBuffer.take	3	6	3
com.ibm.ws.util.BoundedBuffer.poll	2	4	2

```
Thread information:
  Thread type..................... Servle
  name............................ Servle
  thread id....................... 0x2323
  priority........................ 5
  state........................... CW
  Waiting for Web Work............ no
  Executing web or local EJB work.. yes
  Waiting for remote orb work...... no
  Stack:
     java.lang.Thread.sleep in null nati
     org.apache.jsp._test._jspService in
```

Figure 22-20 Thread Aanlyzer Overall thread analysis

22.7 Collector tool

The Collector tool gathers information about your WebSphere Application Server installation and packages it in an output `jar` file. The file can be sent to IBM Customer Support to assist in problem determination and analysis. The information in the file includes logs, property files, configuration files, operating system and Java data, and prerequisite software presence and levels.

Running the Collector Tool

The Collector tool should be run under the **root** or **Administrator** user ID because some of the commands to be executed require system access authority. (However, if you proceed without Administrator authority, much of what the Collector does will work just fine.)

- For Windows systems, log on to the system as **Administrator** or another user with Administrator authority and enter the following:

```
C:\> mkdir work
C:\> cd work
C:\work> c:\WebSphre\AppServer\bin\collector.bat
```

- For Unix systems log on to the system as **root** and do the following:

```
itsosvr:/home/# id
root(....)
itsosvr:/home/# mkdir work
itsosvr:/home/# cd work
itsosvr:/home/work# /usr/WebSphre/AppServer/bin/collector.sh
```

You cannot run the Collector tool in a directory under the WebSphere Application Server installation directory so it is recommended that you fully qualify the PATH to the Collector tool.

Results

The Collector program creates a log file, `Collector.log`, and an output `.jar` file in the current work directory. The `.jar` file name is based on the hostname and package of the server on which the Collector tool was run, in the format: `hostname-ND|Base-WASenv.jar`.

What to do next

Send the `hostname-ND|Base-WASenv.jar` file to IBM Customer Support for analysis.

22.8 First Failure Data Capture logs

The First Failure Data Capture (FFDC) function preserves the information generated from a processing failure and returns control to the affected engines. There are three property files which control the behavior of the FFDC filter:

- `properties/ffdcStart.properties` - used while the server is starting
- `properties/ffdcRun.properties` - used after the server is ready
- `properties/ffdcStop.properties` - used while the server is in the process of stopping.

The captured data is saved automatically in the `<WAS_HOME>/logs/ffdc` directory for use in analyzing the problem, and could be collected by the Collector tool.

The First Failure Data Capture tool is intended primarily for use by IBM Service. It runs as part of the IBM WebSphere Application Server, and you cannot start or stop it. It is recommended that you not attempt to configure the First Failure Data Capture tool. If you experience conditions requiring you to contact IBM Service, your IBM Service representative will assist you in reading and analyzing the First Failure Data Capture log.

22.9 DumpNameSpaces

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 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 -?
```

The generated output will look like Example 24-6, which is the *short* dump format.

```
   dumpNameSpace -host localhost -report short
```

Example 22-16 dumpNameSpace output

```
Getting the initial context
Getting the starting context
==============================================================================
Name Space Dump
   Provider URL: corbaloc:iiop:localhost:2809
   Context factory: com.ibm.websphere.naming.WsnInitialContextFactory
   Requested root context: cell
   Starting context: (top)=ItsoNetwork
   Formatting rules: jndi
   Time of dump: Tue Oct 22 15:57:37 EDT 2002
==============================================================================

==============================================================================
Beginning of Name Space Dump
==============================================================================

    1 (top)
    2 (top)/cell                                        javax.naming.Context
    2    Linked to context: ItsoNetwork
    3 (top)/legacyRoot                                  javax.naming.Context
    3    Linked to context: ItsoNetwork/persistent
    4 (top)/cells                                       javax.naming.Context
    5 (top)/deploymentManager                           javax.naming.Context
    5    Linked to URL: corbaloc::ka0klfr:9809/NameServiceServerRoot
    6 (top)/persistent                                  javax.naming.Context
    7 (top)/persistent/cell                             javax.naming.Context
    7    Linked to context: ItsoNetwork
...
   30 (top)/nodes/ka0klfrManager/node                   javax.naming.Context
   30    Linked to context: ItsoNetwork/nodes/ka0klfrManager
   31 (top)/cellname                                    java.lang.String
   32 (top)/clusters                                    javax.naming.Context
==============================================================================
End of Name Space Dump
==============================================================================
```

22.10 HTTP session monitoring

In the case of session related problems, it is sometimes necessary to see all HTTP session information when you have some kind of session related problems. But, it is not generally supported because of some security issues or several reasons.

Carla .. : Editing stopped here ..

22.10.1 HTTP session tracker servlet

WebSphere Application Server V5 introduces the HTTP session tracker servlet for the demand, called `IBMTrackerDebug` servlet. Open your brower and check it.

```
http://localhost:9080/servlet/com.ibm.ws.webcontainer.httpsession.IBMTrackerDeb
ug
```

Example 22-17 `Result of IBMTrackerDebug servlet`

```
J2EE NAME(AppName#WebModuleName):: DefaultApplication#DefaultWebApplication.war
cloneId : -1

Number of sessions in memory: (for this webapp) : 11
use overflow : true
overflow size (for this webapp) :
Invalidation alarm poll interval (for this webapp) : 304
Max invalidation timeout (for this webapp) : 1800
Using Cookies : true
Using URL Rewriting : false
use SSLId : false
URL Protocol Switch Rewriting : false
Session Cookie Name : JSESSIONID
Session Cookie Comment : SessionManagement
Session Cookie Domain : null
Session Cookie Path : /
Session Cookie MaxAge : -1
Session Cookie Secure : false
Maximum in memory table size : 1000
current time : Wed Oct 23 18:18:37 EDT 2002
integrateWASSec :false
Session locking : false
Session locking timeout: 5
Allow access on lock timeout:true
Sessions Created:11
Active Count:0
Session Access Count:8
Invalidated Sessions Count:0
Invalidated By SessionManager:0
Garbage Collected count:0
```

```
SessionAffinity Breaks:0
Number of times invalidation alarm has run:0
Rejected Session creation requests(overflow off):0
Cache Discards:0
Attempts to access non-existent sessions:2
Number of binary reads from external store:0
Total time spent in reading from external store(ms):0
Total number of bytes read:0
Number of binary writes to external store:0
Total time spent in writing to external store(ms):0
Total number of bytes wriiten out:0
Total size of serializable objects in memory :1859
Total number objects in memory :11
Min size session object size:169
Max size session object size :169
```

22.10.2 HTTP sesison data full dump

If the IBMTrackerDebug servlet does not give you enough information to solve your HTTP session related problems, you can make your own monitoring servlet to view or dump all the HTTP session data in memory.

For more detailed information, see Appendix A, “HTTP session data dump” on page 965.

22.11 Useful UNIX commands

22.11.1 Looking at system core files

If possible, UNIX process(included JVM process) will produce *system core dump* as well as Java stack trace in the process's working directory when it crash. The system core dump can provide useful information as to why the process crashed. The system `core` dump provides a system view of a failing JVM process.

But, they do not understand Java classes. Everything in a system core dump is C library oriented. The system core dump information provided for JVM process refers to Java's C libraries and not the reference Java class files.

On UNIX systems, they usually appear as `core` files. On Window's systems they appear as `drwtsn32.log` files.

System core dump examples

The following programs shown in Example 22-18,Example 22-19, and Example 22-20, are used to show an example of system core dump caused by bad JNI(Java Native Interface) user applications.

Example 22-18 badjni.jsp

```
<%@ page session="false" contentType="text/html" %>
<html><body>Generating system core dump
<%
  itso.BadJni badJni = new itso.BadJni();
  badJni.badJniMethod();
%>
</body></html>
```

Example 22-19 itso.BadJni.java

```
package itso;
public class BadJni
{
    public native void badJniMethod();
    static {
        System.loadLibrary("badjni");
    }
}
```

Example 22-20 itso_BadJni.c

```
#include <jni.h>
#include "itso_BadJni.h"
#include <stdio.h>
void bad_native_method(void)
{
    char c[1];
    char *p = NULL;  /* null pointer */
    strncpy(p,c,10); /* Segmentation fault */
}
JNIEXPORT void JNICALL Java_itso_BadJni_badJniMethod
  (JNIEnv *env, jobject obj)
{
    bad_native_method();
}
```

You can realize that we are trying to copy strings at the NULL pointer variable in the method `bad_native_method`,in Example 22-20. This kind of programs can crash the JVM process and then, a system core dump and Java stack trace should be genetrated in the process's working directory, normally `<WAS_HOME>` directory, as shown in Example 22-21.

Example 22-21 System core dump and Java stack trace

```
# pwd
/usr/WebSphere/AppServer

# ls -alF *core*
-rw-r--r--   1 root system 425440811 Nov 07 19:40 core
-rw-r--r--   1 root system     60986 Nov 07 19:40 javacore22226.1036716049.txt
```

First things what we should look at is to view the Java stack trace, `javacore` file, as we mentioned in 22.6.1, “Understanding Java stack trace” on page 931. The `javacore` file provides a Java class view of a failing JVM process as shown in Example 22-22.

Example 22-22 Java stack trace

```
Thu Nov  7 19:40:49 2002
SIGSEGV received at 0xd32a31b0 in /usr/lib/libbadjni.so. Processing terminated.
J2RE 1.3.1 IBM AIX build ca131-20020821
Current Thread Details
----------------------
    "Servlet.Engine.Transports : 9" sys_thread_t:0x44B20058
         ----- Native Stack -----
        unavailable - iar 0x4574C220 not in text area
-------------------------------------------------------------------------
...
---------------------- XM component Dump Routine  ----------------------
Full thread dump Classic VM (J2RE 1.3.1 IBM AIX build ca131-20020821, native
threads):
    "Servlet.Engine.Transports : 9" (TID:0x3210CCA0, sys_thread_t:0x44B20058,
state:R, native ID:0x2A2F) prio=5
        at itso.BadJni.badJniMethod(Native Method)
        at org.apache.jsp._badjni._jspService(_badjni.java:69)
        at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(HttpJspB...)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        ...
        at com.ibm.ws.http.HttpConnection.run(HttpConnection.java(...))
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:546)
         ----- Native Stack -----
        unavailable - iar 0x4574C220 not in text area
...
```

At best, the `javacore` file provides a clue to where the JVM process crashed, but only traces pure Java methods. In this example the javacore shows the `itso.BadJni.badJniMethod()` method of the `itso.BadJni.java` file to be the problem.

System core dump and dbx

The `core` file on UNIX systems can be interrogated using the `dbx` and `gdb` tools. The `dbx` is a debugging 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.

You can issue the `dbx` command with the Java binary executable file, normally `<WAS_HOME>/java/bin/java`, as the parameter. The commands listed in Example 22-23 show how to find the binary executable and invoke the `dbx` command.

Example 22-23 Invoking dbx

```
# pwd
/usr/WebSphere/AppServer
# ls -alF core
-rw-r--r--   1 root system 425440811 Nov 07 19:40 core
# dbx /usr/WebSphere/AppServer/java/bin/java
Type 'help' for help.
reading symbolic information ...warning: no source compiled with -g
[using memory image in core]
Segmentation fault in strncpy.strncpy [/usr/lib/libbadjni.so] at 0xd32a3318
0xd32a3318 (strncpy+0x118) 9cc50001       stbu   r6,0x1(r5)
(dbx)
```

It shows that a segmentation fault happened (that is `SIGSEGV`, signal # 11). The sample was taken from an AIX 4.3.3 and AIX 5.1 system.

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 | grep COMMAND_LINE
```

Or,

```
# strings core | more
```

After you start `dbx`, the "`where`" command provides a stack trace of where the error occurred, as shown in Example 22-24.

Example 22-24 dbx where command

```
(dbx) where
strncpy.strncpy() at 0xd32a3318
bad_native_method() at 0xd32a315c
Java_itso_BadJni_badJniMethod(0x44b1fe4c, 0x4574c2e0) at 0xd32a31ac
mmisInvoke_V_VHelper(0x32c13ef0, 0x44b235fc, 0x1, 0x44b1fe4c, 0x4574c358) at
0xd2eb6fe4
```

```
mmipInvoke_V_V(??, ??) at 0xd2ed5e6c
```

The culprit is to be found in the `bad_native_method()` method of the native JNI library module. Furthermore, we can realize that the error occured exactly when the `strncpy` function was executing. Look at the native JNI source code again in Example 22-20.

And "`registers`" and "`listi`" commands are also useful to view the system registers information and the instructions.

Example 22-25 Analyzing system core dump with dbx

```
(dbx) unset $noflregs
(dbx) registers
  $r0:0x00000000  $stkp:0x4574c1c8   $toc:0x494a13c4    $r3:0x00000000
  $r4:0x4574c20b    $r5:0xffffffff    $r6:0x00000045    $r7:0x00000074
  $r8:0x000000c2    $r9:0x00000058   $r10:0x40b55a74   $r11:0x000034e0
 $r12:0x00000000   $r13:0x00000000   $r14:0x456cf800   $r15:0x4574c350
 $r16:0x44b1fe4c   $r17:0x40f0b1ec   $r18:0x00000009   $r19:0x00000000
 $r20:0x00000041   $r21:0x5604a124   $r22:0x000000c1   $r23:0x00000001
 $r24:0x32451780   $r25:0x40b20600   $r26:0x44b235fc   $r27:0x30213b08
 $r28:0x3021c118   $r29:0x44b222b4   $r30:0x44b1fe4c   $r31:0x30212418
 $iar:0xd32a3318   $msr:0x0000d0b2    $cr:0x44824844 $link:0xd32a3160
 $ctr:0x0000000a   $xer:0x00000000
          Condition status = 0:g 1:g 2:l 3:e 4:g 5:l 6:g 7:g
  $fr0:0x0000000000000000   $fr1:0x3fe8000000000000   $fr2: 0x0000000000000000
  $fr3:0x0000000000000000   $fr4:0x0000000000000000   $fr5: 0x0000000000000000
  $fr6:0x0000000000000000   $fr7:0x0000000000000000   $fr8: 0x0000000000000000
  $fr9:0x0000000000000000  $fr10:0x0000000000000000  $fr11: 0x0000000000000000
 $fr12:0x0000000000000000  $fr13:0x3fe8000000000000  $fr14: 0x0000000000000000
 $fr15:0x0000000000000000  $fr16:0x0000000000000000  $fr17: 0x0000000000000000
 $fr18:0x0000000000000000  $fr19:0x0000000000000000  $fr20: 0x0000000000000000
 $fr21:0x0000000000000000  $fr22:0x0000000000000000  $fr23: 0x0000000000000000
 $fr24:0x0000000000000000  $fr25:0x0000000000000000  $fr26: 0x0000000000000000
 $fr27:0x0000000000000000  $fr28:0x0000000000000000  $fr29: 0x0000000000000000
 $fr30:0x0000000000000000  $fr31:0x0000000000000000 $fpscr: 0xa6100000
in strncpy.strncpy [/usr/lib/libbadjni.so] at 0xd32a3318
0xd32a3318 (strncpy+0x118) 9cc50001       stbu   r6,0x1(r5)

(dbx) listi .-40,.+40
0xd32a32f4 (strncpy+0xf4) 4182009c       beq   0xd32a3390 (strncpy+0x190)
0xd32a32f8 (strncpy+0xf8) 8cc40001      lbzu   r6,0x1(r4)
0xd32a32fc (strncpy+0xfc) 8ce40001      lbzu   r7,0x1(r4)
0xd32a3300 (strncpy+0x100) 8d040001      lbzu   r8,0x1(r4)
0xd32a3304 (strncpy+0x104) 8d240001      lbzu   r9,0x1(r4)
0xd32a3308 (strncpy+0x108) 2c060000      cmpi   cr0,0x0,r6,0x0
0xd32a330c (strncpy+0x10c) 2c870000      cmpi   cr1,0x0,r7,0x0
0xd32a3310 (strncpy+0x110) 2f080000      cmpi   cr6,0x0,r8,0x0
```

```
0xd32a3314 (strncpy+0x114) 2f890000       cmpi   cr7,0x0,r9,0x0
0xd32a3318 (strncpy+0x118) 9cc50001       stbu   r6,0x1(r5)
0xd32a331c (strncpy+0x11c) 4e400020    bdzgelr
0xd32a3320 (strncpy+0x120) 4182004c        beq   0xd32a336c (strncpy+0x16c)
0xd32a3324 (strncpy+0x124) 9ce50001       stbu   r7,0x1(r5)
0xd32a3328 (strncpy+0x128) 4e400020    bdzgelr
0xd32a332c (strncpy+0x12c) 4186004c       beq   cr1,0xd32a3378 (strncpy+0x178)
0xd32a3330 (strncpy+0x130) 9d050001       stbu   r8,0x1(r5)
0xd32a3334 (strncpy+0x134) 4e400020    bdzgelr
0xd32a3338 (strncpy+0x138) 419a004c       beq   cr6,0xd32a3384 (strncpy+0x184)
0xd32a333c (strncpy+0x13c) 9d250001       stbu   r9,0x1(r5)
0xd32a3340 (strncpy+0x140) 4e400020    bdzgelr
0xd32a3344 (strncpy+0x144) 419e004c       beq   cr7,0xd32a3390 (strncpy+0x190)
(dbx) quit
```

More usefull commands of dbx are:

```
map,thread, thread info, thread <thread_no>, thread current <thread_no>
```

Type “`help`” for help on a command or topic and “`quit`” to exit dbx.

To make sure that a good core file gets generated

Sometimes after a crash there is no core file or the output from dbx shows that the core file is truncated. Ensure that:

1. The file system containing the core file has enough free space.

   ```
   # df -k
   ```

   It is recommaned over than 500 MB, but depends on your WebSphere configuration environment.

2. On AIX, ensure that “Enable full CORE dump” switch is set to true in the system environment.

   ```
   # smitty chgsys
   ```

   and change Enable full CORE dump to true. Alternatively, you can use the following command,

   ```
   # lsattr -El sys0 | grep fullcore
   fullcore     false         Enable full CORE dump  True
   # chdev -a fullcore=true -l sys0
   sys0 changed
   # lsattr -El sys0 | grep fullcore
   fullcore     true          Enable full CORE dump  True
   ```

3. The owner of the running process have write permission to the directory the process dumps the core to.

4. Both of the maximum file and coredump size specification are enough.

```
# ulimit -a
time(seconds)        unlimited
file(blocks)         2097151 <-- !
data(kbytes)         131072
stack(kbytes)        32768
memory(kbytes)       32768
coredump(blocks)     2097151 <-- !
nofiles(descriptors) 2000
```

You can change the file and coredump maximum value by the following command,

```
# ulimit -f nnnnnnn
# ulimit -c nnnnnnn
```

The directory the process dumps the core to is the current working directory of the process running. In the WebSphere 5.0, it is `<WAS_HOME>` directory normally.

Monitoring a running process with dbx

Another use of the `dbx` command is to monitor a running process. The `-a` parameter allows the debug program to be attached to a process that is running. To attach the debug program, you need authority to use the `kill` command on this process. Use the `ps` command to determine the process ID. If you have permission, the `dbx` program interrupts the process, determines the full name of the object file, reads in the symbolic information, and prompts for commands.

Example 22-26 Monitoring a running process with dbx

```
# ps -ef|grep java
  root 18088     1 0 Nov 07  pts/2  9:06 /usr/WebSphere/... dmgr
  root 20648     1 6 Nov 07  pts/2 39:58 /usr/WebSphere/... m10df51f
  root 22234 20648 2 Nov 08  pts/2 32:19 /usr/WebSphere/... server1

# dbx -a 22234
Waiting to attach to process 22234 ...
Successfully attached to java.
Type 'help' for help.
reading symbolic information ...warning: no source compiled with -g

stopped in _event_sleep at 0xd00549dc
0xd00549dc (_event_sleep+0x90) 80410014        lwz   r2,0x14(r1)
(dbx) where
_event_sleep(??, ??, ??, ??, ??) at 0xd00549dc
_event_wait(??) at 0xd0054ec8
_cond_wait_local(??, ??, ??) at 0xd0060e04
_cond_wait(??, ??, ??) at 0xd0061298
pthread_cond_wait(??, ??) at 0xd0061fbc
condvarWait(??, ??, ??) at 0xd3056160
```

```
sysMonitorWait(??, ??, ??, ??) at 0xd305511c
lkMonitorWait(??, ??, ??, ??) at 0xd2f39724
JVM_MonitorWait(??, ??, ??, ??) at 0xd2eab14c
(dbx) detach
#
```

To continue execution of the application and exit dbx, enter “`detach`**”** instead of **“**`quit`**”**.(If you entered “`quit`” to exit, the process would stop running.)

Note: Do not use the command “`dbx -a <pid>`” on the heavily loaded systems because it could be temporarily blocked.

Note: In 22.6.1, “Understanding Java stack trace” on page 931, we explained how to dump Java stack traces using Thread Analyzer or the `kill` command. If a process appears to hang, it is probably a good idea to view both a Java stack trace and a system thread dump:

1. To generate the Java stack trace (generates a javacore file):

   ```
   #kill -3 <java_pid>
   ```

2. To view the system thread dump:

   ```
   #dbx -a <pid>
   ```

errpt command

On AIX, the `errpt` command generates an error report from entries in a system error log. If you have got a system core dump, the event would be logged in the system error log. You can check it as following.

Example 22-27 errpt command

```
# errpt -a > /tmp/errpt.txt
# vi /tmp/errpt.txt
....
---------------------------------------------------------------------------
LABEL:          CORE_DUMP
IDENTIFIER:     C60BB505

Date/Time:       Thu Nov  7 19:40:49 EST
...
Detail Data
SIGNAL NUMBER
          11
USER’S PROCESS ID:
```

```
       22226
...
PROGRAM NAME
java
ADDITIONAL INFORMATION
strncpy 118
bad_nativ 20
Java_itso 18
mmisInvok 2A4
entryCmp FFFFE688
??
...
SYMPTOM CODE
PCSS/SPI2 FLDS/java SIG/11 FLDS/strncpy VALU/118 FLDS/bad_nativ
---------------------------------------------------------------------------
...
```

A little piece of names of the native methods processing at the time of the JVM crash is shown in the error report. Even the `errpt` command did not give us fully qualified information, but it would be enough to start tracing the problem.

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.

22.12 Application debugging and tracing

22.12.1 Debugging with Application Server Toolkit

The Application Server Toolkit, included with the WebSphere Application Server on a separately-installable CD, includes debugging functionality that is built on the Eclipse workbench and that includes the following:

- The WebSphere Application Server debug adapter

  which allows you to debug Web objects that are running on WebSphere Application Server and that you have launched in a browser. These objects include EJBs, JSPs, and servlets.

- The JavaScript debug adapter

which enables server-side JavaScript debugging.

- The Compiled language debugger

  which allows you to detect and diagnose errors in compiled-language applications.

- The Java development tools (JDT) debugger

  which allows you to debug Java.

All of the debug components in the Application Server Toolkit can be used for debugging locally and for remote debugging.

To learn more about the debug components, launch the Application Server Toolkit, select Help > Help Contents and choose the Debugger Guide bookshelf entry.

Debugging WebSphere Application Server applications

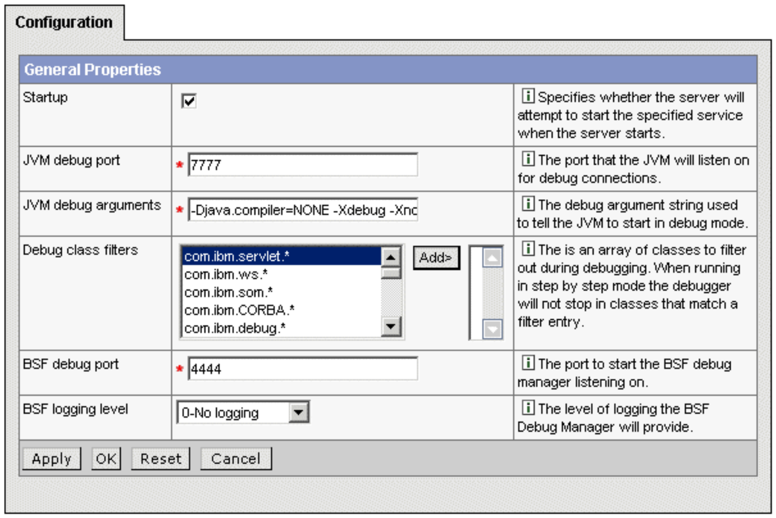
In order to debug your application, you must first create a Java project or a project with a Java nature. You must then import the program that you want to debug into the project. By following the steps below, you can import the WebSphere Application Server examples into a Java project.

There are two debugging styles available. Step-by-step mode will prompt you whenever the server calls a method on a Web object. A dialog allows you to either step into the method or skip it. In the dialog, you can also turn off step-by-step mode. Alternatively, if you know which part of your program you want to debug, you can add breakpoints to this code and run until one of the breakpoints is encountered. Note that breakpoints work with both styles of debugging - step-by-step mode just allows you to see which Web objects are being called without having to set up breakpoints ahead of time.

You do not have to import all of your program into the project. If you do not import all of your program into the project, some of the source may not compile. You can still debug the project and most features of the debugger will work including breakpoints, stepping and viewing/modifying variables. However, the inspect and display features in the source view will not work if the source has build errors. The inspect and display features allow you to select an expression in the source view and evaluate it. You must import any source that you want to set breakpoints in.

Configuring for debugging Service

To view the debugging service configuration on the administrative console, click **Servers** -> **Application Servers** -> ***server*** -> **Debugging Service** as shown in Figure 22-21.



Startup	✓	Specifies whether the server will attempt to start the specified service when the server starts.
JVM debug port	* 7777	The port that the JVM will listen on for debug connections.
JVM debug arguments	* -Djava.compiler=NONE -Xdebug -Xnc	The debug argument string used to tell the JVM to start in debug mode.
Debug class filters	com.ibm.servlet.* com.ibm.ws.* com.ibm.som.* com.ibm.CORBA.* com.ibm.debug.*	The is an array of classes to filter out during debugging. When running in step by step mode the debugger will not stop in classes that match a filter entry.
BSF debug port	* 4444	The port to start the BSF debug manager listening on.
BSF logging level	0-No logging	The level of logging the BSF Debug Manager will provide.

Figure 22-21 Debugging service configuration

- **Startup** specifies whether the server will attempt to start the Debug service when the server starts.
- **JVM debug port** specifies the port that the Java Virtual Machine will listen on for debug connections.
- **JVM debug arguments** specifies the debugging argument string used to start the JVM in debug mode.
- **Debug class filters** specifies an array of classes to ignore during debugging. When running in step-by-step mode, the debugger whill not stop in classes that match a filter entry.
- **BSF debug port** specifies the port that the BSF Debug Manager listens on.
- **BSF logging level** specifies the level of logging provided by the BSF Debug Manager. The valid range is 0-3, with 3 being the highest level of logging.

Author Comment: This section, “Debugging with Applicaiton Server Toolkit” , is just cut and pasted from the InforCenter except the captured image. Would you change the sentence? Actually, I don’t have any idea.

22.12.2 JRas framework

Author Comment: Brief introduction of IBM JRas framework is needed here.
1. Why the Logging framework is needed since from you appliation design stage
2. What the JRas framework is: one of logging and messaging framework developed by IBM.
All information about JRas framwork exists in the InforCenter and previous V4 handbook. But, I think I'm not proper guy to write down in ENGLISH.
Would you mind if ...? Just cuting and pasting from V4 handbook would be enough.

22.13 Other resources

Infocenter

Other redbooks

chek e-fix

web sites

contack IBM

Appendix A.

HTTP session data dump

Compile the following `SessionContextMonitor.java` file.

Example 22-28 SessoinContextMonitor.java

```
package com.ibm.ws.webcontainer.httpsession;
import java.util.*;
public class SessionContextMonitor
{
    public SessionContextMonitor(){}
    public static Enumeration getScrSessionContexts(){
        return SessionContextRegistry.getScrSessionContexts();
    }
    public static Enumeration tableKeys(SessionContext sc){
        return sc.tableKeys();
    }
    public static Object tableGet(SessionContext sc, Object key) {
        return sc.tableGet(key);
    }
}
```

Comipling steps:

- For Windows systems,

```
C:\> set WAS_HOME=C:\WebSphere\AppServer
C:\> set JAVA_HOME=%WAS_HOME%\java
C:\> set PATH=%JAVA_HOME%\bin;%PATH%

C:\> mkdir work
C:\> cd work
C:\work> javac -classpath %WAS_HOME%\lib\httpsession.jar;%WAS_HOME%\lib\web
container.jar -d .   c:\downloads\SessoinContextMonitor.java
c:\work> jar -cvf httpsessionmon5.jar com
...
c:\work> copy httpsessionmon5.jar %WAS_HOME%\lib\
```

For Unix systems,

```
# ksh
$ export WAS_HOME=/usr/WebSphere/AppServer
$ export JAVA_HOME=$WAS_HOME/java
$ export PATH=$JAVA_HOME/bin:$PATH

$ md /home/work
$ cd /home/work
/home/work/$ javac -classpath $WAS_HOME/lib/httpsession.jar:$WAS_HOME/lib/
webcontainer.jar -d .   /tmp/SessoinContextMonitor.java
/home/work/$ jar -cvf httpsessionmon5.jar com
...
/home/work/$ cp httpsessionmon5.jar $WAS_HOME/lib/
```

Now, restart the WebSphere Application Server if already started, and put the following `sessoinmon5.jsp` file into the proper directory you can access via a web browser.

Example 22-29 sessionmon5.jsp

```
<%@ page session="false" buffer="none" contentType="text/html" %>
<%@ page import="java.io.*,java.util.*,com.ibm.ws.webcontainer.httpsession.*"
%>
<html><head><title>IBM WebSphere 5.x Session Monitor</title></head>
<body>
<table width=600><tr><td>
<center><h3>IBM WebSphere 5.x Session Monitor</h3></center>
</td></tr></table>
<% String ip =java.net.InetAddress.getLocalHost().getHostAddress();
   java.text.SimpleDateFormat df =
      new java.text.SimpleDateFormat("yyyy.MM.dd HH:mm:ss");
   String date = df.format(new java.util.Date());
%>
OS : <%= System.getProperty("os.name")+System.getProperty("os.version")%><br>
IP Address : <%= ip %><br>
Date : <%= date %><br><br>
JDK fullversion : <%= System.getProperty("java.fullversion") %><br>
user.language : <%= System.getProperty("user.language") %><br>
user.regiong : <%= System.getProperty("user.region") %><br>
user.timezone : <%= System.getProperty("user.timezone") %><br>
file.encoding : <%= System.getProperty("file.encoding") %><br>
<%
try{
   Enumeration sessionContexts =
        SessionContextMonitor.getScrSessionContexts();
   while(sessionContexts.hasMoreElements()) {
       SessionContext sc = (SessionContext)sessionContexts.nextElement();
%>
<hr>
<table width=600><tr><td>
<%= sc.toHTML() %></UL>
</td></tr></table>
<hr>
<table width=600><tr><td>
<center><h3>Session Dump Internals</h3></center>
</td></tr></table>
<pre>
<%
    int session_count = 0;
    Enumeration enum = SessionContextMonitor.tableKeys(sc);
    while(enum.hasMoreElements()){
        try {
            String key = (String)enum.nextElement();
            SessionData data = (SessionData)
                SessionContextMonitor.tableGet(sc,key);
            session_count ++;
            out.println(
                "<b>" + (session_count) +
                "</b>   " + data.getId());
            out.println("create time : " +
             df.format(new java.util.Date(data.getCreationTime())));
            long l = data.getLastAccessedTime();
            if ( l != -1L )
                out.println("last access : " +
                    df.format(new java.util.Date(l)));
            else
              out.println("last access : " +
                df.format(new java.util.Date(data.getCreationTime())));
            out.println("max inactive interval : " +
                data.getMaxInactiveInterval());
            out.println("user name : " + data.getUserName());
            out.println("valid session : " + data.isValid());
            out.print("new session : ");
            try { out.println(data.isNew()); }
            catch(Exception e){out.println(e.toString());}
            out.println("overflowed : " + data.isOverflow());

            //Cookie cookie = data.getCookie();
            //if ( cookie != null )
            //    out.println("cookie : " + cookie.toString());

            out.print("data : {");
            Enumeration s_keys = data.getNames();
            for(boolean first = true; s_keys.hasMoreElements(); ){
                if ( first )  first = false;
                else out.print(',');
                try {
                    String k = (String)s_keys.nextElement();
                    Object value = data.getValue(k);
                    out.print(k + "=" + value.toString());
                }catch(Exception e){out.println(e.toString());}
            }
            out.println('}');
            out.println();

            out.println("--------------------");
            //out.println(data);
        }catch(Exception e){ out.println(e.toString()); }
    }
    out.println("Total Http Session Count : " + session_count + "</b></pre>");
   } // end of while for each WebModules
}
catch(Exception e){
   out.println(e.toString() + "</pre>");
}
%>
<BR><hr>NOTE: This is only for debugging<br></body></html>
```

Author Comment: The above program source would be changed.

Appendix B.

Signal information table

Table 22-5 Signal table

Signal #	name	comment
1	SIGHUP	Hangup (POSIX)
2	SIGINT	Interrupt (ANSI)
3	SIGQUIT	Quit (POSIX)
4	SIGILL	Illegal instruction (ANSI)
5	SIGTRAP	Trace trap (POSIX)
6	SIGABRT SIGIOT	Abort (ANSI) IOT trap (4.2 BSD)
7	SIGBUS	BUS error (4.2 BSD)
8	SIGFPE	Floating-point exception (ANSI)
9	SIGKILL	Kill, unblockable (POSIX)
10	SIGUSR1	User-defined signal 1 (POSIX)
11	SIGSEGV	Segmentation violation (ANSI)
12	SIGUSR2	User-defined signal 2 (POSIX)
13	SIGPIPE	Broken pipe (POSIX)
14	SIGALRM	Alarm clock (POSIX)
15	SIGTERM	Termination (ANSI)
16	SIGSTKFLT	Stack fault
17	SIGCHLD SIGCLD	Child status has changed (POSIX) Same as SIGCHLD (System V)
18	SIGCONT	Continue (POSIX)

Signal #	name	comment
19	SIGSTOP	Stop, unblockable (POSIX)
20	SIGTSTP	Keyboard stop (POSIX)
21	SIGTTIN	Background read from tty (POSIX)
22	SIGTTOU	Background write to tty (POSIX)
23	SIGURG	Urgent condition on socket (4.2 BSD)
24	SIGXCPU	CPU limit exceeded (4.2 BSD)
25	SIGXFSZ	File size limit exceeded (4.2 BSD)
26	SIGVTALRM	Virtual alarm clock (4.2 BSD)
27	SIGPROF	Profiling alarm clock (4.2 BSD)
28	SIGWINCH	Window size change (4.3 BSD, Sun)
29	SIGIO SIGPOLL	I/O now possible (4.2 BSD) Pollable event occurred (System V)
30	SIGPWR	Power failure restart (System V)
31	SIGUNUSED	

23

Command line administration and scripting

In this chapter we introduce the WebSphere scripting solution called wsadmin and describe how some of the basic tasks that are performed by WebSphere Administrators, can be done using the scripting solution.There are two type of tasks, namely, the operational task and the configurational task. We will cover both type of tasks using wsadmin. The operational tasks deal with currently running objects in WebSphere installation and the configurational tasks deal with the configuration of WebSphere installations.

This chapter contains the following:

- Overview of scripting concept
- Overview of wsadmin basics.
- Common operational administration tasks using wsadmin
- Common configurational administration tasks using wsadmin
- Case study of managing webbank application using wsadmin
- Migration of WebSphere 4.0 WSCP scripts.

23.1 Overview of scripting solution

WebSphere Application Server 5.0 provides a new scripting interface called wsadmin. This scripting interface is different than wscp, provided in WebSphere Application server 4.0 or 3.5. Wasadmin is developed based on Bean Scripting Framework(BSF). The Bean Scripting Framework is an open source project to implement an architecture for incorporating scripting into Java applications and applets. Therefore the BSF architecture works as a interface between Java application and scripting languages. It allows scripting language to do the following:

- Lookup a pre-registered bean and access a pre-declared bean
- Register a newly created bean
- Do all bean operations
- Bind events to scripts in the scripting language

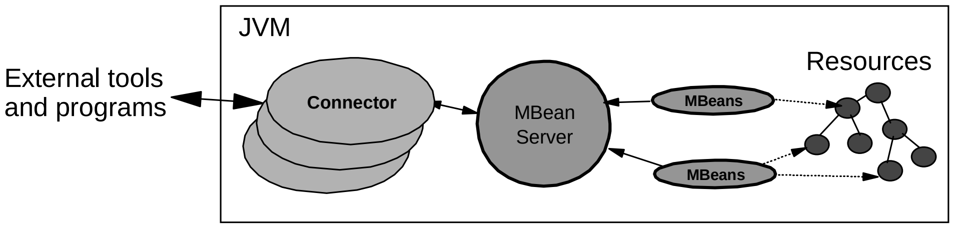


Figure 23-1 Scripting Solution of WebSphere 5.0

Figure 23-1 on page 972 depicts major components involved in WebSphere 5.0 scripting solution.

Since wsadmin uses BSF, it can make various Java objects available through language specific interfaces to scripts.There are four objects available to scripts as shown in Figure 23-1 on page 972. These objects are as follows:

- AdminControl
- AdminConfig
- AdminApp
- Help

The AdminControl object is used to invoke operational commands. AdminConfig is used to invoke configurational command to create or modify WebSphere configurational elements. AdminApp is used for administering applications. The Help Object is used for general help.

The scripts use those objects to communicate with Mbeans running in WebSphere server process.The supported scripting languages all have ways to invoke methods on the exposed Java objects, like Mbeans.The time of writing this book only three scripting languages are tested and supported: Java Command language based on Tcl (JACL), Javascript and jpython. We use JACL to provide examples in Command line administration and scripting chapter.

Mbeans or Managed beans are Java objects those represent Java Management Extensions (JMX) resources. JMX is a technology developed by Sun Microsystems and leading companies in the management field, for providing a simple and standard way to instrument Java objects. JMX is an optional package addition to J2SE.

23.2 Java Management Extensions (JMX)

Note to Reviewer: This section has been copied from the system management Chapter. In the System management chapter a reference needs to be added that information for JMX is available in this chapter.

Java Management Extensions (JMX) is a framework that provides a standard way of exposing Java resources (application servers, for example) to a system management infrastructure. The JMX framework allows a provider to implement functions such as listing the configuration settings, and allows users to edit the settings. It also includes a notification layer that can be used by management applications to monitor events such as the startup of an application server.

The use of JMX opens the door to third-party management tool providers. Users of WebSphere 5.0 are no longer restricted to IBM supplied management tools.

JMX is a Java specification (JSR-003) that is part of J2SE 1.4.

Author Comment: Might need to move this section to the wsadmin chapter. It is really only of interest to those writing scripts or building their own admin interface.

23.2.1 JMX key features

The key features of the WebSphere implementation of JMX include:

- All WebSphere 5.0 processes run the JMX agent.
- All WebSphere 5.0 runtime administration is performed through JMX operations.
- Connectors are used to connect a JMX agent to a remote JMX-enabled management application. The following connectors were supported at the time of writing:
  - SOAP JMX Connector
  - RMI/IIOP JMX Connector
- Protocol adapters provide a management view of the JMX agent through a given protocol. Management applications that connect to a protocol adapter are usually specific to a given protocol. The following protocol adapters were supported at the time of writing.

Author Comment: missing a list here, or are the following bullets part of the list and just need a name on the front?

  –
- Allows a runtime object's configuration settings to be queries and updated.
- Allows application components and resources to be loaded, initialized, changed and monitored in the runtime.

23.2.2 JMX benefits

The use of JMX for management functions in IBM WebSphere Application Server provides the following benefits:

- Enables Java applications to be managed without heavy investment.

  Relies on a core managed object server that acts as a management agent. Java applications simply need to embed a managed object server and make some of its functionality available as one or several MBeans registered with the object server.

- Provides a scalable management architecture.
  - Every JMX agent service is an independent module that can be plugged into the management agent.
  - The API is extensible, allowing new WebSphere and custom application features to be easily added and exposed through this management interface.
  - Integrates existing management solutions
  - JMX smart agents are capable of being managed through HTML browsers or by various management protocols such as Web Services, JMS and SNMP.
  - Each process is self-sufficient when it comes to the management of its resources. There is no central point of control. In principle, a JMX-enabled management client could be connected to any managed process and interact with the MBeans hosted by that process.
  - JMX does allow a single, flat, domain-wide approach to system management. Separate processes interact through MBean proxies allowing a single management client to seamlessly navigate through a network of managed processes.
- Defines only the interfaces necessary for management.

  A standard API for exposing application and administrative resources to management tools.

23.2.3 JMX architecture

The Java Management Extensions (JMX) architecture is structured into three layers:

- Instrumentation layer

  Dictates how resources can be wrapped within special Java Beans, called Management Beans (MBeans).
- Agent layer

  Consists of the MBean server and agents which provide a management infrastructure. Services implemented include:
  - Monitoring
  - Event notification
  - Timers
- Management layer

Defines how external management applications can interact with the underlying layers, in terms of protocols, APIs, etc. This layer uses an implementation of the *Distributed Services* specification (JSR-077), which is not yet part of the J2EE specification

The layered architecture of JMX is summarized in Figure 23-2.

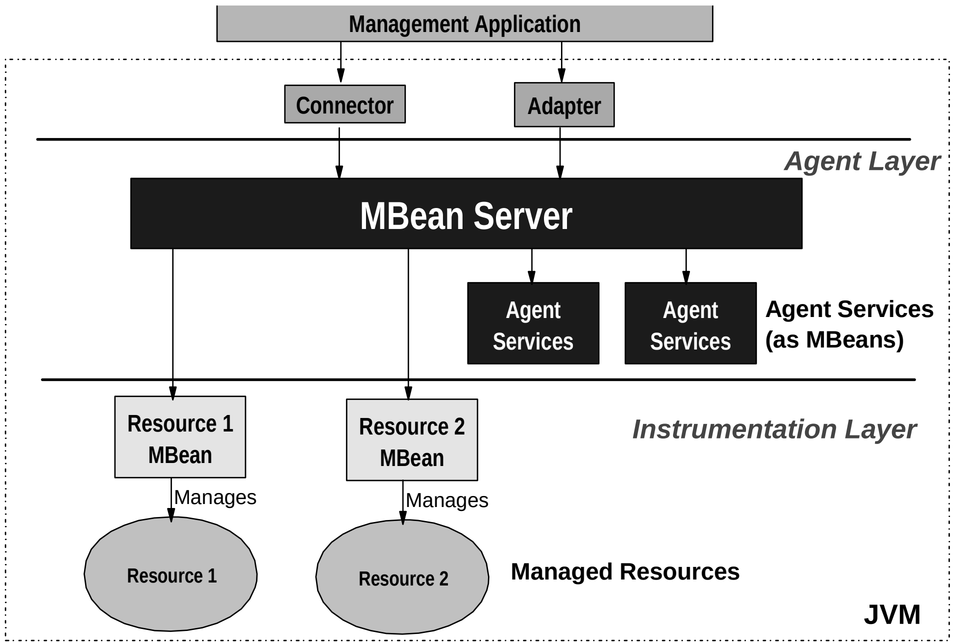


Figure 23-2 JMX architecture

How does JMX work?

Resources are managed by JMX Management Beans (MBeans). These are not EJBs, but simple JavaBeans that need to conform to certain design patterns outlined in the JMX specification.

Providers that want to instrument their systems with JMX need to provide a series of MBeans. Each MBean is meant to wrap (or represent) a certain runtime resource. For instance, in order to expose an Application Server as a manageable resource, WebSphere needs to provide an Application Server MBean.

External applications can interact with the MBeans through the use of JMX connectors and protocol adapters, as shown in Figure 23-1 on page 972

Connectors are used to connect an agent with a remote JMX-enabled management application. This form of communication involves a connector in the JMX agent and a connector client in the management application.

The key features of JMX connectors are:

- Connectors are oriented to the transport mechanism. For example, a provider may provide an RMI connector that allows Java applications to interact remotely with the MBeans.
- The connector translates JavaBeans calls to the a protocol stream.
- There is a 1:1 mapping between client method invocations and MBean operations.
- This is the low level API for accessing MBeans.

Note:

1. Connectors cannot be used with existing non-JMX aware management applications, as connectors communicate “native JMX” information.
2. The RMI connector requires that the ORB be configured and running in the process. The JMSServer does not need the ORB (other than if the RMI connector were desired) and the ORB adds a 30 Mbyte overhead to the process. Therefore the default configuration for the JMSServer is not to include the ORB and therefore the RMI connector cannot be configured either.

Protocol adapters

Protocol adapters provide a management view of the JMX agent through a given protocol. Management applications that connect to a protocol adapter are usually specific to the given protocol.

Author Comment: Would a list of protocol adapters be appropriate here?

The key features of JMX protocol adapters are:

- Protocol adapters adapt operations of MBeans and the MBean Server into a representation in the given protocol, and possibly into a different information model, for example SNMP or HTTP.
- There is not a 1:1 mapping between client method invocations and MBean operations.
- This is the high level API for accessing MBeans.

MBean Server

Each JMX enabled JVM contains an MBean Server that registers all the MBeans in the system. It is the MBean Server that provides access to all of its registered MBeans.

Note: There is only one MBean Server per JVM.

Both connectors and protocol adapters use the services of the MBean Server in order to apply the management operation they receive to the MBeans, and in order to forward notifications to the management system. Connector and protocol adapter communication is summarized in Figure 23-3.

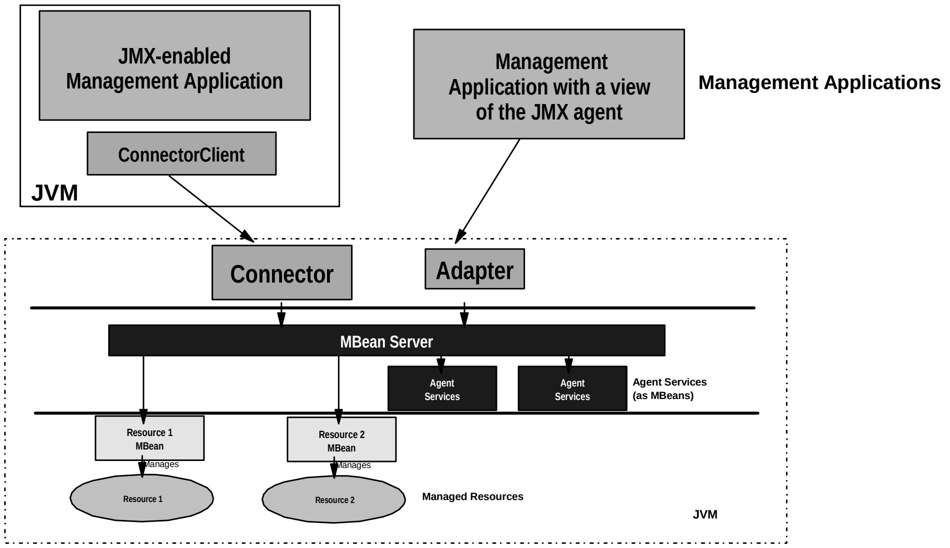


Figure 23-3 JMX Connectors and Adapters

23.2.4 JMX distributed administration

Figure 23-4 shows how the JMX architecture fits into the overall distributed administration topology of IBM WebSphere Application Server Network Deployment.

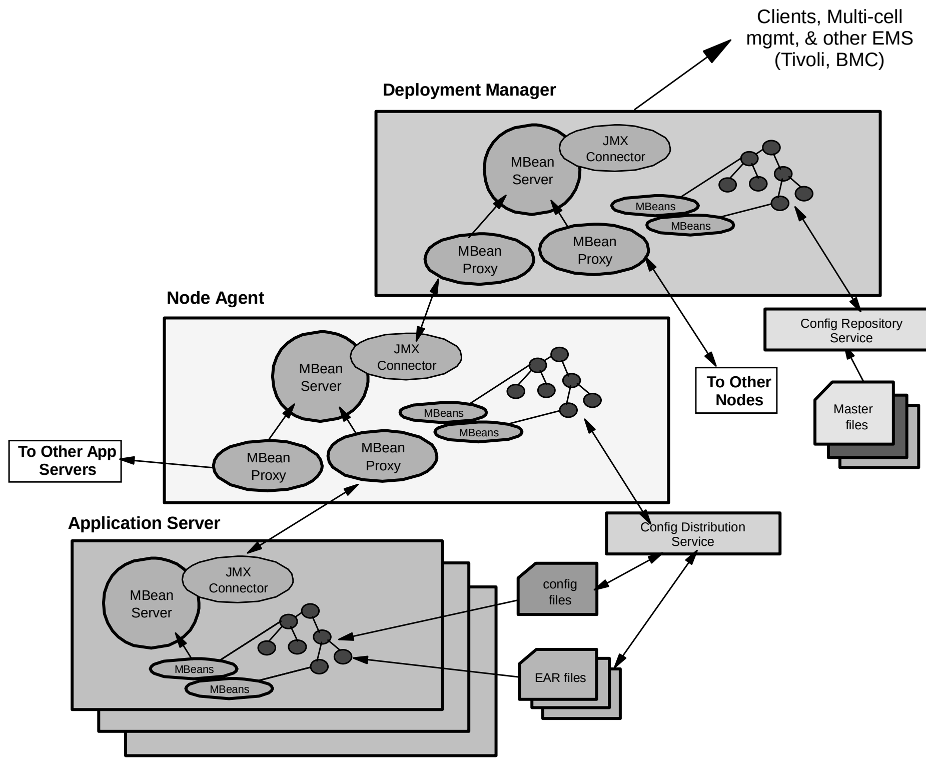


Figure 23-4 WebSphere 5.0 JMX distributed administration

The key points of this distributed administration architecture are:

- Internal MBeans (local to the JVM) register with the local MBean Server.
- External MBeans have a local proxy to their MBeanServer. The proxy registers with the local MBean Server. The MBean proxy allows the local MBean Server to pass the message to an external MBean Server located on:
  - Another server
  - Node agent
  - Deployment manager
- A node agent has an MBean proxy for all servers within its node. However, MBean proxies for other nodes are not used.
- The deployment manager has MBean proxies for all node agents in the cell.

The configuration of MBean proxies is shown in Figure 23-5.

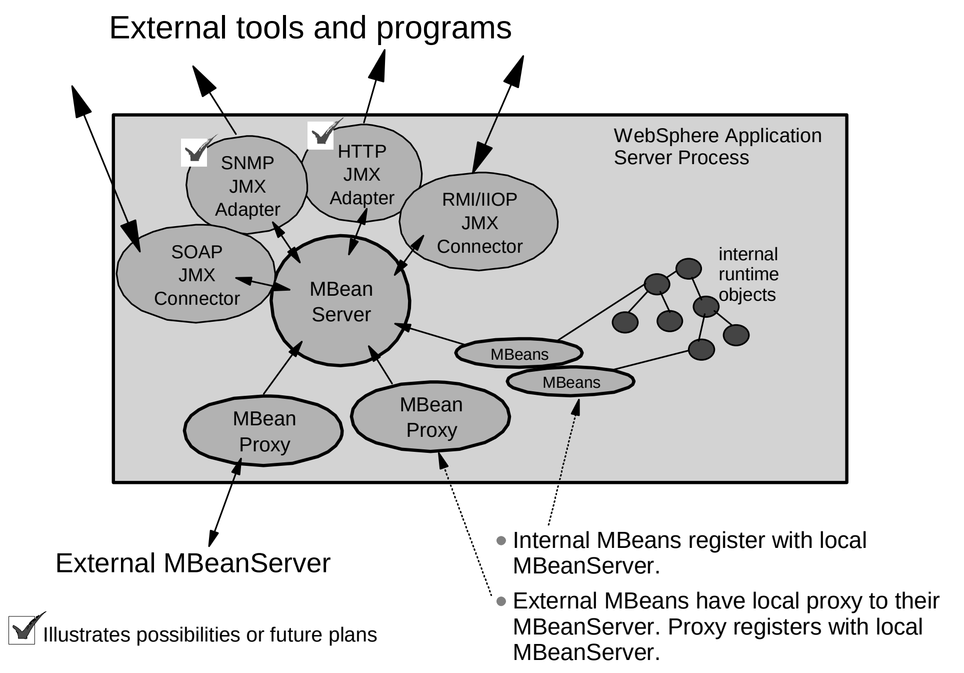

External tools and programs
WebSphere Application Server Process
SNMP JMX Adapter
HTTP JMX Adapter
RMI/IIOP JMX Connector
SOAP JMX Connector
MBean Server
internal runtime objects
MBeans
MBeans
MBean Proxy
MBean Proxy
External MBeanServer
Internal MBeans register with local MBeanServer.
External MBeans have local proxy to their MBeanServer. Proxy registers with local MBeanServer.
Illustrates possibilities or future plans

Figure 23-5 WebSphere 5.0 JMX architecture

23.2.5 JMX MBeans

IBM WebSphere Application Server provides a number of Management Beans (MBeans), each of which may have different functions / operations available. For example:

- An application server MBean may expose operations such as start and stop.
- An application MBean may expose operations such as install and uninstall.

For detailed information on how to access and use the provided JMX MBeans of WebSphere 5.0, see ......

Author Comment: Update with specific reference when scripting chapter is finished

23.2.6 JMX usage scenarios

Some of the more common JMX usage scenarios you will encounter are:

- **Internal product usage**:

All the WebSphere 5.0 administration clients use JMX:

- Web administration console.
- wsadmin scripting client
- Admin client Java API.

- **External programmatic administration**

  In general, most external users will not be exposed to the use of JMX. Instead, they will access administration functions through the standard WebSphere 5.0 admin clients.

  However, external users would need to access JMX in the following scenarios:

  - External programs written to control the IBM WebSphere Application Server Network Deployment runtime and its WebSphere resources by programmatically accessing the JMX API.
  - Third party applications that include custom JMX MBeans as part of their deployed code, allowing the applications components and resources to be managed via the JMX API.

23.3 Overview of wsadmin basics

In this section we describe what needs to be configured for launching wsadmin, how to start wsadmin and getting on line information.

23.3.1 Configuring wsadmin

The scripting environment for wsadmin can be set either by command line or specifying those information into a file called properties file. The properties file can be set in three level. The first level is using the system default properties file called wsadmin.properties. The file is located in <$WAS_ROOT>/properties directory. If decided not to use the system default properties file, this file can be customized and it can be placed in the user home directory or in $user_home. The third and final level is launching wsadmin and specifying properties on the command line.

The properties file gets invoked before the main script or invoking wsadmin interactively so that proper environment is set for wsadmin. The important properties included in the properties files are listed as follows:

- com.ibm.ws.scripting.connectionType - SOAP, RMI or JMX
- com.ibm.scripting.host - Hostname of system
- com.ibm.ws.scripting.defaultLang - jacl, Javascript or Jython

- com.ibm.ws.scripting.traceFile - File for trace information
- com.ibm.ws.scripting.traceString=com.ibm.*=all=enabled

Some of the listed properties in the wsadmin.properties file are commented out by default. An example is com.ibm.ws.scripting.traceString. If WebSphere Administrator wants to trace activities of wsadmin, the comment sign #, has to be taken out in wsadmin.proeperties file.

Similarly some of the properties contain values. For example, the property com.ibm.ws.scripting.connectionType has default value SOAP. The significance of setting this value to SOAP means when a scripting process is invoked SOAP connector is used to communicate with the WebSphere server.If necessary administrator can change those values to appropriate values.

Besides properties file there is another type of file that needs consideration. This file is called profile. A profile is just a script but it is invoked before the main script or before invoking wsadmin in interactive mode. The purpose of the profile file is to customize the environment in which scripts run. For example a profile can be set for Jacl scripting language that makes Jacl specific variables or procedure available to the interactive session or main script.

23.3.2 Launching wsadmin

Starting wsadmin in WebSphere 5.0 is simple. On Windows platform, at the command prompt, from the bin directory in WebSphere installation root directory type **`wsadmin`** and press **`Enter`** as follows:

`<WAS_HOME>\bin\wsadmin.bat`.

On an Unix machine wsadmin can be invoked at the command prompt by specifying the full qualified path for **`wsadmin.sh`** script or by typing **`./wsadmin.sh`** from <$WAS_HOME>/bin directory.

Since we use Windows platform for running WebSphere we will deal with command syntax for Windows only.

To get syntax related help for wsadmin.bat or wsadmin.sh script type **`wsadmin -?`** and press **`Enter`**.In Example 23-1 on page 983 we show the output of running **`wsadmin`** with **`-?`** option. Example 23-1 on page 983 shows the available command line options for **`wsadmin command`**. Some of these options have equivalent in properties file. However options specified on the command line will override those set in a properties file. Some of the important options are discussed in detail.

Example 23-1 wsadmin command-line options

```
C:\PROGRA~1\WEBSPH~1\APPSER~1\bin>wsadmin -?|more
WASX7001I: wsadmin is the executable for WebSphere scripting.
Syntax:

wsadmin
        [ -h(elp)  ]
        [ -?  ]
        [ -c <command> ]
        [ -p <properties_file_name>]
        [ -profile <profile_script_name>]
        [ -f <script_file_name>]
        [ -lang  language]
        [ -conntype
                SOAP
                        [-host host_name]
                        [-port port_number]
                        [-user userid]
                        [-password password] |
                RMI
                        [-host host_name]
                        [-port port_number]
                        [-user userid]
                        [-password password] |
                JMS <jms parms> |
                NONE
        ]
        [ script parameters ]
```

Running a single command

The **-c** option is used to execute a single command using wsadmin. Example 23-2 on page 983 shows the usage of single command execution using wsadmin. In the example we use AdminControl object to query the Node name of the WebSphere server process.

Example 23-2 Running a Single command in wsadmin

```
C:\PROGRA~1\WEBSPH~1\APPSER~1\bin>wsadmin -c "$AdminControl getNode"

WASX7209I: Connected to process "server1" on node mka0kkwd using SOAP
connector;  The type of process is: UnManagedProcess
mka0kkwd
```

Running scripts

The -f option is used to execute a script file. Example 23-3 on page 984 shows a two line JACL script named myScript.jacl and its execution in wsadmin. The script has an extension jacl and thereby wsadmin BSF knows it is an JACL script. If no extension is used **`-lang`** option is required to identify the scripting language. However if wsadmin.properties file exists either in <WAS_HOME>\properties directory or in the home directory of the wsadmin user and if it has the property com.ibm.ws.scripting.defaultLang set to a scripting language, wsadmin BSF knows what scripting engine to be used.

Example 23-3 Running a script in wsadmin

```
C:\SCRIPT~1>more myScript.jacl
puts "This is an example JACL script"
puts "[$AdminControl getNode]"

C:\Program Files\WebSphere\AppServer\bin>wsadmin -f
c:\ScriptExamples\myScript.jacl

WASX7209I: Connected to process "server1" on node mka0kkwd using SOAP
connector;  The type of process is: UnManagedProcess

This is an example JACL script
mka0kkwd
```

Running profile

The **`-profile`** command line option can be used to specify a profile script. The profile can be used to perform whatever standard initialization is required. Several **`-profile`** options can be used on the command line and those are invoked in the order given.

Specifying properties file

The **`-p`** option is used to specify a properties file other than wsadmin.properties either located in the <WAS_HOME>\properties directory or in the $user_home directory. Example 23-4 on page 985 shows invoking wsadmin and executing myScript.jacl and specifying a properties file named, custom.properties. For this example we made a copy of wsadmin.properties in the temp directory and renamed the file as custom.properties

Example 23-4 Specifying properties file on the command line

```
C:\Program Files\WebSphere\AppServer\bin>wsadmin -f
c:\scriptexamples\myScript.jacl -p c:\temp\custom.properties

WASX7209I: Connected to process "server1" on node mka0kkwd using SOAP
connector;  The type of process is: UnMana
gedProcess

This is an example JACL script
mka0kkwd
C:\Program Files\WebSphere\AppServer\bin>
```

23.3.3 Getting online help

In section 23.1, “Overview of scripting solution” on page 972 and in Figure 23-1 on page 972 we mentioned that wsadmin exposed four different objects, namely, AdminControl, AdminConfig, AdminApp and Help for managing WebSphere from the command line. While writing script for WebSphere administration it is necessary to get information about available methods, those can be executed on those objects. The Help object provide a means to get information about the available method.

Example 23-5 on page 985 shows the syntax of getting online help for public methods avaialbe for AdminControl object. Similarly public method for AdminConfig, AdminApp and Help itself can be listed.

Example 23-5 Excerpt of online Help for AdminControl Object

```
C:\Program Files\WebSphere\AppServer\bin>wsadmin -c "$Help AdminControl"
...................
completeObjectName
                Return a String version of an object name given a
                template name
getAttribute_jmx
                Given ObjectName and name of attribute, returns value of
                attribute
getAttribute    Given String version of ObjectName and name of attribute,
                returns value of attribute
getAttributes_jmx
                Given ObjectName and array of attribute names, returns
                AttributeList
getAttributes   Given String version of ObjectName and attribute names,
                returns String of name value pairs

getCell         returns the cell name of the connected server
getConfigId     Given String version of ObjectName, return a config id for
                the corresponding configuration object, if any.
getDefaultDomain        ?
getDomainName   returns "WebSphere" .........
```

Getting help for a specific command

You can find help for any command provided by the wsadmin objects using the following syntax:

```
wsadmin> <wsadmin_object> help <method_name>
```

Example 23-6 shows the usage of the above mentioned syntax. This example shows how to obtain help information for **`completeObjectName`** AdminControl object. Similarly you can obtain information related to any command/method of any object.

Example 23-6 Getting help for a command

```
wsadmin>$AdminControl help completeObjectName
WASX7049I: Method: completeObjectName

        Arguments: object name, template

        Description: Returns a String version of an object name that matches
        the "template."  For example, the template might be "type=Server,*"
        If there are several MBeans that match the template, the first match
        is returned.
```

Finding information for running MBeans

Mbeans are discussed in Section 23.2.5, “JMX MBeans” on page 980. In this section we describe how can you find detail information about MBeans by interrogating the MBean server with wsadmin objects.

MBeans represent running objects in WebSphere. Each WebSphere server contains an MBean server. Each Mbean has a type. The AdminControl object provided by wsadmin is used to interact with running MBeans. In particular the queryNames can be used to find what Mbeans are running. The simples form of this command in Jacl is as follows:

```
$AdminControl queryNames *
```

This returns a list of all MBeans of all types to the MBean server. Depending on what kind of server your scripting client is attached to, this list may contain MBeans that are running in many different servers.

- If the client is attached to a stand alone WebSphere Application server, the list will contain only MBeans running on that server.
- If the client is attached to a Node Agent in a Network Deployment environment, the list will contain Mbeans running in the Node agent as well as MBeans running on all application servers on that node.
- If the client is attached to a Deployment Manager, the list will contain MBeans running in the Deployment Manager, in all Node Agents communicating with that Deployment Manager, and all application servers on all the nodes served by those Node Agents.

Example 23-7 shows a jacl script to collect the detail information of running MBeans into a file called mbean.txt. In the file each MBean information is preceded by a label called “ObjectName” to clearly identify each MBean information. The script is written for windows platform and before running the script a file, named mbean.txt needs to be created in the temp folder. To run this script on UNIX platform the absolute path of the mbean.txt file has to be modified to /tmp/mbean.txt within the script.

Example 23-7 Finding information for running MBeans

```
set file "c:\\temp\\mbean.txt"
set logFile [open $file a]
set mblist [$AdminControl queryNames "*:*"]
foreach item $mblist {
puts $logFile "ObjectName: $item"
}
close $logFile
```

The items comprising the list returned by queryNames are string representation of JMX ObjectName objects. For example one such item might be as follows:

```
WebSphere:cell=M055245Network,name=dmgr,mbeanIdentifier=server.xml#Server_1
,type=Server,node=M055245Manager,process=dmgr,processType=DeploymentManager
```

This represents a Deployment Manager, named dmgr running in M055245 cell on Node M055245Manager. You can see in the above string that ObjectNames have two parts: a domain and a list of key properties. For MBeans created by WebSphere will be WebSphere. WebSphere includes the following key properties on its ObjectNames:

- Name

- Type
- Cell
- Node
- Process
- mbeanIdentifier

You can use any of these key properties to narrow the scope of the queryNames. For example you can list of all MBeans that represent “Server” objects on the node myNode as follows:

```
$AdminControl queryNames WebSphere:type=Server,node=myNode,*
```

Note: You will get an empty list back if you don’t use * at the end of the ObjectName. Without the * the argument to queryNames is not a wild card.

Finding attributes and operations for running MBeans

The Help object can be used to find information about any running MBeans in the system. The easiest way to use either the “attributes” or “operations” command for help. You first must get the ObjectName for the running MBeans. This can be done using the completeObjectName method of the AdminControl object. Then you can use Help object to examine the attributes.

Example 23-8 shows the commands to find attributes information for a running MBeans representing a Server on node named mka0kkwd. The first command initialize a variable called serv to the completeObjectName of the Servers running on Node mkaokkwd. The attribute command of Help object lists all the available attributes for the MBean.

Example 23-8 Finding attributes for a running MBean

```
wsadmin>set serv [$AdminControl completeObjectName type=Server,node=mka0kkwd,*]

WebSphere:cell=mka0kkwdNetwork,name=mka0kkwd,mbeanIdentifier=server.xml#Server_
1,type=Server,node=mka0kkwd,process=mka0kkwd,processType=NodeAgent

wsadmin>$Help attributes $serv
Attribute                       Type                            Access
name                            java.lang.String                RO
pid                             java.lang.String                RO
cellName                        java.lang.String                RO
deployedObjects                 [Ljava.lang.String;             RO
javaVMs                         [Ljava.lang.String;             RO
nodeName                        java.lang.String                RO
```

```
processType                     java.lang.String                RO
resources                       [Ljava.lang.String;             RO
serverVersion                   java.lang.String                RO
serverVendor                    java.lang.String                RO
state                           java.lang.String                RO
platformName                    java.lang.String                RO
platformVersion                 java.lang.String                RO
```

Similarly you can use the operations command to find out what operations are supported by this MBean. Example 23-9 shows the usage of operation command and its output.

Example 23-9 Finding operations information for a running MBean

```
wsadmin>$Help operations $serv
Operation
java.lang.String getName()
java.lang.String getPid()
java.lang.String getCellName()
[Ljava.lang.String; getDeployedObjects()
[Ljava.lang.String; getJavaVMs()
java.lang.String getNodeName()
java.lang.String getProcessType()
[Ljava.lang.String; getResources()
java.lang.String getServerVersion()
java.lang.String getServerVendor()
java.lang.String getState()
java.lang.String getPlatformName()
java.lang.String getPlatformVersion()
java.lang.String getProductVersion(java.lang.String)
java.lang.String getComponentVersion(java.lang.String)
java.lang.String getEFixVersion(java.lang.String)
java.lang.String getExtensionVersion(java.lang.String)
```

Finding information about configurational object

You can find information about configurational objects of WebSphere Application Server using commands provided by the AdminConfig object. Prior to making changes to a configuration you need to know what objects are available to work on, their default value, and the parent for a particular object. The AdminConfig object provides commands that can be used to get the basic information. These commands are as follows:

- **`types`** - this returns a list of config object types a user can manipulate. Example 23-10 shows the partial output of the types command. Each of this object type can be created or modified.

Example 23-10 Output of types command

```
wsadmin>$AdminConfig types

AdminService
Agent
ApplicationConfig
ApplicationContainer
ApplicationDeployment
ApplicationServer
AuthMechanism
AuthenticationTarget
AuthorizationConfig
AuthorizationProvider
AuthorizationTableImpl
BackupCluster
CMPConnectorFactory
CORBAObjectNameSpaceBinding
Cell
CellManager
Classloader
ClusterMember
ClusteredTarget
CommonSecureInterop
Component
ConfigSynchronizationService
```

- **`getid`** - This command returns the configuration id for an object.Configuration objects are named using a convention that uses a combination of the “display name” for the object and its configuration id.The config id uniquely identifies an object and therefore config id can be used in any configuration command that requires a config object name. You can see in Example 23-11 we obtain the unique configuration id for default server, server1, running on node mka0kkwd.
  To obtain the configuration id for server1 we pass a hierarchical strings within double quote as an argument. The passing argument **“**`/Node:mka0kkwd/Server:server1/`**”** tells the getid command to obtain the configuration id for server1, a Server type object running on node mka0kkwd, a Node type object. You can see the usage of “**/**” to separate one set of object type and its values from another set of object type and value pair. The “**:**” is used to separate the value from the object type in a object type and value pair.

Example 23-11 Finding configuration information of an object

```
wsadmin>$AdminConfig getid "/Node:mka0kkwd/Server:server1/"

server1(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#Server_
1)
```

Note: Configuration objects are named using a combination of the “display name” and its configuration id. The display name comes first, followed by the configuration id in parenthesis. An example of such an object name is :

```
server1(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#Serv
er_1)
```

For those pieces of configuration data that do not have display names, the name of the object simply consists of the configuration id in parenthesis. An example of such an object name is as follows:

```
(cells/testcell/nodes/testnodes/resource.xml#J2CSecurityPermission_1)
```

Since the “config id” part of the name is completely unique, a user can always use it without the pre-pended display name in any command that requires a config object name.

- **`list`** - this returns a list of objects of a given type. In an WebSphere Application Server environment there are several object types and there are many objects configured those have the same object type. You can display all the objects of same object type using the **`list`** command.
  Example 23-12 list all objects of object type **DynamicCache** in our test environment. You can see the list command returns two objects of DynamicCache type, one for server1 and another for the WebbankAS server.

Example 23-12 Finding objects of the same object type

```
wsadmin>$AdminConfig list DynamicCache

(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/WebbankAS:server.xml#DynamicCache
_1)
(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#DynamicCache_1
)
```

- **`defaults`** - this returns a table of attributes, thier types and defaults if any. Each object in WebSphere environment has an object type and each object type has attributes which may or may not have default values.

Example 23-13 shows the usage of those default command to list the attributes and default values for those attributes for object type **`DynamicCache`**.

Example 23-13 Finding attributes and default values for an object type

```
wsadmin>$AdminConfig defaults DynamicCache
Attribute                       Type                            Default
enable                          Boolean
cacheSize                       Integer                         2000
defaultPriority                 Integer                         1
replicationType                 ENUM
pushFrequency                   Integer                         0
enableDiskOffload               Boolean                         false
diskOffloadLocation             String
hashSize                        Integer
context                         ServiceContext
properties                      Property
cacheGroups                     ExternalCacheGroup
cacheReplication                DRSSettings
```

- **`parents`** - this return a list of object types that can serve as the parent of the given type. In WebSphere environment an object can contain other objects. Therefore a parent child relationship exists in WebSphere configuration. For example a Node type object contains Server type object. Therefore Node is a parent to server type object. In the Example 23-11 of getid command to obtain the configuration id for server1 we pass the value of parent type object Node along with the value of Server type object. In order to know which object type can be used as parent type object you can use the parents command as shown in Example 23-14.

Example 23-14 Finding the parent type objects for an object type

```
wsadmin>$AdminConfig parents Server
Node
```

Input and output of config object attributes

The **`attributes`** command of AdminConfig objects are part of the wsadmin on-line help features. The information displayed does not represent any particular config object but it represents config object types or object “meta-data”. The meta-data is used to show, modify and create actual config objects.In this section we describe how to interpret the output of those commands.

The attribute command displays the type and name of each attribute defined for a given type of configuration object. The list of attributes is displayed slightly different in different language. In Jacl each attribute representation is separated by comma, so the output can be treated as Tcl list. The name of each attribute is always a string, generally beginning with a lower-case letter. But the types of attributes vary on board range. We use an example to show various types of attributes.

Example 23-15 shows the output of the **attribute** command for configurational object called **DynamicCache**. There are twelve attributes listed. You can see there are four simple integer attributes, two are Boolean attributes and one is a String attribute. The **cacheGroups** and *properties* objects are list of objects indicated by * at the end of the **ExternalCacheGroup** and **Property** respectively. This is called nested attribute. We can run another **attribute** command to see the composition of these nested attribute. You can see the second **attribute** command for **ExternalCacheGroup** returns three attributes.

Example 23-15 Output of attribute command of AdminConfig object

```
wsadmin>$AdminConfig attributes DynamicCache

"cacheGroups ExternalCacheGroup*"
"cacheReplication DRSSettings"
"cacheSize Integer"
"context ServiceContext@"
"defaultPriority Integer"
"diskOffloadLocation String"
"enable Boolean"
"enableDiskOffload Boolean"
"hashSize Integer"
"properties Property(TypedProperty)*"
"pushFrequency Integer"
"replicationType ENUM(PULL, PUSH, PUSH_PULL, NONE)"

wsadmin>$AdminConfig  attributes ExternalCacheGroup
"members ExternalCacheGroupMember*"
"name String"
"type ENUM(SHARED, NOT_SHARED)"

wsadmin>$AdminConfig  attributes TypedProperty

"description String"
"name String"
"required Boolean"
"type String"
"validationExpression String"
"value String"
```

You can see in Example 23-15 that the **properties** attribute has a value that is also a list of objects of the **Property** type. The **Property** type is a generic type, so its sub-types are listed, that is TypedProperty. The **replicationType** attribute is an **ENUM** type attribute whose value must be one of the four strings provided in parenthesis.

The **show** command of AdminConfig object can be used to display the top level attributes of a given object. You can see the top level attributes for object server1 in Example 23-16 using the **show** command.

Example 23-16 Finding top level attributes for a given object

```
wsadmin>set serv [$AdminConfig getid "/Node:mka0kkwd/Server:server1/"]
server1(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#Server_
1)
wsadmin>$AdminConfig show $serv
{components
{(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#NameServer_1)
(cells/mka0kkwdNetwork/node/mka0kkwd/servers/server1:server.xml#ApplicationServ
er_1)}}
{customServices {}}
{errorStreamRedirect
(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#StreamRedirect
_1)}
{name server1}
{outputStreamRedirect
(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#StreamRedirect
_2)}
{processDefinition
(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#JavaProcessDef
_1)}
{services
{(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#PMIService_1)
(cells/mka0kkwdNetwork/nodes/
ka0kkwd/servers/server1:server.xml#AdminService_1)
(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#Tra
eService_1)
(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#RASLoggingServ
ice_1) (cells/mka0kkwdNetwor
/nodes/mka0kkwd/servers/server1:server.xml#ObjectRequestBroker_1)}}
{stateManagement
(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#StateManageabl
e_1)}
{statisticsProvider
(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#StatisticsProv
ider_1)}
```

If you want to see values for a particular attribute you can use **showAttribute** command as shown in Example 23-17. We list values for name attribute and the services attribute of server1.

Example 23-17 Finding values for particular attribute for a given object

```
wsadmin>$AdminConfig showAttribute $serv name
server1

wsadmin>$AdminConfig showAttribute $serv services

{(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#PMIService_1)
(cells/mka0kkwdNetwork/nodes/mka0kkwd/s
rvers/server1:server.xml#AdminService_1)
(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#TraceService_
)
(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:server.xml#RASLoggingServ
ice_1) (cells/mka0kkwdNetwork/nodes/mk
0kkwd/servers/server1:server.xml#ObjectRequestBroker_1)}
```

Another useful command to list all attributes and thier values is **`showall`** command of AdminConfig object. This command returns all the attributes of a given object.

23.4 Common operational administrative tasks using wsadmin

In this section we describe how wsadmin can be used to carry out common operation tasks in an WebSphere environment. The section is organized into two sections as follows:

- General approach for operational tasks
- Specific examples of common administrative tasks

Note: Some of the examples used in this section need WebSphere ND installed. Therefore we install both WebSphere 5.0 base and ND on the same machine. To show the command syntax we use WebSphere sample applications that get installed while installing WebSphere.

23.4.1 General approach for operational tasks

In order to invoke an operation on a running MBean, you first need to know the ObjectName of the running object. Then you invoke the command on a fully qualified objectName. This means that invoking operations usually involves two type of commands:

- Find the ObjectName

- Invoke the operation

In simple cases two commands can combined into one command.

Note: You can use **`queryNames`** and **`completeObjectName`** command of AdminControl object to identify the objectName of a running object. We discuss about how to find running object, their attributes and operations supported by them in 23.3.3, “Getting online help” on page 985

Similarly in order to change an attribute of a running object, you first need to know the ObjectName of that running object. This means that getting or setting attributes involves a sequence of two commands:

- Fin the ObjectName of the running object/MBeans
- Get or set attributes of for that running object

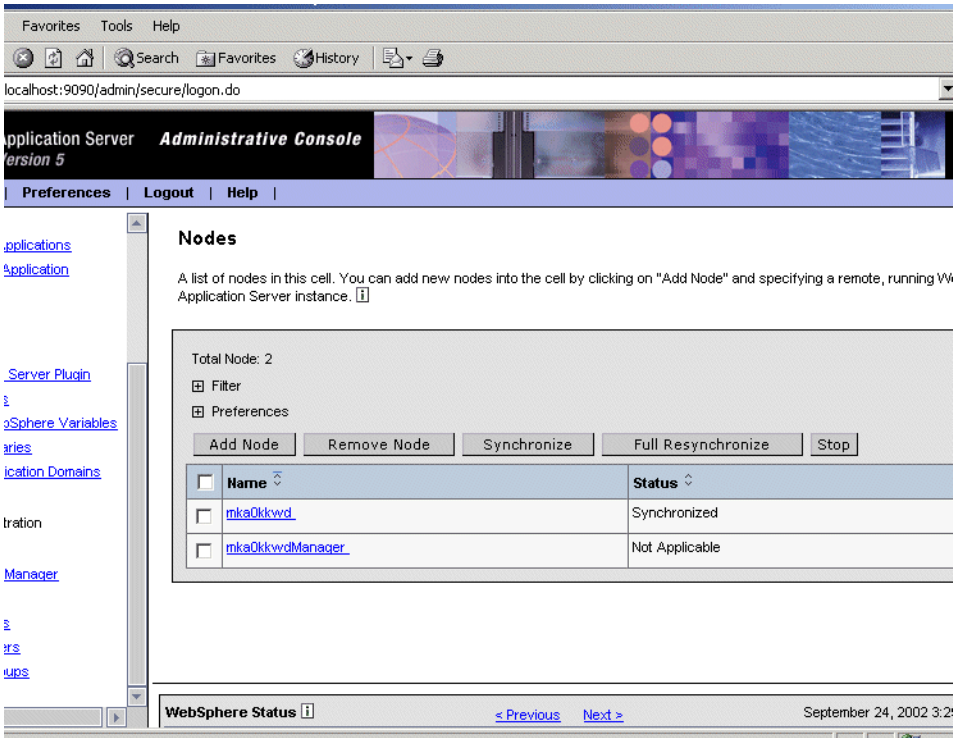
23.4.2 Specific examples of common administrative tasks

Common operational tasks performed using wsadmin include:

- Deployment Manager
  - Start the deployment manager
  - Stop the deployment manager
- Node
  - Start a node agent
  - Stop a node agent
- Application Server
  - Start an application server
  - stop an application server
- Enterprise application
  - Start an enterprise application
  - Stop an enterprise application
  - View installed applications
- Miscellaneous
  - Start a cluster
  - Stop a cluster

- – Regenerate plug-in configuration file
- – Enable tracing for WebSphere component

To understand the examples described in this chapter you need to know our WebSphere Application Server setup. We installed WebSphere Application Server V5.0 base and ND on the same machine to keep the environment simple. We used the default sever Server1 and default applications for our examples. Figure 23-6 on page 997 shows the Node information for our test environment.



Nodes

A list of nodes in this cell. You can add new nodes into the cell by clicking on "Add Node" and specifying a remote, running W Application Server instance.

Total Node: 2

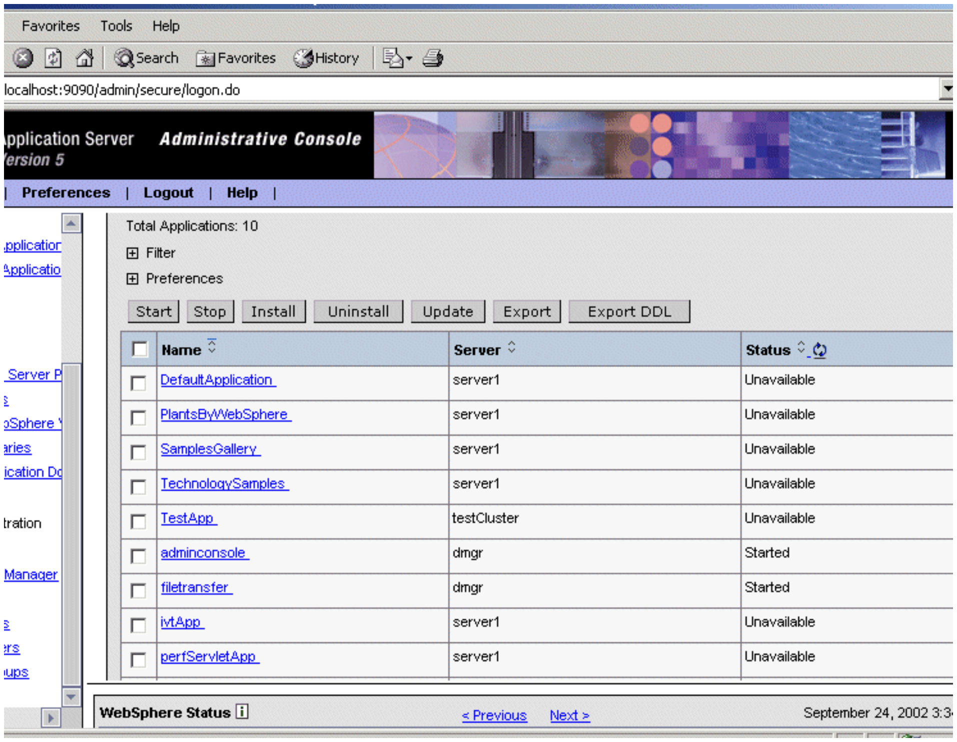
⊞ Filter

⊞ Preferences

	Name	Status
☐	mka0kkwd	Synchronized
☐	mka0kkwdManager	Not Applicable

Figure 23-6 Node setup for test environment

You can see our all installed application and Servers information of our test environment in Figure 23-7 on page 998. Again all operational examples are provided based on the environment depicted in Figure 23-6 and Figure 23-7.


Favorites Tools Help
Search
Favorites
History
localhost:9090/admin/secure/logon.do
pplication Server Administrative Console
ersion 5
Preferences | Logout | Help |
Start
Stop
Install
Uninstall
Update
Export
Export DDL

Total Applications: 10

⊞ Filter

⊞ Preferences

	Name	Server	Status
☐	DefaultApplication	server1	Unavailable
☐	PlantsByWebSphere	server1	Unavailable
☐	SamplesGallery	server1	Unavailable
☐	TechnologySamples	server1	Unavailable
☐	TestApp	testCluster	Unavailable
☐	adminconsole	dmgr	Started
☐	filetransfer	dmgr	Started
☐	ivtApp	server1	Unavailable
☐	perfServletApp	server1	Unavailable

WebSphere Status < Previous Next > September 24, 2002 3:3

Figure 23-7 Installed applications and Servers

Deployment Manager

This section describes how to start and stop tasks on the deployment manager using the WebSphere scripting interface wsadmin.

Starting the deployment manager

In a Network Deployment environment deployment manager runs either on a dedicated machine or on a machine that runs as a node also. Wsadmin works on Mbean only and since the Mbean representing the deployment manager is not available unless deployment manager process is started, therefore wsadmin can not start the deployment manager. To start the deployment manager see 12.7.10, “Start the Network Deployment environment” on page 442

Stopping the deployment manager

The deployment manager can be stopped using the AdminControl object and invoking the stopserver method. To invoke the stopServer method it is required to provide the deployment manager name and the node name.Example 23-18 on page 999 shows the an example of stopping deployment manager.

Example 23-18 Stopping deployment manager using a single line command

```
wsadmin>$AdminControl stopServer dmgr MO55245Manager

WASX7337I: Invoked stop for server "dmgr" Waiting for stop completion.
WASX7264I: Stop completed for server "dmgr" on node "MO55245Manager"
```

The similar stop operation can be performed by invoking stop method from AdminControl object on the Mbean representing the deployment manager. We need to identify the Mbean that represents the deployment manager using queryNames method of AdminControl object.Example 23-19 on page 999 shows the example command to query Mbeans information the deployment manager and the nested command to stop the deployment manager.

Example 23-19 Getting Mbean information for deployment Manager and nested command to stop the deployment manager

```
wsadmin>$AdminControl queryNames type=Server,node=MO55245Manager,*

WebSphere:cell=MO55245Network,name=dmgr,mbeanIdentifier=server.xml#Server_1,typ
e=Server,node=MO55245Manager,process=dmgr,processType=DeploymentManager

wsadmin>$AdminControl invoke [$AdminControl queryNamestype=Server,node=MO55245M
anager,*] stop {}
```

Node

This section describes how to perform common administration tasks on nodes and thier node agent using wsadmin.

Starting a node agent

It is mentioned that wsadmin works on Mbeans and therefore if a JVM process does not exist wsadmin fails to work on Mbeans because Mbeans are not available. If node agent server process is not running the Mbean representing the node agent is not available. Therefore the node agent can not be started using wsadmin.

To start node agent on a node follow the steps below:

1. Change directory to bin directory of the base application server installation for that node.
2. Run **`startNode`** from the command line.

If successful, the node agent server process id will be displayed on the screen. If there are any errors check the log file for the node agent process:

<WAS_ROOT>/logs/dmgr/SystemOut.log

Stopping a node agent

Node agent is a JVM process that controls all of the WebSphere managed processes on a node. Therefore stopping a node agent limits the ability to issue any further commands against managed servers.In a WebSphere cell there is one node agent per node.

Node agent can be stopped using by invoking stopServer method of AdminControl object. As argument we need to supply the name of the node agent server and the name of the node.Example 23-20 on page 1000 shows the command to stop a node agent.

Example 23-20 Single line command to stop node agent

```
wsadmin>$AdminControl stopServer mka0kkwd mka0kkwd

WASX7337I: Invoked stop for server "mka0kkwd" Waiting for stop completion.
WASX7264I: Stop completed for server "mka0kkwd" on node "mka0kkwd"
```

The stop operation of node agent can be performed by invoking the stop operation on the Mbean representing the node agent. We need to identify the Mbean for node agent queryNames method. Example 23-21 on page 1000 shows the command syntax to query Mbean information for the node agent.Also in this example we show the nested command to invoke stop method on the identified Mbean.

Example 23-21 Getting Mbean information for node agent

```
wsadmin>$AdminControl queryNames type=Server,node=mka0kkwd,name=mka0kkwd,*

WebSphere:cell=NetworkDeploymentCell,name=mka0kkwd,mbeanIdentifier=cells/Networ
kDeploymentCell/nodes/mka0kkwd/servers/mka0kkwd/server.xml#Server_1,type=Server
,node=mka0kkwd,process=mka0kkwd,processType=NodeAgent
wsadmin>
wsadmin>$AdminControl invoke [$AdminControl queryNames
type=Server,node=mka0kkwd,name=mka0kkwd,*] stop {}
```

Application Server

Start an application server

In Network Deployment environment node agent can start an application server. The easiest way to do this by using the AdminControl’s startServer command.In this section we use the default application server server1 to show the starting of the application server. Example 23-22 on page 1001 shows the command for starting an application server.

Example 23-22 Start an application server

```
wsadmin>$AdminControl startServer mka0kkwd_server1
WASX7262I: Start completed for server "mka0kkwd_server1" on node "mka0kkwd"
```

The similar operation can be performed invoking method **`launchProcess`** on the Mbean of node agent on the node.We find the NodeAgent Mbean for the node and then invokes **`launchProcess`** on the node agent Mbean.

Example 23-23 Start an application server by invoking launchProcess on node agent Mbean

```
wsadmin>$AdminControl queryNames type=NodeAgent,node=mka0kkwd,*
WebSphere:cell=NetworkDeploymentCell,name=NodeAgent,mbeanIdentifier=NodeAgent,t
ype=NodeAgent,node=mka0kkwd,process=mka0kkwd

wsadmin>$AdminControl invoke [$AdminControl queryNames
type=NodeAgent,node=mka0kkwd,*] launchProcess mka0kkwd_server1
```

Stop an application server

This example shows how an application server can be stopped on a node. We will show the example using default application server, server1, installed on a node where we are running both WebSphere Network Deployment Manager and a node. Example 23-24 on page 1002 shows the command for stopping an application server.

Example 23-24 Stop an application server

```
wsadmin>$AdminControl stopServer mka0kkwd_server1
```

We can also use AdminControl object of wsadmin to invoke the stop method on the application server. To invoke the stop method on the application server we need to identify the Mbean representing the application server uniquely. AdminControl has a queryName method to query Mbeans information running in a WebSphere cell. Example 23-25 on page 1002 shows the command to query Mbean information of the application server.

Example 23-25 Mbean information for Application Server server1

```
wsadmin>$AdminControl queryNames
type=Server,node=mka0kkwd,name=mka0kkwd_server1,*

WebSphere:cell=NetworkDeploymentCell,name=mka0kkwd_server1,mbeanIdentifier=cell
s/NetworkDeploymentCell/nodes/mka0kkwd/servers/mka0kkwd_server1/server.xml#Serv
er_1,type=Server,node=mka0kkwd,process=mka0kkwd_server1,processType=ManagedProc
ess
```

Example 23-26 on page 1002 shows the nested command syntax to stop the application server.

Attention: The examples of this section uses specific node name, application server name based on our installation. The general syntax for stopping an application server should be as follows:

```
>$AdminControl invoke [$AdminControl completeObjectName
type=Server,node=<node name>,name=< name of the application
server>,*] stop {}
```

Example 23-26 Stopping an Application server

```
wsadmin>$AdminControl invoke [$AdminControl completeObjectName
type=Server,node=mka0kkwd,name=mka0kkwd_server1,*] stop {}
```

The nested command in Example 23-26 on page 1002 can be simplified by setting a variable to the object name. In Example 23-27 on page 1003 we show how to set variable to an object name and then we can invoke stop method on that variable.

Example 23-27 Set a variable to object name

```
wsadmin>set myserv [$AdminControl completeObjectName
type=Server,node=mka0kkwd,name=mka0kkwd_server1,*]

$AdminControl invoke $myserv stop {}
```

If there are multiple application servers running on a node it can be scripted to stop all the servers from a single script. Example 23-28 on page 1003 lists the script to stop all application servers on the node mka0kkwd. We hard coded the node name in the script to make script simple. It is possible to write JACL code that will accept a node name from the command line or a menu and stop all application servers on the specified node.

Example 23-28 Stop all application servers on a node

```
set servername [AdminControl queryNames type=Server,node=mka0kkwd,*]
foreach item $servername {
puts “Stopping server : $item”
if {[catch {AdminControl invoke $item stop {}} result]} {
puts “Caught exception invoking stop: $result”
} else {
puts “invoked stop $item”
}
}
```

Enterprise applications

This section describes how to perform common administration tasks on enterprise applications using the scripting interface, wsadmin.

View installed application

The command to list to applications installed under an application server is as follows:

$AdminApp list

Example 23-29 on page 1004 shows the command to list application in interactive mode and the output of the command

Example 23-29 Listing installed applications

```
wsadmin> $AdminApp list

adminconsole
WebbankV5
TechnologySamples
DefaultApplication
petstore
ivtApp
PlantsByWebSphere
MDBSamples
SamplesGallery
filetransfer
```

Also by querying Mbeans for running applications we can list the installed applications on a node. Example 23-30 on page 1004 shows the command to list the running applications on node mka0kkwd and it also shows the excerpt of the output of the command.

Example 23-30 Listing applications by Mbeans query

```
wsadmin>$AdminControl queryNames WebSphere:type=Application,node=mka0kkwd,*

WebSphere:cell=NetworkDeploymentCell,name=DefaultApplication,mbeanIdentifier=ce
lls/NetworkDeploymentCell/applications/DefaultApplication/deployment.xml#Applic
ationDeployment_1,type=Application,node=mka0kkwd,Server=mka0kkwd_server1,proces
s=mka0kkwd_server1,J2EEName=DefaultApplication.......
```

The running objects are represented by MBeans. If an object is not running the MBean for that object does not exists. Based on this we can write a simple Jacl script that will display if an application is running or not running. The script to display the status of applications is not an straight forward. In Example 23-31 we list all the installed applications with **AdminApp** object. The data obtained via **AdminApp list** command is configurational data and therefore we can not interrogate the configuration data to find out what application is running and what application is not running. So for each installed application we query names of MBeans to see if MBean exists or not using **AdminContro**l object. If the application is running the **queryNames** for **AdminControl** object returns a name otherwise it returns a null value.

Example 23-31 Script to display the status of Applications

```
set application [$AdminApp list]
foreach app $application {
   set objName [$AdminControl queryNames type=Application,name=$app,*]
   if {[ llength $objName] ==0} {
   puts "The Application $app is not running"
   } else {
   puts "The Application $app is running"
   }
    }
```

Stop a running application

To stop a running application we use AdminControl command and invoke stopApplication method on the Mbean of the running application. We need to identify the Mbean for the application server before we invoke the stopApplication method. Example 23-32 on page 1005 shows the sequence of command to stop application. The first command identifies the Mbean for the application server and then in the second command is nested command to run stopApplication method for DefaultApplication.

Example 23-32 Stopping a running application

```
wsadmin> set appservername [$AdminControl queryNames
type=ApplicationManager,node=mka0kkwd,*]

wsadmin>$AdminControl invoke $appservername stopApplication DefaultApplication
```

Start a stopped application

To start a stopped application we use AdminControl command and invoke startApplication method on the stopped application. We need to pass the identity of the Mbean of the application server before we invoke the startApplication method. Example 23-33 on page 1006 shows the sequence of command to start the DefaultApplication application.

Example 23-33 Start an stopped application

```
wsadmin> set appservername [$AdminControl queryNames
type=ApplicationManager,node=mka0kkwd,*]

wsadmin>$AdminControl invoke $appservername starApplication DefaultApplication
```

Miscellaneous

Start a cluster

In a Network Deployment environment it is possible to implement server cluster to take advantage of Work Load manager. In this section we describe the command to start an existing cluster.You can see the sequence of commands inExample 23-34. The first command list all the configured clusters in our WebSphere cell. It returns with one cluster information because we have only one cluster configured. The second command initializes a variable called “clid” with the cluster object name. The final command invokes start method on the cluster id object.

Example 23-34 Start a Cluster

```
wsadmin>$AdminControl queryNames type=Cluster,*
WebSphere:cell=mka0kkwdNetwork,name=testCluster,mbeanIdentifier=testCluster,typ
e=Cluster,node=mka0kkwdManager,process=dmgr

wsadmin>set clid [$AdminControl completeObjectName
type=Cluster,name=testCluster,*]
WebSphere:cell=mka0kkwdNetwork,name=testCluster,mbeanIdentifier=testCluster,typ
e=Cluster,node=mka0kkwdManager,process=dmgr

wsadmin>$AdminControl invoke $clid start
```

Stop a cluster

In this section we describe the command to stop an existing cluster.You can see the sequence of commands in Example 23-35 on page 1006. The first command list all the configured clusters in our WebSphere cell. It returns with one cluster information because we have only one cluster configured. The second command initialize a variable called clid with the cluster object name. The final command invokes stop method on the cluster id object.

Example 23-35 Stop a cluster

```
wsadmin>$AdminControl queryNames type=Cluster,*
WebSphere:cell=mka0kkwdNetwork,name=testCluster,mbeanIdentifier=testCluster,typ
e=Cluster,node=mka0kkwdManager,process=dmgr
```

```
wsadmin>set clid [$AdminControl completeObjectName
type=Cluster,name=testCluster,*]
WebSphere:cell=mka0kkwdNetwork,name=testCluster,mbeanIdentifier=testCluster,typ
e=Cluster,node=mka0kkwdManager,process=dmgr

wsadmin>$AdminControl invoke $clid stop
```

Regenerating Web server plug-in configuration

Certain WebSphere configuration changes need to regenerate the plug-in file. There is a PluginCfgGenerator MBean. We can use this MBean to regenerate the plug-in on a node by invoking **`generatePluginConfigFile`** command. We need to pass Mbean of a node as the argument to execute the command on a particular node. Example 23-36 on page 1007 shows the sequence of commands to regenerate the plug-in file. The first command identify the MBean for plug-in on node mka0kkwdNetmgrNode and then the second command try to generate the plug-in.

Example 23-36 Regenerate plug-in

```
wsadmin>set generator [$AdminControl queryNames
type=PluginCfgGenerator,node=mka0kkwdNetmgrNode,*]

wsadmin>$AdminControl invoke $nodename generatePluginConfigFile
```

Enable tracing for WebSphere component

In a problem determination situation it is required to set tracing for different WebSphere components. More information about how to enable tracing is available in Chapter 22, “Troubleshooting” on page 903. In this section we show a simple example of enabling tracing for a server process using **`setAttribute`** command on `TraceService` Mbean.

In a Network Deployment environment there is multiple server processes and therefore multiple . You can see in Example 23-37 on page 1007 queryNames command lists three TraceService Mbeans for three server processes.

Example 23-37 List of TraceService MBeans

```
wsadmin>$AdminControl queryNames type=TraceService,*

WebSphere:cell=NetworkDeploymentCell,name=TraceService,mbeanIdentifier=cells/Ne
tworkDeploymentCell/nodes/mka0kkwd/servers/mka0kkwd/server.xml#TraceService_1,t
ype=TraceService,node=mka0kkwd,process=mka0kkwd
```

```
WebSphere:cell=NetworkDeploymentCell,name=TraceService,mbeanIdentifier=cells/Ne
tworkDeploymentCell/nodes/mka0kkwd/servers/mka0kkwd_server1/server.xml#TraceSer
vice_1,type=TraceService,node=mka0kkwd,process=mka0kkwd_server1

WebSphere:cell=NetworkDeploymentCell,name=TraceService,mbeanIdentifier=cells/Nt
workDeploymentCell/nodes/mka0kkwdNetmgrNode/servers/netmgr/server.xml#TraceSeri
ce_1,type=TraceService,node=mka0kkwdNetmgrNode,process=netmgr
```

We need to locate the TraceService MBean for server1 process using the **completeObject** command of AdminControl Object as shown in Example 23-38 on page 1008. You can see in the example, to make the command simple we initialize a variable called “ts” to the value of the object name of the Mbean running tracing service on node mka0kkwdd and in the second step we execute the **setAttribute** command to enable the tracing.

Example 23-38 Enable tracing using TraceService mbean

```
wsadmin>set ts [$AdminControl completeObjectName
type=TraceService,process=mka0kkwd,*]

wsadmin>$AdminControl setAttribute $ts traceSpecification
com.ibm.ejs.*=all=enabled
```

23.5 Common configuration administrative tasks

In this section we describe how wsadmin can be used to create, modify and change WebSphere Application server configuration. The section is described into tow parts as follows:

- General approach for configurational administrative tasks
- Specific examples of configurational administrative tasks

23.5.1 General approach for configurational administrative tasks

The possible modifications of WebSphere configuration is big. It is not possible to document every possible modification in the limited span of the section. Instead we describe a general approach of modifying or creating WebSphere configuration with examples. The general approach has three steps:

- Find the object you want to change
  - Use **AdminConfig** object with **getid** command

- Make the change or create the configuration
  - Use **`AdminConfig`** object with **`modify`** or **`create`** command
- Save the change
  - Use **`AdminConfig`** object with **`save`** command

The **`create`** or **`modify`** command use attribute list. In general, the attribute supplied as a list of Jacl lists. A Jacl list can be constructed using name and value pair as follows

{ { name1 value1} {name2 value2} {name3 value3}.........}

The attribute for a WebSphere configuration object are often deeply nested. If it is required to modify a nested attribute you can get the id of the object you want to modify and modify it directly. This method is preferred method though it requires more lines of scripting.

23.5.2 Specific examples of WebSphere configurational task

This section describes how WebSphere configurational tasks can be done using wsadmin.The common configurational tasks include:

- Application Server
  - Create an application server
  - Remove an application server
- Enterprise Application
  - Install an enterprise application
  - Uninstall an enterprise application
  - Change attribute of an enterprise application
- Configure and modify WebSphere configuration
  - Configure virtual hosts
  - Configure JDBC providers
  - Edit an application server
  - Create a Cluster
  - Add member to a cluster

Note: We use the environment described in **Figure 23-6 on page 997** and **Figure 23-7 on page 998** to provide all examples of this section.

Application Server

Create an application server

In this section we describe how to create an application server using scripting interface wsadmin. We already have a default server server1 running on our node. The name of the 2nd server will be TestServer.

An application server can be created with two simple JACL commands. You can see the sequence of commands in Example 23-39 on page 1010. The first command initialize a variable called node to the value of the node configuration id, on which application server will be running. The second command execute the create method to define the server.

Example 23-39 Creating an application server

```
wsadmin>set node [$AdminConfig getid /Node:mka0kkwd/]

wsadmin>$AdminConfig create Server $node {{name testserver}}
```

The general syntax is as follows:

```
$adminConfig create <type> <node configuration id> <attribute
list>
```

Remove an application server

To remove an application server we execute to remove command of AdminConfig object. It is required to pass the application server configuration id as argument to remove command. You can see sequence of command in Example 23-40 on page 1010, the first command sets the configuration id of the application server as a value for the variable server. The second command executes the remove method.

Example 23-40 Remove an application server

```
wsadmin>set server [$AdminConfig getid /Node:mka0kkwd/Server:testserver/]

wsadmin>$AdminConfig remove $server
```

The general syntax for removing a server is as follows:

```
$AdminConfig remove <Config id of the application server>
```

Enterprise Application

Install an enterprise application

The **AdminApp** object is used to install an enterprise application. There are two ways **AdminApp** can be used to install an application:

- Interactive installation using **installInteractive** command
- Non interactive installation using **install** command

Options of install command

We install the `PerfSevletApp` application under our testserver, created in section, "Create an application server" on page 1010. We use the **install** command to install the application. If the **install** command is used. the required options must be provided on the command line in a single string.

There are many possible options for the **install** command.Each option consists of a dash followed by the option name.There are some options corresponds to actual install tasks, and take lists of value. There are some options can be categorized as non task options. We can lists the non-task options using the **option** command of **AdminApp** object. We list all the available non-task options in Example 23-41 on page 1011

Note: All options supplied for install command must be supplied in a single string. In JACL single string is formed by collecting all options within curly braces or double quotes:

```
$AdminApp install c:/temp/application.ear { -server serv2 -appname
-TestApp}
```

Example 23-41 List of non-task options of install command

```
wsadmin>$AdminApp options
WASX7105I: The following options are valid for any ear file

server
cluster
cell
node
installdir
was.install.root
```

```
configroot
appname
local - deprecated
verbose
contextroot
redeploy
redeploy.ignore.old
redeploy.ignore.new
depl.extension.reg
defaultbinding.datasource.jndi
defaultbinding.datasource.username
defaultbinding.datasource.password
defaultbinding.cf.jndi
defaultbinding.cf.resauth
defaultbinding.ejbjndi.prefix
defaultbinding.virtual.host
defaultbinding.force
defaultbinding.strategy.file
```

To list the task and non task options for a particular ear file run the option command against the ear file, as shown in Example 23-42 on page 1012. The task options take lists of values. Each member of the list corresponds to one instance of the required option data. More specific information about how to supply task options can be found in , “Edit an enterprise application” on page 1014

Example 23-42 List task and non-task option for an ear file

```
wsadmin>$AdminApp options c:/temp/PerfServletApp.ear

WASX7112I: The following tasks are valid for "c:/temp/PerfServletApp.ear"

MapRolesToUsers
MapWebModToVH
MapModulesToServers
preCompileJSPs
nopreCompileJSPs
distributeApp
nodistributeApp
deployejb
nodeployejb
useMetaDataFromBinary
nouseMetaDataFromBinary
usedefaultbindings
nousedefaultbindings
createMBeansForResources
```

```
nocreateMBeansForResources
reloadInterval
deployejb.classpath
deployejb.rmic
deployejb.dbtype
deployejb.dbschema
server
cluster
cell
node
installdir
was.install.root
configroot
appname
local - deprecated
verbose
contextroot
redeploy
redeploy.ignore.old
```

For more information about task options check online help information for **taskInfo** command of AdminApp object. Also we provide detail example of the taskInfo command usage in 23.6.6, “Installing the Webbank enterprise application” on page 1028

Installing perfservletapp application

In this section we use PerfServletApp ear file for our example installation. The perfServletApp is already installed under server1 as part of WebSphere base installation, therefore we install it with a new name called TestApp under the testserver. Also we made a change of context root of perfServletApp.ear file with **AAT**, prior to installing the application as TestApp. We changed the context root of perfServletApp.ear file to TestWasPerfTool to avoid the conflict with already installed application under server1. You can see the command syntax to install the TestApp application in Appendix 23-43, “Installing perfSevletApp under a new application server” on page 1013. In the example the name of the ear file is Deployed_PerfServletApp.ear because we regenerated the deployment code after changing the context root and saved the generated code as Deployed_PerfServletApp under c:\temp directory.

Example 23-43 Installing perfSevletApp under a new application server

```
wsadmin>$AdminApp install c:/temp/Deployed_PerfServletApp.ear {-server
testserver -appname TestApp}
```

```
wsadmin>$AdminConfig save
```

The general syntax for installing enterprise application is as follows:

`$AdminApp install <location of the ear file> { task or non-task option}`

Uninstall an enterprise application

An enterprise application can be removed with uninstall command of AdminApp object.You can see sequence of command in to remove our TestApp created in section , "Installing perfservletapp application" on page 1013.

Example 23-44 Uninstalling an enterprise application

```
wsadmin>$AdminApp uninstall TestApp

ADMA5017I: Uninstallation of TestApp started.
ADMA5102I: Deletion of config data for TestApp from config repository completed
successfully.
ADMA5106I: Application TestApp uninstalled successfully.

wsadmin>$AdminConfig save
```

The general syntax for uninstalling enterprise application is as follows:

`$AdminApp uninstall <name of the application>`

Edit an enterprise application

Editing of an enterprise application can be done either interactively or non-interactively. The following commands are available for editing:

- Non interactive - **`edit`** command
- Interactive - **`editInteractive`** command

In this section we show how to edit our TestApp with edit command. We want to map back our TestApp from testservr to server1. The command syntax is shown in Example 23-45, "Edit an enterprise application in non interactive mode" on page 1015. The example also demonstrates how you could supply task information while editing an application. The same syntax also applies in the event of installing an enterprise application using the **`install`** command with task options.

Note: Supplying the task option either for editing or installing enterprise application:

Each task is specified by its task name, followed by all the information needed to update the task in the following format

`-taskname {{item1a item2a item3a} {item1b item2b item3b}.......}`

Example 23-45 Edit an enterprise application in non interactive mode

```
wsadmin>$AdminApp edit TestApp { -MapModulesToServers {{perfservlet
perfServletApp.war,WEB-INF/web.xml
WebSphere:cell=mka0kkwdNetwork,node=mka0kkwd,server=server1}}}

wsadmin>$AdminConfig save
```

The general syntax for editing an enterprise application in non interactive mode is as follows:

`$AdminApp edit <name of the application> {-taskname {{item1a item2a item3a} {item1b item2b item3b}.......}}`

Configure and modify WebSphere configuration

Create a Virtual Host

In this section we create a virtual host, called Test_host following the approach mentioned in , “Configure and modify WebSphere configuration” on page 1015. Our first step is, finding out the object we want to change. Virtual host is a WebSphere resource defined in a WebSphere cell. Therefore by creating a virtual host we are modifying the configuration of the WebSphere cell object. We therefore need the id for our WebSphere cell.You see the command syntax to find the id of our WebSphere cell in Example 23-46 on page 1015. The example also shows that we initialize a variable called “cl” to the object id of the WebSphere cell.

Example 23-46 Find an object using AdminConfig command

```
wsadmin>set cl [$AdminConfig getid /Cell:mka0kkwdNetwork/]
mka0kkwdNetwork(cells/mka0kkwdNetwork:cell.xml#Cell_1)
```

Once we have the Object id we can make the necessary changes.
Example 23-47 on page 1016 shows the command syntax to create the virtual host. Once the virtual host is created we save the configuration.

Example 23-47 Create a virtual host called test_host

```
wsadmin>$AdminConfig create VirtualHost $cl {name test_host}

wsadmin>$AdminConfig save
```

Modify a virtual host

In this section we describe how to modify object with nested attributes, using virtual host as an example. You can see the command sequence to modify the test_host in Example 23-48 on page 1016. The first command to find the object for virtual host, named test_host. The second command modifies the test_host configuration with nested attribute.

Example 23-48 Modifying a virtual host

```
wsadmin>set vh [$AdminConfig getid /VirtualHost:test_host/]
test_host(cells/mka0kkwdNetwork:virtualhosts.xml#VirtualHost_3)

wsadmin>$AdminConfig modify $vh {{aliases {{{hostname *} {port 9080}}
{{hostname *} {port 80}} {{hostname *} {port 9081}}}}}

wsadmin>$AdminConfig save
```

Configure JDBC providers

Creating JDBC provider and JDBC driver are common configurational task while installing an application. In this section we configure aDB2 JDBC provider for our test node mka0kkwd. Since we are going to configure the JDBC provider at the node level we need to identify the object id for the node mka0kkwd.You can see the sequence of commands in Example 23-49 on page 1017. In this example we use **`createUsingTemplate`** command instead of simple create command. In the example the first command gets the config id of the node; the second command gets the config id of a specific template we would like to use and the final command creates a new object using that template.

Note: A group of templates are supplied with the WebSphere installation in config/template directory. You might want to use a template when creating a config object because the template might have values appropriate for your needs.The **`listTemplates`** command of the AdminConfig object prints a list of templates matching a given type. These templates may be used in invocations of **`createUsingTemplate`** method.

Example 23-49 Configuring a JDBC Driver

```
wsadmin>set node [$AdminConfig getid /Node:mka0kkwd/]

wsadmin>set temp1 [$AdminConfig listTemplates JDBCProvider "DB2 JDBC Provider"]

wsadmin>$AdminConfig createUsingTemplate JDBCProvider $node {{name testdriver}}
$temp1
```

Modify an application server

In WebSphere 5.0 updates made to the server's configuration never takes effect until the server is stopped and started.We change the ping interval for our testserver in Example 23-50 on page 1017. We find modifying application server from the command line is not an easy task because there are many nested objects and it is cumbersome to find an object. You can see the complexity of finding an object in Example 23-50 on page 1017

Example 23-50 Modify an application server

```
sadmin>$AdminControl stopServer testserver mka0kkwd

wsadmin>set s1 [$AdminConfig getid /Node:mka0kkwd/Server:testserver/]

wsadmin>set pdef [$AdminConfig list ProcessDef $s1]

wsadmin>$AdminConfig mp [$AdminConfig list MonitoringPolicy $pdef]

wsadmin>$AdminConfig modify $mp {{pingInterval 120}}

wsadmin>$AdminConfig save

sadmin>$AdminControl startpServer testserver mka0kkwd
```

In Example 23-50 on page 1017 the first command is find the object id for our testserver on node mka0kkwd. The second command set the variable pdef with the object id of embedded object ProcessDef for testserver. The second command uses **`list`** command to get the object id. The reason for using list command is, ProcessDef object has nested list attributes. The third command obtain the object id for another embedded object called MonitoringPolicy with the help of ProcessDef object id. The final command makes the modification of ping interval using the MonitoringPolicy object id as parent. The save command saves the change and the application server is restarted to pick up the change.

Tip: To find the parent object you can run **`$AdminConfig showall <Object id of the application server>`** and then analyze the output to understand the relationship among objects.

Another easy approach is to follow the AdminConsole screen to get an rough idea what are the parents object. For example, to change the PingInterval on the console you need to traverse through the following link:

Application Server->Process Definition ->Monitoring Policy -Ping Interval

However this is just a rough guess. To find the complex nested object you have to go through the output of the **`showall`** command.

Create a cluster

If you want to create a new ServerCluster, named as testserver using the existing testserver as a model. all you need is the config id of the existing server and the name to use for the new cluster. You can see the sequence of commands in Example 23-51 on page 1018.

Example 23-51 Create a ServerCluster Object

```
wsadmin>set serverid [$AdminConfig getid /Server:testserver/]

wsadmin>$AdminConfig convertToCluster $serverid testCluster
wsadmin>$AdminConfig save
```

Note: There is no command to remove a server from a cluster. If you no longer wish your server object part of the cluster, you should remove the server object.

Add a member to an existing ServerCluster

If you want to create a new member in an existing cluster you can use the command createClusterMember. You also need to know what node you want to run this cluster member. You can see we add a cluster member called, testserver2 on node mka0kkwd using the command sequence shown in Example 23-52 on page 1019.

Example 23-52 Add a cluster member

```
wsadmin>set testcluster [$AdminConfig getid /ServerCluster:testCluster/]

wsadmin>set nodeid [$AdminConfig getid /Node:mka0kkwd/]

wsadmin>$AdminConfig createClusterMember $testcluster $nodeid {{memberName
testserver2}}

wsadmin>$AdminConfig save
```

23.6 Case study of configuring and managing webbank application using wsadmin

In this section we explain how to write wsadmin script to create, configure and install all the bits and pieces needed to run the WebBank sample applications.

23.6.1 Assumptions

We make the following assumptions in this example:

- Your Web server is configured for serving the Webbak application on www.webbank.itso.ibm.com port 90.
- The WebSphere base and ND is installed on your machine and configured properly.
- The variable DB2_ROOT_PATH is set to C:\Program Files\SQLLIB\ on the Manage WebSphere Variable screen
- The Variable DB2_JDBC_PROVIDER_PATH is set to ${DB2_ROOT_PATH}/java on the Manage WebSphere Variable screen.
- The IBM DB2 instance is up and running
- DB2 administrative user is db2admin and password is db2admin
- Database WEBBANK is already created and populated

- The <WAS_ROOT> is c:\Program Files\WebSphere\
- The WebBank application EAR file is available at <WAS_ROOT>\AppServer\installableApps directory.
- To avoid naming conflicts, we assume that only the default installation has been performed.
- Target platform for the provided examples and scripts is Windows. For UNIX you only need to change the path specifications

The following sections show the necessary steps in the order of correct sequence. We run all **wsadmin** commands in interactive mode.

23.6.2 Creating the virtual host

Example 23-53 shows the command to create the virtual host

Example 23-53 Creating the virtual host webbank_vhost

```
wsadmin>set cl [$AdminConfig getid /Cell:mka0kkwdNetwork/]

wsadmin>$AdminConfig create VirtualHost $cl {{name webbank_vhost}}

wsadmin>$AdminConfig save
```

Example 23-54shows the command to add hostname and the port information for webbank_vhost.

Example 23-54 Modify the webbank_vhost alias information

```
wsadmin>set vh [$AdminConfig getid /VirtualHost:webbank_vhost/]

wsadmin>$AdminConfig modify $vh {{aliases {{{hostname www.webbank.itso.ibm.com}
{port 80}}}}}}

wsadmin>$AdminConfig save
```

With these settings the application server will accept URLs connecting to host name www.webbank.itso.ibm.com. on port 80. You do not need to specify the MIME table, since it will be created by default.

23.6.3 Creating the JDBC provider

Example 23-55 shows the command to configure the JDBC driver for a DB2 database.

Example 23-55 Creating the JDBC provider

```
wsadmin>set node [$AdminConfig getid /Node:mka0kkwd/]

wsadmin>$AdminConfig listTemplates JDBCProvider "DB2 JDBC Provider"
"DB2 JDBC Provider
(XA)(templates/system:jdbc-resource-provider-templates.xml#JDBCProvider_2)"
"DB2 JDBC
Provider(templates/system:jdbc-resource-provider-templates.xml#JDBCProvider_1)"

wsadmin>set temp1 [lindex [$AdminConfig listTemplates JDBCProvider "DB2 JDBC
Provider"] 1]
DB2 JDBC
Provider(templates/system:jdbc-resource-provider-templates.xml#JDBCProvider_1)


wsadmin>$AdminConfig createUsingTemplate JDBCProvider $node {{name
WebbankDB2Driver}} $temp1

wsadmin>$AdminConfig save
```

Note: lindex is a Jacl command. It retrieves an element from a list. This command treats list as Tcl list and returns the indexed element from it. For example 0 refers to the first element of the list. For further information about Jacl commands check the following URL:

http://www.javasim.org/ref.script/tcljacl.htm

After creating the driver you need to modify the attribute of the DB2 driver to set the “native path” variable correctly. Example 23-56 shows the commands to set the value of the “native path” variable of the DB2 driver.

Example 23-56 Setting the value of native path variable of DB2 driver

```
wsadmin>wsadmin>set dbprovider [$AdminConfig getid
/Node:mka0kkwd/JDBCProvider:WebbankDB2Driver/]
```

```
$AdminConfig modify $dbprovider {{nativepath "${DB2_ROOT_PATH}/bin"}}
wsadmin>$AdminConfig save

wsadmin>$AdminConfig modify $dbprovider {{nativepath
"${DB2_ROOT_PATH}/function"}}

wsadmin>$AdminConfig save
```

23.6.4 Creating data source

Note to Reviewer: This section has to be reviewed carefully to find out if there is a better way to create and modify data source. So far no documentation available to create a data source that is CMP-enabled and that uses a specific database. In this section each steps are broke down into details to make reader comprehend the complexity. Therefore couple commands for getting config id for JDBCProvider, datasource and save command are repeated. It is possible to remove these repetition if a better way to create data source is found.

To create a data source that ties with the `JDBCProvider` we need to follow the following steps:

- Create data source using a template
- Modify attributes `jndiName` for the data source
- Modify data source custom properties for `databaseName` attribute
- Create a `CMPConnectorFactory` to make the data source CMP enabled
- Create `J2C Auth Data Entries` for the JDBC data source and modify data source attribute Set Authentication Alias
- Test connectivity using the created data source

Example 23-57 shows the command sequence to create a data source. You can see in the example, first we get the configuration id of the `JDBCProvider` that we created earlier. Then we list all the available templates those can be used for creating data source using the **`listTemplates`** command. you need to select a particular template for your use with the **`lindex`** command. The final command of this example actually create the data source.

Example 23-57 Creating a data source using a template

```
wsadmin>set dbprovider [$AdminConfig getid
/Node:mka0kkwd/JDBCProvider:WebbankDB2Driver/]
WebbankDB2Driver(cells/mka0kkwdNetwork/nodes/mka0kkwd:resources.xml#JDBCProvide
r_1)
wsadmin>$AdminConfig listTemplates DataSource
(templates/system:jdbc-resource-provider-templates.xml#DataSource_1)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_2)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_3)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_4)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_DD_1)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_DD_1a)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_DD_2)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_DD_3)...

wsadmin>set temp2 [lindex [$AdminConfig listTemplates DataSource] 0]
(templates/system:jdbc-resource-provider-templates.xml#DataSource_1)

wsadmin>$AdminConfig createUsingTemplate DataSource $dbprovider {{name
webbankds}} $temp2
webbankds(cells/mka0kkwdNetwork/nodes/mka0kkwd:resources.xml#DataSource_2)

wsadmin>$AdminConfig save
```

Once you create the data source you need to set a **JNDI** name for the data source. This step is simply modifying the attribute called `jndiName`. You can see in Example 23-58 that we obtain the data source config id first and in the second command we modify the attribute using that config id.

Example 23-58 Set jndiiName for data source

```
wsadmin>set datasource [$AdminConfig getid
/JDBCProvider:WebbankDB2Driver/DataSource:webbankds/]
webbankds(cells/mka0kkwdNetwork/nodes/mka0kkwd:resources.xml#DataSource_2)

wsadmin>$AdminConfig modify $datasource {{jndiName webbankds}}

wsadmin>$AdminConfig save
```

You need to point your data source to correct database name . Our webbank application uses a db2 database named webbank. The `databaseName` attribute is a nested attribute and therefore we need to identify where it is nested. Like before we first obtain the config id for the **webankds** data source. Then we list all the `propertySet` to find out the location of the `databaseName` attribute. From the output of the **`showall`** command you can see the `databaseName` attribute is nested within the `resourcesProperties` attribute. Once you identify the location of the databaseName attribute, you can modify it using the modify command as shown in Example 23-59.

Note to Reviewer: The modify command instead of changing the existing databaseName attribute it adds another attribute with the same name but with a different value. As a result on the admin console it shows two databaseName attributes. One of them has the default value and the other has the correct value webbank. It has to be verified if this is a bug or a if there is better way of changing nested property for data source.

Example 23-59 Modify the databaseName attribute for the data source

```
wsadmin>set datasource [$AdminConfig getid
/JDBCProvider:WebbankDB2Driver/DataSource:webbankds/]
webbankds(cells/mka0kkwdNetwork/nodes/mka0kkwd:resources.xml#DataSource_2)


wsadmin>$AdminConfig showall $datasource propertySet
{propertySet {{resourceProperties {{{description "This is a required property.
The database name. For example, e
nter sample to make your Data Source point to jdbc:db2:sample."}
{name databaseName}
{required true}
{type java.lang.String}
{value sample}} {{description "The name of the database server."}
{name serverName}
{type java.lang.String}
{value {}}} {{description "The TCP/IP port number where the jdbc Provider
resides."}
{name portNumber}
{type java.lang.Integer}
{value {}}} {{description "The connection attributes. Refer to the DB2
reference for the list of connection attr
ibutes."}
{name connectionAttribute}
{type java.lang.String}
{value cursorhold=0}} {{description "The maximum time to attempt to connect a
database. If this value is non-zer
o, attempt to connect to the database will timeout when this specified value is
reached."}
{name loginTimeout}
{type java.lang.Integer}
{value 0}} }}}}}
```

```
wsadmin>$AdminConfig modify $datasource {{propertySet {{resourceProperties
{{{name databaseName} {value webbank}}}}}}}
```

You need to make your data source CMP enabled. To do that you need to create a `CMPConnectorFactory` for the given data source. To create a CMPConnectorFactory you need to know what is the parent object type of CMPConnectorFactory. This information is obtained with the parent command of the AdminConfig object. The `J2CResourceAdapter` is the parent object type for CMPConnectorFactory. you then need to identify for which J2CResourceAdapter the CMPConnectorFactory will be created. We Choose to use `J2CResourceAdapter` at the Node level. The final command is to create the CMPConnectorFactory with name and cmpDataSource attributes.

Example 23-60 Creating CMPConnectorFactory for a data source

```
wsadmin>$AdminConfig parents CMPConnectorFactory
J2CResourceAdapter


wsadmin>$AdminConfig list J2CResourceAdapter
"WebSphere Relational Resource
Adapter(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/server1:resources.xml#buil
tin_rra)"
"WebSphere Relational Resource
Adapter(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/webbankas:resources.xml#bu
iltin_rra)"
"WebSphere Relational Resource
Adapter(cells/mka0kkwdNetwork/nodes/mka0kkwd:resources.xml#builtin_rra)"
"WebSphere Relational Resource
Adapter(cells/mka0kkwdNetwork/nodes/mka0kkwdManager:resources.xml#builtin_rra)"
"WebSphere Relational Resource
Adapter(cells/mka0kkwdNetwork:resources.xml#builtin_rra)"


wsadmin>set rsadapter [lindex [$AdminConfig list J2CResourceAdapter] 2]
WebSphere Relational Resource
Adapter(cells/mka0kkwdNetwork/nodes/mka0kkwd:resources.xml#builtin_rra)


wsadmin>$AdminConfig create CMPConnectorFactory $rsadapter {{name webbankCMP}
{cmpDatasource
webbankds(cells/mka0kkwdNetwork/nodes/mka0kkwd:resources.xml#DataSource_2)}}

webbankCMP(cells/mka0kkwdNetwork/nodes/mka0kkwd:resources.xml#CMPConnectorFacto
ry_1)
```

```
wsadmin>$AdminConfig save
```

Note to Reviewer: The webbank application works even without creating the J2C Authentication Data Entry. !!!!. This is most likely we use default db2 user name and password. However it is required to add a step about to creates J2C Auth Data Entry and how to tag it with the data source definition.

The final step of creating a data source is to test if data source can successfully connect to database. You can see the command sequence in Example 23-61. The AdminControl object provide a command called **testConnection** to test the data source connectivity.

Example 23-61 Testing a data source

```
wsadmin>set datasource [$AdminConfig getid
/JDBCProvider:WebbankDB2Driver/DataSource:webbankds/]
webbankds(cells/mka0kkwdNetwork/nodes/mka0kkwd:resources.xml#DataSource_2)

sadmin>$AdminControl testConnection $datasource {{user db2admin} {password
db2admin}}
SRA8025I: Successfully connected to DataSource.
```

23.6.5 Creating the application server

To install the Webbank enterprise application you create an application server called webbankas. You can see the commands to create the application server inExample 23-62.

Example 23-62 Create application server WebbankAS

```
wsadmin>set node [$AdminConfig getid /Node:mka0kkwd/]

wsadmin>$AdminConfig create Server $node {{name webbankas}}

wsadmin>$AdminConfig save
```

The following section shows how to modify the new application server. Normally all of the following attributes are supplied with the **`create`** command. We do it this way for demonstration purposes.

Changing the Java virtual machine properties of the server

To change the values for JVM properties we need to locate nested attribute. Example 23-63 shows the step-by-step commands to locate nested attributes for JVM properties. Once we know what attributes contain what other attributes, it is to modify the attribute value.

Example 23-63 Locate nested attribute for Java virtual machine properties

```
wsadmin>set as [$AdminConfig getid /Node:mka0kkwd/Server:webbankas/]
WebbankAS(cells/mka0kkwdNetwork/nodes/mka0kkwd/servers/webbankas:server.xml#Ser
ver_1)
wsadmin>$AdminConfig show $as

wsadmin>wsadmin>set pd [$AdminConfig list ProcessDef $as]

wsadmin>$AdminConfig show $pd

wsadmin>set jvmattr [$AdminConfig list JavaVirtualMachine $pd]

wsadmin>$AdminConfig show $jvmattr

{bootClasspath {}}
{classpath {}}
{debugArgs "-Djava.compiler=NONE -Xdebug -Xnoagent
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=7777"}
{debugMode false}
{disableJIT false}
{genericJvmArguments {}}
{hprofArguments {}}
{initialHeapSize 0}
{maximumHeapSize 256}
{runHProf false}
{systemProperties {}}
{verboseModeClass false}
{verboseModeGarbageCollection false}
{verboseModeJNI false}
wsadmin>
```

Example 23-64 shows the command to change the `intialHeapSize` and `verboseModeGarbageCollection` attributes for the webbank application server. In

Example 23-63 on page 1027 we set a variable jvmattr and we use that variable to change the attribute.

Example 23-64 Changing JVM properties

```
wsadmin>$AdminConfig modify $jvmattr {{initialHeapSize 128}}

wsadmin>$AdminConfig modify $jvmattr {{verboseModeGarbageCollection true}}

wsadmin>$AdminConfig save
```

Changing the HTTP port for Web server plug-in

To change the default Web container port of the application server WebbankAS you need to get the id of the of the server's WebContainer and then modify the "transports" attribute of the WebContainer. Example 23-65 show s the command to change the HTTP transport of the Web container.

Example 23-65 Changing the HTTP transport of the application server

```
set as [$AdminConfig getid /Node:mka0kkwd/Server:WebbankAS/]

wsadmin>set webcontainer [$AdminConfig list WebContainer $as]

wsadmin>$AdminConfig modify $webcontainer {{transports:HTTPTransport
{{{sslEnabled true} {sslConfig DefaultSSLSettings} {address {{host *} {port
9084}}}}}}}

wsadmin>$AdminConfig save
```

23.6.6 Installing the Webbank enterprise application

You can use the wsadmin to deploy complete enterprise application. In this section we use the **`install`** command of **`AdminApp`** object to install our Webbank application. The install command needs to specify installation task and non task options on the command line. Since we created a new virtual host and a datasource we need to specify two task options. This two task options are as follows:

- DataSourceFor20CMPBeans - Specifying Data Sources for Individual 2.x CMP Beans
- MapWebModToVH - Selecting Virtual Hosts for Web Modules

DataSourceFor20CMPBeans task option

This is a task to specify a data source for EJBs in a EAR file overriding the default binding. You can see the default data source specified in the EAR file with the **`taskInfo`** command, as shown in Example 23-66. You can see from the output of the command that there are two EJBs and each of them has five attributes as follows:

- EJBModule
- EJB name
- URI
- DataSource JNDI Name
- Resource Authorization

Example 23-66 Find information of DataSourceFor20CMPBeans for a EAR file

```
wsadmin>$AdminApp taskInfo c:/new/Deployed_WebbankV5bis.ear
DataSourceFor20CMPBeans

DataSourceFor20CMPBeans: Specifying Data Sources for Individual 2.x CMP Beans

Specify an optional data source for each 2.x CMP bean. Mapping a specific data
source to a CMP bean will overrid
e the default data source for the module containing the Enterprise bean.
WASX7348I: Each element of the DataSourceFor20CMPBeans task consists of the
following 5 fields: "EJBModule", "EJ
B", "uri", "JNDI", "resAuth".
Of these fields, the following may be assigned values: "JNDI" "resAuth"
and the following are required:
The current contents of the task after running default bindings are:
EJBModule: webbankEntityEJBs
EJB: BranchAccount
uri: webbankEntityEJBs.jar,META-INF/ejb-jar.xml
JNDI: eis/webbank/jdbc/webbank_CMP
resAuth: cmpBinding.perConnectionFactory

EJBModule: webbankEntityEJBs
EJB: CustomerAccount
uri: webbankEntityEJBs.jar,META-INF/ejb-jar.xml
JNDI: eis/webbank/jdbc/webbank_CMP
```

```
resAuth: cmpBinding.perConnectionFactory
```

We want to change the JNDI name of the data source from `eis/webbank/jdbc/webbank_CMP to eis/webbankds_CMP` to match the name of our created data Source. You can see how to bundle up the attribute list in Example 23-67 using the Jacl **`list`** command.

Example 23-67 Building an attribute list for data source

```
wsadmin>set ds1 [list webbankEntityEJBs BranchAccount
webbankEntityEJBs.jar,META-INF/ejb-jar.xml eis/webbankds_CMP cmpBinding.pe
rConnectionFactory]

wsadmin>set ds2 [list webbankEntityEJBs CustomerAccount
webbankEntityEJBs.jar,META-INF/ejb-jar.xml eis/webbankds_CMP cmpBinding.
perConnectionFactory]

wsadmin>set ds [list $ds1 $ds2]
```

MapWebModToVH task option

This task option allow you to bind your virtual host of the WAR module, overriding the virtual host specified in the EAR file.You can see the specified default virtual host information running the taskInfo command against the EAR file, as shown inExample 23-68. You can see from the output of the **`taskinfo`** command that `Deployed_WebbankV5bis.ear` has two WAR modules, called `webbankWeb` and`webbankUnitTesting.wa`. Both of these are bound to default_host virtual host. We want to change this to Webbank_vhost.

Example 23-68 Finding information of Virtual host in WebBank EAR file

```
wsadmin>$AdminApp taskInfo c:/new/Deployed_WebbankV5bis.ear MapWebModToVH

MapWebModToVH: Selecting Virtual Hosts for Web Modules

Specify the virtual host where you want to install the Web modules contained in
your application. Web modules ca
n be installed on the same virtual host or dispersed among several hosts.

WASX7348I: Each element of the MapWebModToVH task consists of the following 3
fields: "webModule", "uri", "virtu
alHost".
Of these fields, the following may be assigned values: "virtualHost"
```

```
and the following are required: "virtualHost"

The current contents of the task after running default bindings are:
webModule: WebbankUnitTesting
uri: webbankUnitTesting.war,WEB-INF/web.xml
virtualHost: default_host

webModule: webbankWeb
uri: webbankWeb.war,WEB-INF/web.xml
virtualHost: default_host
```

you can see the sequence of **`list`** commands in to create another attribute list for the MapWebModToVH task option.

Example 23-69 Building attribute list for MapWebModToVH

```
wsadmin>set vh1 [ list WebbankUnitTesting
webbankUnitTesting.war,WEB-INF/web.xml webbank_vhost]

wsadmin>set vh2 [list webbankWeb webbankWeb.war,WEB-INF/web.xml webbank_vhost]

wsadmin>set vh [list $vh1 $vh2]
}
```

Create the attribute list for Webbank EAR file

We have to combine the attribute list ds and the vh to create a combined attribute list before we invoke the install command. You can see the command to create the combined attribute list in Example 23-70.

Example 23-70 Creating combined attribute list and installing the application

```
wsadmin>set attrs [list -DataSourceFor20CMPBeans $ds -MapWebModToVH $vh -node
mka0kkwd -server webbankas -appname webbankApp]

wsadmin>$AdminApp install c:/new/Deployed_WebbankV5bis.ear $webbankattr
```

Regenerate Web server plug-in

Starting and stopping the enterprise application

23.7 Migration of WebSphere Application Server V4.0 wscp scripts to version V5

There is no straight migration process from WebSphere Application Server V4.0 to version 5.0. However there is equivalent tasks (both operational and configuration tasks) available in WebSphere Application Server V5.0. The following table maps the equivalent task between two versions.

Note to Reviewer: A table is available in the latest scripting document. That table needs to be added here.

Part 5

Appendixes

Note to Author: Optionally, describe the book part here. If you are not using Part files, you need to restart the page numbering in the first chapter file of your book:

- In FrameMaker 5.5: **Open .book > select first chapter> File > Set Up File > Page Numbering: > Restart at 1 > Set**
  - Now delete the three Part files (p01, p02 and p03) from your book: **Open .book > select a file > File > Rearrange Files > Delete > Done**
- In FrameMaker 6.0: **Open .book > select first chapter> Format > Document > Numbering > Page "tab" > select First Page # radio button and set Page # to 1 > Set**
  - Now delete the three Part files (p01, p02 and p03) from your book: **Open .book > select a part file > Edit > Delete File from Book**

In this part we introduce/provide/describe/discuss...

A

Command-line tools

In this appendix we provide/describe/discuss ...

A.1 startManager

Use

The command reads the configuration file for the deployment manager process and constructs a launch command for it. Depending on the options specified, the command launches a new Java Virtual Machine (JVM) to run the manager process, or writes the launch command data to a file.

Notes

1. The command must be run from the *bin* directory of the deployment manager installation root, eg. c:\ibm\was50\DeploymentManager\bin.
2. By default, the process logs to: <WASND_ROOT>/logs/dmgr/SystemOut.log and to *startServer.log* written to the directory in which the command is run.

Syntax

```
startManager.bat(sh) [options]

where all arguments are optional
```

Table 0-1 Options for startManager

Option	Description
-nowait	Tells the command not to wait for successful initialization of the launched deployment manager process.
-quiet	Suppresses progress information printed in normal mode.
-trace	Generates trace information into a file for debugging purposes.
-script [<script filename>]	Generate a launch script instead of starting the server. The script filename is optional. If the filename is not provided, the default script filename is start_<server>. The script needs to be saved to the *bin* directory of the node installation.
-timeout <seconds>	Waiting time before server initialization times out and returns an error.
-statusport <portnumber>	Set the port number for server status callback
-J-<java option>	Options to be passed through to the Java interpreter.
-help	Prints a usage statement to the console
-?	Prints a usage statement to the console

Examples

Example 0-1 startManager usage

```
$ cd c:\ibm\was50\DeploymentManager\bin
$ startManager.bat
$ startManager.bat -script (produces start_dmgr.bat or .sh)
$ startManager.bat -trace (produces startmanager.log file)
```

A.2 stopManager

Use

The command reads the configuration file for the deployment manager process. It sends a JMX command to the manager telling it to shutdown.

Notes

1. The command must be run from the *bin* directory of the deployment manager installation root, eg. c:\ibm\was50\DeploymentManager\bin.
2. By default, the command waits for the manager to complete shutdown before it returns control to the command line. There is a -nowait option to return immediately, as well as other options to control the behavior of the command.
3. By default, the command logs to <WASND_ROOT>/logs/dmgr/SystemOut.log and to *stopServer.log* written to the directory in which the command is run.

Syntax

```
stopManager.bat(sh) [options]

where all arguments are optional.
```

Table 0-2 Options for stopManager

Option	**Description**
-nowait	Tells the command not to wait for successful shutdown of the deployment manager process.
-quiet	Suppresses progress information printed in normal mode.
-trace	Generates trace information into a file for debugging purposes.

Option	Description
-script [<script filename>]	Generate a launch script instead of starting the server. The script filename is optional. If the filename is not provided, the default script filename is start_<server>. The script needs to be saved to the *bin* directory of the node installation.
-timeout <seconds>	Waiting time before server initialization times out and returns an error.
-statusport <portnumber>	Set the port number for server status callback
-conntype <type>	JMX connector to use for connection. Valid values are SOAP or RMI. If not specified, SOAP is assumed.
-host <hostname>	The deployment manager JMX hostname to use explicitly, so that configuration files do not have to be read to obtain the information.
-port <portnumber>	The deployment manager JMX port to use explicitly, so that configuration files do not have to be read to obtain the information.
-username <username>	Username for authentication if WebSphere security is enabled.
-password <password>	Password for authentication if WebSphere security is enabled.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-2 stopManager usage

```
$ cd c:\ibm\was50\DeploymentManager\bin
$ stopManager.bat
$ stopManager.bat -nowait
$ stopManager.bat -trace (produces stopmanager.log file)
$ stopManager.bat -conntype SOAP -host <deploymgr host> -port <deploymgr port>
```

A.3 startNode

Use

The command reads the configuration file for the node agent process, and constructs a launch command to run it. Depending on the options that are specified, the command creates a new Java Virtual Machine (JVM) to run the agent process, or writes the launch command data to a file.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin.
2. By default, the process logs to <WAS_ROOT>/logs/<node>/SystemOut.log and to *startServer.log* written to the directory in which the command is run.

Syntax

```
startNode.bat(sh) [options]

where all arguments are optional.
```

Table 0-3 Options for startNode

Option	**Description**
-nowait	Tells the command not to wait for successful initialization of the launched node agent process.
-quiet	Suppresses progress information.
-trace	Generates trace information into a file for debugging purposes.
-script [<script filename>]	Generate a launch script instead of starting the server. The script filename is optional. If the filename is not provided, the default script filename is start_<server>. The script needs to be saved to the *bin* directory of the node installation.
-timeout <seconds>	Waiting time before server initialization times out and returns an error.
-statusport <portnumber>	Set the port number for server status callback
-J-<java option>	Options to be passed through to the Java interpreter.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-3 startNode usage

```
$ cd c:\ibm\was50\AppServer\bin
$ startNode.bat
$ startNode.bat -script (produces start_node.bat or .sh)
$ startNode.bat -trace (produces startnode.log file)
```

A.4 stopNode

Use

The command reads the configuration file for the node agent process and sends a JMX command to the node agent telling it to shutdown.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin.
2. By default, the command waits for the node agent to complete shutdown before it returns control to the command line. There is a -nowait option to return immediately, as well as other options to control the behavior of the command.
3. By default, the command logs to <WAS_ROOT>/logs/<node>/SystemOut.log and to *stopServer.log* written to the directory in which the command is run.

Syntax

```
stopNode.bat(sh) [options]

where all arguments are optional.
```

Table 0-4 Options for stopNode

Option	Description
-nowait	Tells the command not to wait for successful shutdown of the node agent process.
-quiet	Suppresses progress information printed in normal mode.
-trace	Generates trace information into a file for debugging purposes.
-timeout <seconds>	Waiting time before server initialization times out and returns an error.
-statusport <portnumber>	Set the port number for server status callback

Option	Description
-conntype <type>	JMX connector to use for connection. Valid values are SOAP or RMI. If not specified, SOAP is assumed.
-host <hostname>	The node agent JMX hostname to use explicitly, so that configuration files do not have to be read to obtain the information.
-port <portnumber>	The node agent JMX port to use explicitly, so that configuration files do not have to be read to obtain the information.
-username <username>	Username for authentication if WebSphere security is enabled.
-password <password>	Password for authentication if WebSphere security is enabled.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-4 stopNode usage

```
$ cd c:\ibm\was50\AppServer\bin
$ stopNode.bat
$ stopNode.bat -nowait
$ stopNode.bat -trace (produces stopnode.log file)
```

A.5 addNode

Use

Adds a new node to an existing cell.

The actions performed by this command are:

1. Attempts to stop all running application servers on the standalone node.
2. Backs up the current standalone node configuration to:
   – config/backup/base - base cell configuration.
   – config/backup/apps - base installed application binaries.
3. Copies standalone node configuration to a new cell structure that matches the deployment manager structure at the cell level.
4. Delete the base adminconsole directory.

5. Creates a new local config directory and definition (*server.xml*) for the node agent.
6. Creates a separate *jmsserver* process containing the embedded messaging functionality. This is performed whether or not the base IBM WebSphere Application Server had embedded messaging installed.
7. Strips out the embedded messaging service from all application server definitions.
8. Adjusts the IP ports used for all EndPoints for all managed processes.
9. Calls the deployment manager to copy the documents from the new node to the cell repository.
10. Performs the first file synchronization for the new node. This pulls everything down from the cell to the new node.
11. Fixes the setupCmdLine script to reflect the new cell name.
12. Launches the node agent.
13. Creates a queue manager for the new node.

Important: The following points must be borne in mind when adding a node to a cell.

- The cell must already exist.
- The cell's deployment manager must be running before addNode can be executed.
- The new node must have a unique name. If an existing node already has the same name, addNode will fail.
- By default, addNode does not carry over the applications (installed on the standalone node) when added to the cell. Although the -includeApps option can be used, this is not the recommended approach since the application binaries will not be moved to the standard Network Deployment directory / location.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, e.g. c:\ibm\was50\AppServer\bin. It cannot be run from the deployment manager installation.
2. Depending on the size and location of the node being incorporated into the cell, the command can take a few minutes to complete.
3. The command logs to an *addNode.log* file written to the directory in which it is run.

Syntax

```
addNode.bat(sh) <deploymgr host> <deploymgr port> [options]
```

where the first two arguments are mandatory.

Author Comment: The port is obviously 8879 by default. Point to procedure where we might have changed it. The sentence would be .. “The default port for the deployment manager is 8879. If you changed this port (see .. ref) you will need to specify the new port. You can see the port in xxx.xml.”

Table 0-5 Options for addNode

Option	Description
-nowait	Tells the command not to wait for successful initialization of the launched node agent process.
-quiet	Suppresses progress information.
-trace	Generates trace information into a file for debugging purposes.
-conntype <type>	JMX connector to use for connection. Valid values are SOAP or RMI. If not specified, SOAP is assumed.
-username <username>	Username for authentication if WebSphere security is enabled.
-password <password>	Password for authentication if WebSphere security is enabled.
-includeapps	Attempt to include the applications in the incorporation of the base node into a cell. Default is not to include the applications.
-noagent	Indicates that the new node agent (generated as part of adding the node to a cell) is not be be started at the end. Default is to start the node agent.
-newtracefile	Start a new trace file for this operation.
-help	Prints a usage statement.
-?	Prints a usage statement.

Author Comment: How do I choose a connector? also .. the wsadmin script refers to these connectors as if there are three alternatives: jmx, rmi, soap. This makes it sound as if there are two: rmi, soap and both are jmx connectors.

Examples

Example 0-5 addNode usage

```
$ cd c:\ibm\was50\AppServer\bin
$ addNode.bat testhost 8879
$ addNode.bat deploymgr 8879 -trace (produces addNode.log file)
$ addNode.bat host25 8879 -nowait (does not wait for node agent process)
$ addNode.bat <deploymgr host> <deploymgr port> -conntype SOAP -includeApps
```

Author Comment: I tried `addNode.bat m23vnx86 8879 I got Soap connection errors. I can see in the server.xml that this is the correct port.`

A.6 removeNode

Use

Detaches a node from a cell and returns it to a base IBM WebSphere Application Server configuration.

The command performs the following operations:

1. Connects to the deployment manager process to read the configuration data.
2. Stops all of the running server processes of the node, including the node agent process.
3. Restores the backed up standalone node configuration - backed up when the node was originally added to the cell.

Note: This loses all configuration changes made to the managed servers since the node was originally added to the cell.

4. Copies the original application server cell configuration into the active configuration.
5. Removes the node's configuration from the master configuration repository of the cell.

> **Note:** The local copy of the configuration repository held on each node will get updated at the next synchronization point for each node agent. Although the complete set of configuration files are not pushed out to other nodes, some directories and files are pushed out to all nodes.

> **Tip:** To decouple an application server from distributed administration, while retaining its configuration (for example, applications), use the admin client to set the server's **`standalone`** property to **`true`**.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin. It cannot be run from the deployment manager installation.
2. Depending upon the size and location of the node, the command can take a few minutes to complete.
3. The command logs to an *removeNode.log* file written to the directory in which it is run.
4. If the **`-host`** and **`-port`** arguments are specified, the command contacts the deployment manager directly. Otherwise the deployment manager details are read from the local configuration repository.

Syntax

```
removeNode.bat(sh) [options]

where all the arguments are optional.
```

Table 0-6 Options for removeNode

Option	Description
-quiet	Suppresses progress information printed in normal mode.
-trace	Generates trace information into a file for debugging purposes.

Option	Description
-conntype <type>	The Java Management Extension (JMX) connector type to use to connect to the deployment manager. Valid types are SOAP and RMI. If not specified, SOAP is assumed.
-host <dep mgr host>	Hostname of the deployment manager machine.
-port <dep mgr port>	Port number of deployment manager process.
-username <username>	Username for authentication if WebSphere security is enabled.
-password <password>	Password for authentication if WebSphere security is enabled.
-force	Cleans up the local node configuration regardless of whether the deployment manager can be reached for cell repository cleanup.
-newtracefile	Start a new trace file for this operation.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-6 removeNode usage

```
$ cd c:\ibm\was50\AppServer\bin
$ removeNode.bat -trace
$ removeNode.bat -conntype SOAP -quiet
```

A.7 syncNode

Use

Force the synchronization of a node’s local configuration repository with the master cell configuration repository on the deployment manager node.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin. It cannot be run from the deployment manager installation.
2. The command logs to an *syncNode.log* file written to the directory in which it is run.

Syntax

```
syncNode.bat(sh) <deploymgr host> <deploymgr port> [options]

where the first two arguments are mandatory.
```

Table 0-7 Options for syncNode

Option	Description
-nowait	Tells the command not to wait for successful initialization of the launched node agent process.
-quiet	Suppresses progress information printed in normal mode.
-trace	Generates trace information into a file for debugging purposes.
-conntype <type>	JMX connector type to use for connection to the deployment manager. Valid values are SOAP or RMI. If not specified, SOAP is assumed.
-stopservers	Indicates that the node agent and all managed servers of the node should be stopped prior to synchronizing the node’s configuration with the cell.
-restart	Indicates that the node agent is to be restarted after synchronizing the node’s configuration with the cell.
-username <username>	Username for authentication if WebSphere security is enabled.
-password <password>	Password for authentication if WebSphere security is enabled.
-newtracefile	Start a new trace file for this operation, rather than appending to the existing trace file.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-7 syncNode usage

```
$ cd c:\ibm\was50\AppServer\bin
$ syncNode.bat <deploymgr host> 8879
$ syncNode.bat <deploymgr host> 8879 -trace
$ syncNode.bat <deploymgr host> 4444 -stopservers -restart
$ syncNode.bat <deploymgr host> <deploymgr port> -conntype SOAP
```

A.8 cleanupNode

Use

Cleans up a node configuration from a partially created network deployment distributed administration cell.

> **Important:** Only use this command if a partial **`addNode`** command or **`removeNode`** command fails to remove all configuration data from the administration configuration

Notes

1. The command must be run from the *bin* directory of the deployment manager installation root on the node machine, eg. c:\ibm\was50\DeploymentManager\bin. It cannot be run from the node installation.
2. The command logs all output to the *cleanupNode.log* file written to the same directory in which the command is run.
3. The command does not provide a communication type argument. SOAP is assumed.

Syntax

```
cleanupNode.bat(sh) <node name> <deploymgr host> <deploymgr port> [options]

where the first argument is mandatory.
```

Table 0-8 Options for cleanupNode

Option	Description
-quiet	Suppresses progress information that the command prints in normal mode.
-trace	Generates trace information into a file for debugging purposes.
-username <username>	Username for authentication if WebSphere security is enabled.
-password <password>	Password for authentication if WebSphere security is enabled.
-newtracefile	Start a new trace file for this operation.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-8 cleanupNode usage

```
$ cd c:\ibm\was50\AppServer\bin
$ cleanupNode.bat mynode <deploymgr host> <deploymgr port>
$ cleanupNode.bat mynode -quiet
```

A.9 startServer

Use

The command reads the configuration file for the specified server process and constructs a launch command for the server. Depending on the options you specify, a new Java Virtual Machine (JVM) can be launched to run the server process or to write the launch command data to a file.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin. It cannot be run from the deployment manager installation.
2. By default, the process logs to: <WAS_ROOT>/logs/<server>/SystemOut.log. In addition, each invocation of the command logs to a *startServer.log* file (after deleting any existing file of that name) in the directory in which the command is run.

Syntax

```
startServer.bat(sh) <server> [options]

where <server> is the name of the server to be started. The argument is
required.
```

Table 0-9 Options for startServer

Option	Description
-nowait	Tells command not to wait for successful initialization of the launched server process.
-quiet	Suppresses progress information.
-trace	Generates trace information into a file for debugging purposes.

Option	Description
-script [<script filename>]	Generate a launch script instead of starting the server. The script filename is optional. If the filename is not provided, the default script filename is start_<server>. The script needs to be saved to the *bin* directory of the node installation.
-timeout <seconds>	Waiting time before server initialization times out and returns an error.
-statusport <portnumber>	Set the port number for server status callback
-J-<java option>	Options to be passed through to the Java interpreter.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-9 startServer usage

```
$ cd c:\ibm\was50\AppServer\bin
$ startServer.bat server1
$ startServer.bat server1 -script (produces start_server1.bat or .sh)
$ startServer.bat server1 -trace (produces startserver.log file)
```

A.10 stopServer

Use

The command reads the configuration file for the specified server process. It sends a JMX command to the server telling it to shutdown.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin. It cannot be run from the deployment manager installation.
2. By default, the command does not return control to the command line until the server completes shutting down. There is a -nowait option to return immediately, as well as other options to control the behavior of the stopServer utility
3. By default, the process logs to: <WAS_ROOT>/logs/<server>/SystemOut.log. In addition, each invocation of the command logs to a *stopServer.log* file (after

deleting any existing file of that name) in the directory in which the command is run.

Syntax

```
stopServer.bat(sh) <server> [options]

where <server> is the name of the server to be stopped. The argument is
required.
```

Table 0-10 Options for stopServer

Option	Description
-nowait	Tells the command not to wait for successful shutdown of the server process.
-quiet	Suppresses progress information rpinted in normal mode.
-trace	Generates trace information into a file for debugging purposes.
-timeout <seconds>	Waiting time before server initialization times out and returns an error.
-statusport <portnumber>	Set the port number for server status callback.
-conntype <connector type>	Type of JMX connector to use for connection to the deployment manager. Valid values are SOAP or RMI. If not specified, SOAP is assumed.
-host <hostname>	The server JMX hostname to use explicitly, so that configuration files do not have to be read to obtain the information.
-port <portnumber>	The server JMX port to use explicitly, so that configuration files do not have to be read to obtain the information.
-username <username>	Username required for authentication if WebSphere security is enabled.
-password <password>	Password required for authentication if WebSphere security is enabled.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-10 stopServer usage

```
$ cd c:\ibm\was50\AppServer\bin
$ stopServer.bat server1
$ stopServer.bat server1 -nowait
$ stopServer.bat server1 -trace (produces stopserver.log file)
$ stopServer.bat myserver -conntype SOAP -host <server host> -port <server SOAP
port>
```

A.11 serverStatus

Use

Obtains the status of one or all of the servers configured on a node.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin. It cannot be run from the deployment manager installation.
2. The command logs all output to the *serverStatus.log* file written to the same directory in which the command is run.

Syntax

```
serverStatus.bat(sh) <server>|-all [options]
```

```
The first argument is required. The argument is either the name of the server
for which status is desired, or the -all keyword, which requests status for all
servers defined on the node.
```

Table 0-11 Options for serverStatus

Option	Description
-trace	Generates trace information into a file for debugging purposes.
-user <username>	Username required for authentication if WebSphere security is enabled.
-password <password>	Password required for authentication if WebSphere security is enabled.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-11 serverStatus usage

```
$ cd c:\ibm\was50\AppServer\bin
$ serverStatus.bat server1
$ serverStatus.bat -all (returns status for all defined servers in node)
$ serverStatus.bat -trace (produces serverStatus.log file)
```

A.12 createmq

Use

Creates the messaging broker, Queue Manager and supporting messaging objects for a node.

The actions performed by this command are:

1. Uses the MQSeries **`crtmqm`** command to create on the local machine a new MQSeries Queue Manager called the following:

   WAS_<cell name>_<node name>

2. Uses the MQSeries **`strmqm`** command to start the new MQSeries Queue Manager.
3. Uses the MQSeries **`runmqsc`** command and the MQSeries script located at

   <WAS_ROOT>\bin\createmq.mqsc

   to perform the following adjustments to the Queue Manager:

   a. Creates new queues to support JMS point-to-point and publish-subscribe messaging.

   b. Creates new channel definition.

   c. Deletes all standard channels.

   d. Deletes all default queues.
4. Uses the **`wempsdeletebroker`** command to delete any existing broker of the following name on the local machine:

   WAS_<cell name>_<node name>

5. Uses the **`wempscreatebroker`** command to create a new broker of the following name and associate it with the Queue Manager:

   WAS_<cell name>_<node name>

6. Registers the broker as a Windows service (on Windows only).
7. Uses the MQSeries **`endmqm`** command to stop the new Queue Manager.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin. It cannot be run from the deployment manager installation.
2. The command must be run after the installation of WebSphere MQ, the broker and the JMS client (MA88) classes. The command assumes the paths have been configured correctly.
3. The command logs all output to a *createmq.<servername>.log* file written to the directory in which the command is run.
4. The **`createmq`** command script is specific to each platform, but createmq.mqsc is common to all platforms.

Syntax

```
createmq.bat(sh) <WAS root> <Cell name> <Node name> <Server name> <MQSeries
root> <Broker root>

where all arguments are mandatory.
```

Table 0-12 Options for createmq

Option	Description
WAS root	The root WebSphere Application Server installation directory of the node, eg. c:\ibm\was50\AppServer
Cell name	Name of the cell, eg. NetworkDeploymentCell
Node name	Name of the local node.
Server name	Name of the managed server in which to install and configure embedded messaging. For Network Deployment configurations, the server on each node is always *jmsserver*.
MQSeries root	Root directory of the local WebSphere MQ (MQSeries) installation, eg. C:\Program Files\IBM\MQSeries
Broker root	Root directory of the WebSphere MQ (MQSeries) publish-subscribe broker software, eg C:\Program Files\IBM\WebSphere Embedded Messaging Publish And Subscribe\

Examples

Example 0-12 `createmq usage`

```
$ cd c:\ibm\was50\AppServer\bin
$ createmq.bat “c:\ibm\was50\AppServer" NetworkDeploymentCell Net1_JH jmsserver
"C:\Program Files\IBM\MQSeries" "C:\Program Files\IBM\WebSphere Embedded
Messaging Publish And Subscribe\"
```

A.13 deletemq

Use

Deletes the messaging broker, Queue Manager and supporting messaging objects for a node.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin. It cannot be run from the deployment manager installation.
2. The command does not uninstall the MQSeries or broker products.
3. The command logs all output to a *deletemq.<servername>.log* file written to the directory in which the command is run.

Syntax

```
deletemq.bat(sh) <Cell name> <Node name> <Server name>

where all arguments are mandatory.
```

Table 0-13 Options for deletemq

Option	Description
Cell name	Name of the cell.
Node name	Name of the local node.
Server name	Name of the managed server from which embedded messaging is to be removed. For Network Deployment configurations, the server on each node is always *jmsserver*.

Examples

Example 0-13 deletemq usage

```
$ cd c:\ibm\was50\AppServer\bin
$ deletemq.bat NetworkDeploymentCell Net1_JH jmsserver
```

A.14 backupConfig

Use

Backup the configuration of the node to a ZIP archive file.

The actions performed by this command are:

1. Stops all running processes (node agent, application servers and JMS Server) of the local node.
2. Archives the contents of the <WAS_ROOT>/config directory (and subdirectories) to a named ZIP file, or *WebSphereConfig_<yyyy-mm-dd>.zip*, created in the same directory in which the command is run.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root of the local node machine to be backed up, eg. c:\ibm\was50\AppServer\bin. The command does not support backup of remote nodes from a central location, eg. the deployment manager machine.
2. If specified, the backup archive filename does not require the zip file extension. The command will create an archive named <filename>.zip.
3. The command logs all output to the *backupConfig.log* file written to the same directory in which the command is run.
4. The command does not restart the processes that were stopped (if any) to perform the backup. These processes will require a manual restart by the administrator.
5. The master cell configuration repository can be backed up by running **`backupConfig`** on the machine hosting the deployment manager.

Syntax

```
backupConfig.bat(sh) <backup_file> [options]

where <backup_file> specifies the name of the ZIP file to which the backup is
written. If not specified, a unique filename is used. All arguments are
optional.
```

Table 0-14 Options for backupConfig

Option	**Description**
-nostop	Do not stop the servers before backing up the configuration.
-quiet	Suppresses the progress information printed in normal mode.

Option	Description
-logfile <filename>	Location of the log file to which information gets written.
-replacelog	Replaces the log file instead of appending to the current log.
-trace	Generates trace information into a file for debugging purposes.
-user <username>	Username required for authentication if WebSphere security is enabled.
-password <password>	Password required for authentication if WebSphere security is enabled.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-14 backupConfig usage

```
$ cd c:\ibm\was50\AppServer\bin
$ backupConfig.bat name
$ backupConfig.bat name -nostop
```

A.15 restoreConfig

Use

Restores the configuration of the node after backing up the configuration using the backupConfig command.

The actions performed by this command are:

1. Stops all running processes (node agent, application servers and JMS Server) of the local node. This prevents a node synchronization from occuring during the install.
2. If the <WAS_ROOT>/config configuration directory already exists, it is renamed to *config.old*.
3. Restores the contents of the archive to <WAS_ROOT>/config on the local node.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin. It cannot be run from the deployment manager installation.
2. The command logs all output to the *restoreConfig.log* file written to the same directory in which the command is run.
3. The command does not restart the processes that were stopped (if any) to perform the restore. These processes will require a manual restart by the administrator.

Syntax

```
restoreConfig.bat(sh) <archive_file> [options]

where <archive_file> specifies the backup archive to be restored. The argument
is mandatory.
```

Table 0-15 Options for restoreConfig

Option	Description
-nostop	Do not stop the servers before restoring the configuration.
-quiet	Suppresses the progress information printed in normal mode.
-logfile <filename>	Location of the log file to which information gets written.
-replacelog	Replaces the log file instead of appending to the current log.
-trace	Generates trace information into a file for debugging purposes.
-user <username>	Username required for authentication if WebSphere security is enabled.
-password <password>	Password required for authentication if WebSphere security is enabled.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-15 restoreConfig usage

```
$ cd c:\ibm\was50\AppServer\bin
$ restoreConfig.bat name
$ restoreConfig.bat name -nostop
```

A.16 instance

Use

Creates a new and independent instance of the IBM WebSphere Application Server base installation on the same machine.

The actions performed by this command are:

1. Loads the *instance.xml* build file from the same directory in which the command is run. This file contains a definition of all the components to be copied or generated for the new instance.
2. The following directories are generated for the new instance and files are copied from the original instance. The only files required are those that distinguish the new instance from the original.
   - bin

     Contains a new *setupCmdLine.bat(sh)* to configure the new instance's environment.
   - config

     Complete copy of the original instance's *config* directory (and subdirectories), but with new node and server1 (default server).
   - etc

     Complete copy of the original instance's *etc* directory.
   - installableApps
   - installedApps
   - logs
   - properties

     Complete copy of the original instance's *properties* directory.
   - temp
   - translog
   - wstemp
3. Creates the *configinst.props* file in the same directory as where the command is run. This file contains the path to the newly created instance directory.

4. Creates the <hostname>_<instancename>_portdef.props file in the same directory as where the command is run. This file contains the list of port numbers assigned for the new instance. For example,

```
BOOTSTRAP_ADDRESS=2810
SOAP_CONNECTOR_ADDRESS=8881
DRS_CLIENT_ADDRESS=7874
INTERNAL_JMS_SERVER=5560
JMSSERVER_QUEUED_ADDRESS=5561
JMSSERVER_DIRECT_ADDRESS=5562
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=0
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=0
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=0
HTTP_TRANSPORT=9081
HTTP_TRANSPORT_ADMIN=9091
HTTPS_TRANSPORT=9444
HTTPS_TRANSPORT_ADMIN=9044
```

5. Creates a new Queue Manager and supporting objects for the embedded messaging of the new instance. Uses **createmq** and logs all output to *createMQ_<hostname>_<instancename>.txt*.

Notes

1. The command must be run from the *bin* directory of the IBM WebSphere Application Server installation root on the node machine, eg. c:\ibm\was50\AppServer\bin. It cannot be run from the deployment manager installation.
2. The command logs all output to the *WebSphereConfig_<yyyy-mm-dd>.log* file written to the same directory in which the command is run.
3. Applications installed on the base installation are not carried over into the new instance. This includes the Web administration console application.
4. **addNode** must still be used to add the new instance to an existing Network Deployment cell.
5. Commands and actions for the new instance are invoked by:
   a. Changing directory to the new instance's *bin* directory.
   b. Invoking the required command, eg. **startserver**. This invokes the command from the original installation's *bin* directory, but uses the new instance's *setupCmdLine* script to configure the environment correctly for the new instance.

Syntax

```
instance.bat(sh) -name instanceName -path instancePath -host hostName
[-startingPort startingPort] -create|-delete  [-debug]
```

Table 0-16 Options for instance

Option	Description
-name <instancename>	Unique name for this instance on this host.
-path <instancepath>	Path under which the new instance is to be created. Takes the places of <WAS_ROOT>. The specified directory will contains a bin, config, properties etc directories.
-host <hostname>	Hostname of the machine. Used to encode a unique node name for the instance, as <hostname>_<instancename>
-startingPort <port>	Port number to use as a basis for the port numbers assigned to the new instance.
-create | delete	Determines whether to create a new base instance or delete an existing instance.
-debug	Generates debugging information.
-help	Prints a usage statement.
-?	Prints a usage statement.

Examples

Example 0-16 instance usage

```
$ cd c:\ibm\was50\AppServer\bin
$ instance.bat -name inst2 -path “c:\ibm\was50\inst2” -host myhost -create
```

A

Installing the infocenter

Note to Author: Describe the appendix contents here using these or similar words. Optionally add level 2 headings to a list using:
Special > Cross-Reference > Format: Head > Insert

This appendix provides/describes/discusses/contains ...

In this appendix we provide/describe/discuss ...

In this appendix:
In this appendix, the following are described:
This appendix provides/describes/discusses/contains the following:

- ...
- ...
- ...
- Sample level 2 appendix heading
  (created by **Special > Cross-Reference > Format: Head > Insert**)
- Sample next level 2 appendix heading

Sample level 2 heading (yHead1Appendix), new page

Note to Author: The first level 2 heading in an appendix should be the (yHead1Appendix) tag, skip to new page.

Add text here (Body0).

Sample level 2 heading (yHead2Appendix)

Add text here (Body0).

Sample level 3 heading (yHead3Appendix)

Add text here (Body0).

Sample level 4 heading (yHead4Appendix)

Add text here (Body0).

Sample level 5 heading (yHead5Appendix)

Add text here (Body0).

A

Port usage

Note to Author: Describe the appendix contents here using these or similar words. Optionally add level 2 headings to a list using:
Special > Cross-Reference > Format: Head > Insert

This appendix provides/describes/discusses/contains ...

In this appendix we provide/describe/discuss ...

In this appendix:
In this appendix, the following are described:
This appendix provides/describes/discusses/contains the following:

- ...
- ...
- ...
- Sample level 2 appendix heading
  (created by **Special > Cross-Reference > Format: Head > Insert**)
- Sample next level 2 appendix heading

Sample level 2 heading (yHead1Appendix), new page

Table 0-1 Default ports & settings

Port	Process	where
ND: serverindex.xml		
7277	CELL_DISCOVERY_ADDRESS	
9809	BOOTSTRAP_ADDRESS	
7989	DRS_CLIENT_ADDRESS	
8879	SOAP_CONNECTOR_ADDRESS	
9100	ORB_LISTENER_ADDRES	
9401	SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	
9402	CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	
9403	CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	
Base: serverindex.xml		
2809	BOOTSTRAP_ADDRESS	
8880	SOAP_CONNECTOR_ADDRESS	
7873	DRS_CLIENT_ADDRESS	
5558	JMSSERVER_QUEUED_ADDRESS	
5559	JMSSERVER_DIRECT_ADDRESS	
0	SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	
0	CSIV2_SSL_SERVERAUTH_LISTENER_ADDRES	
0	CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRES	
server.xml		
9080	HTTPTransport_1	
9443	HTTPTransport_2	
9090	HTTPTransport_3	
9043	HTTPTransport_4	

Port	Process	where

-

A

Setting admin preferences in XML

Note to Author: Describe the appendix contents here using these or similar words. Optionally add level 2 headings to a list using:
Special > Cross-Reference > Format: Head > Insert

This appendix provides/describes/discusses/contains ...

In this appendix we provide/describe/discuss ...

In this appendix:
In this appendix, the following are described:
This appendix provides/describes/discusses/contains the following:

- ...
- ...
- ...
- Sample level 2 appendix heading
  (created by **Special > Cross-Reference > Format: Head > Insert**)
- Sample next level 2 appendix heading

Sample level 2 heading (yHead1Appendix), new page

Note to Author: The first level 2 heading in an appendix should be the (yHead1Appendix) tag, skip to new page.

Add text here (Body0).

Sample level 2 heading (yHead2Appendix)

Add text here (Body0).

Sample level 3 heading (yHead3Appendix)

Add text here (Body0).

Sample level 4 heading (yHead4Appendix)

Add text here (Body0).

Sample level 5 heading (yHead5Appendix)

Add text here (Body0).

A

Quick start for Windows users

Step 1: Install the base

- Install the base (did not elect to run WAS as a service)
- Start the application server using First Steps or c:\WAS_install\AppServer\bin\startServer server1
- Check the status with c:\WAS_home\AppServer\bin\serverStatus -all. You should see one server called server1.
- Verify the installation using First Steps
- Problems? Check the log.txt

Step 1a: Check the base installation

- Start the admin console using First Steps or http://hostname:9090/admin
- Enter an ID you would like to use to track your changes.
- Applications>Enterprise applications (you will see the adminconsole app, + any samples you installed. Add some other checks .. how to see status of servers, etc.
- Try the samples:
  - Open a new web browser and test Plants By WebSphere: http://<hostname>:9080/PlantsByWebSphere, where <hostname> is the name you provided for your hostname during installation.

What you have .. An application server with the sample applications installed (if you selected them during the install).

- View the context root of a web application using the Plants By WebSphere Example
  - The context root for each sample is in the application.xml deployment descriptor: config/cells/*host_name*/application/*sample_name*.ear/deployments/*sample_name*/META-INF/application.xml.
  - To view the context root from the Admin Console, select “Plants by WebSphere” in the Enterprise Application listing. This will display the configuration and local toplogy for that application.
  - In the “Configuration” tab, locate “View Deployment Descriptor”. The application deployment descriptor, application.xml, contains a list of the context roots used by this application.
  - Select “View Deployment Descriptor”, and a list of Web modules, EJB Modules, and Application Client Modules will be displayed.

   - Expland “Web Modules”. For the Plants By WebSphere Application there are many web applications installed. To see the contect root for each web application
   - Expand the first web module in the list “Web Module URI: PlantsByWebSphere.war”. This will display the context root for that web module.
- To find the port a web module is using, select Enterprise Applications, select an application, select “Map virtual hosts for Web modules” under the Configuration tab. You will see the virtual host name (most likey “default_host”).
   - From the topology on the left of the screen, select Environment>Virtual Hosts. Select “default_host”, then under “Additional Properties” select “Host Aliases”. The host alias specifies the IP address, DNS host name with domain name suffix, or just the DNS host name, used by a client to request a Web application resource (such as a servlet, JSP file, or HTML page). An “ * “ identifies any allowed value or no value is specified. The “ * “ ususally defaults to values specified during installation.

In Summary, you should now understand how to determine the hostname, port number and context root of a web application in the Administrative Console. You can use this information to launch web applications from a web browser.

Step 1b: Stop the server

- c:\was_home\AppServer\bin\stopServer -server1

Step 2: Install ND

- Install ND (note the node name - sadtlerManager, and cell name - sadtlerNetwork, during the install).
- c:\websphere\deploymentmanager\bin\startManager (starts dmgr)
- c:\websphere\appserver\bin\addNode localhost 8879

> **Note:** If ND and base are on the same machine, you must specify “localhost” as the host name. Specifying it by its real host name will give errors on the addNode (Soap connection errors .. port not open, etc).

- open the admin console (note the differences)
- System Administration > Nodes (to see new node, synch status)

Step 3: Install Webbank

Create the database

1. Start a command prompt, and start a DB2 command window:

   ```
   db2cmd.
   ```

   Perform all actions below from this DB2 command window.

2. Create, initialize & populate the Webbank database (execute the Buildwebbank.cmd):

   ```
   DB2 CREATE DB WEBBANK
   DB2 CONNECT TO webbank user wasadmin using wasadmin

   DB2 CREATE SCHEMA ISABELLE
   DB2 CREATE TABLE ISABELLE.BRANCHACCOUNT (BRANCHID VARCHAR(8) NOT
   NULL,BRANCHNAME VARCHAR(30), BRANCHADDRESS VARCHAR(30), BALANCECENT
   INTEGER)

   DB2 ALTER TABLE ISABELLE.BRANCHACCOUNT ADD CONSTRAINT BRANCHACCOUNTPK
   PRIMARY KEY (BRANCHID)

   DB2 CREATE TABLE ISABELLE.CUSTOMERACCOUNT (ACCOUNTTYPE CHARACTER(1),
   CUSTOMERID VARCHAR(8) NOT NULL, CUSTOMERNAME VARCHAR(40), ACCOUNTNUMBER
   VARCHAR(2) NOT NULL,  BALANCECENT INTEGER)

   DB2 ALTER TABLE ISABELLE.CUSTOMERACCOUNT  ADD CONSTRAINT CUSTOMERACCOUNTPK
   PRIMARY KEY (CUSTOMERID, ACCOUNTNUMBER)

   DB2 insert into isabelle.customeraccount
   (ACCOUNTTYPE,CUSTOMERID,CUSTOMERNAME,ACCOUNTNUMBER,BALANCECENT) values
   ('C','Isabelle','Customer #Isabelle','A1',1000.0)
   DB2 insert into isabelle.branchaccount
   (BRANCHID,BRANCHNAME,BRANCHADDRESS,BALANCECENT) values
   ('Sophia','Unknown','Branch #Sophia',100000.0)
   ```

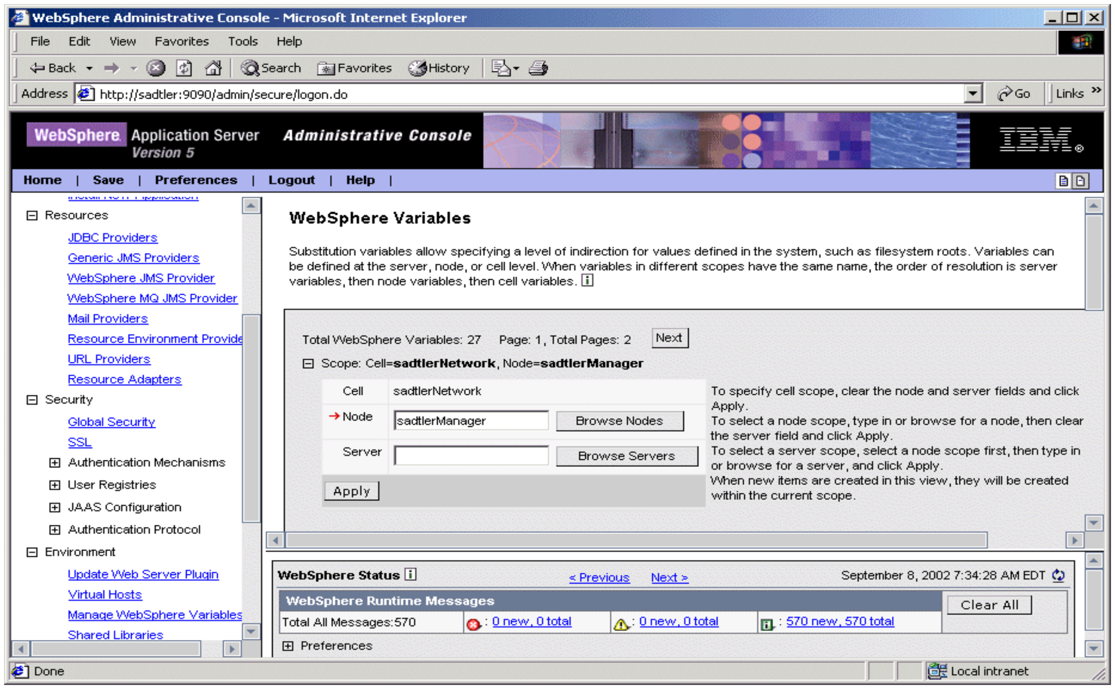
3. Use the DB2 control center to grant All access to the user ID you want the application to use to access the db. (The user ID must be a valid user ID on your local system or Windows domain.

Add variables

Using the WebSphere administration console, add two variables at the node level:

1. Open the administration console from a Web browser by specifying the url http://*host_name*:9090/admin.

2. Select **Environment > Manage WebSphere Variables**
3. Ensure the scope is set to your node.



WebSphere Administrative Console - Microsoft Internet Explorer

Address http://sadtler:9090/admin/secure/logon.do

WebSphere Application Server Version 5 Administrative Console

Home | Save | Preferences | Logout | Help |

Resources
- JDBC Providers
- Generic JMS Providers
- WebSphere JMS Provider
- WebSphere MQ JMS Provider
- Mail Providers
- Resource Environment Provide
- URL Providers
- Resource Adapters

Security
- Global Security
- SSL
- Authentication Mechanisms
- User Registries
- JAAS Configuration
- Authentication Protocol

Environment
- Update Web Server Plugin
- Virtual Hosts
- Manage WebSphere Variables
- Shared Libraries

WebSphere Variables

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

Total WebSphere Variables: 27 Page: 1, Total Pages: 2 Next

Scope: Cell=**sadtlerNetwork**, Node=**sadtlerManager**

Cell	sadtlerNetwork	
Node	sadtlerManager	Browse Nodes
Server		Browse Servers
Apply		

To specify cell scope, clear the node and server fields and click Apply.
To select a node scope, type in or browse for a node, then clear the server field and click Apply.
To select a server scope, select a node scope first, then type in or browse for a server, and click Apply.
When new items are created in this view, they will be created within the current scope.

WebSphere Status < Previous Next > September 8, 2002 7:34:28 AM EDT

WebSphere Runtime Messages Clear All

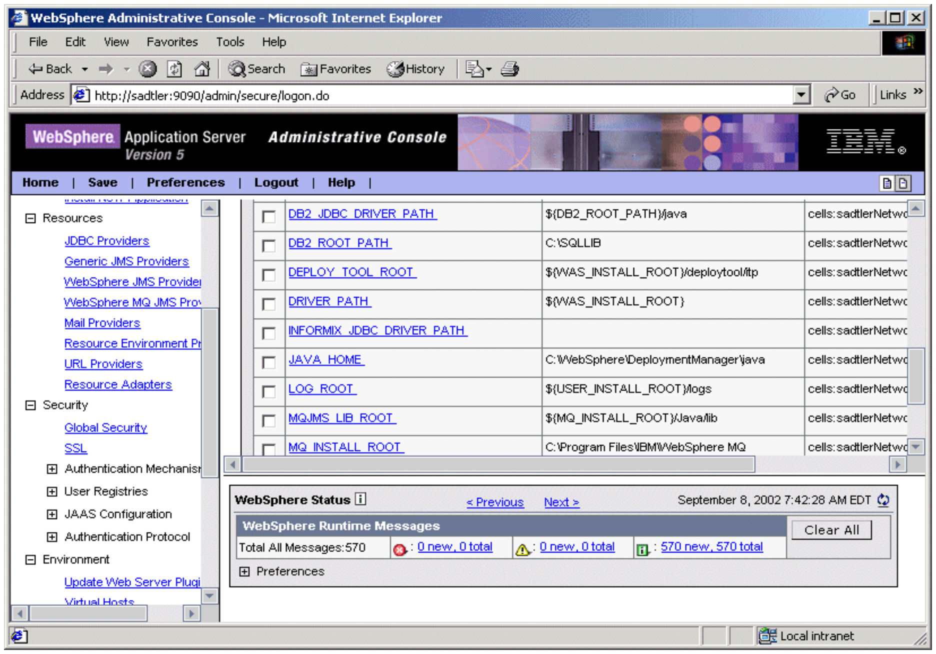
Total All Messages:570 : 0 new, 0 total : 0 new, 0 total : 570 new, 570 total

Preferences

Done Local intranet

Figure 0-1

4. Click **New.**
   a. Name = DB2_ROOT_PATH
   b. Value= <*fully qualified path to DB2 SQLLIB directory*>, for example c:\sqllib
5. Select DB2_JDBC_PROVIDER_PATH (this is created during install but not populated with a value.
   a. Name=DB2_JDBC_PROVIDER_PATH
   b. Value= ${DB2_ROOT_PATH}/java



☐	DB2_JDBC_DRIVER_PATH	${DB2_ROOT_PATH}/java	cells:sadtlerNetwo
☐	DB2_ROOT_PATH	C:\SQLLIB	cells:sadtlerNetwo
☐	DEPLOY_TOOL_ROOT	${WAS_INSTALL_ROOT}/deploytool/itp	cells:sadtlerNetwo
☐	DRIVER_PATH	${WAS_INSTALL_ROOT}	cells:sadtlerNetwo
☐	INFORMIX_JDBC_DRIVER_PATH		cells:sadtlerNetwo
☐	JAVA_HOME	C:\WebSphere\DeploymentManager\java	cells:sadtlerNetwo
☐	LOG_ROOT	${USER_INSTALL_ROOT}/logs	cells:sadtlerNetwo
☐	MQJMS_LIB_ROOT	${MQ_INSTALL_ROOT}/Java/lib	cells:sadtlerNetwo
☐	MQ_INSTALL_ROOT	C:\Program Files\IBM\WebSphere MQ	cells:sadtlerNetwo

WebSphere Status   < Previous   Next >   September 8, 2002 7:42:28 AM EDT

WebSphere Runtime Messages   Clear All

Total All Messages:570 | : 0 new, 0 total | : 0 new, 0 total | : 570 new, 570 total

Preferences

Figure 0-2

Configure the JDBC provider and datasource

Create a DB2 (non-XA) JDBC Provider:

1. **Resources>JDBC Providers**
2. Ensure the scope is set to your node.
3. Click **New.**
   a. JDBC Providers=DB2 JDBC Provider
   b. Click **OK**.
   c. class path = ${DB2_JDBC_DRIVER_PATH}/db2java.zip
   d. native path =

      ${DB2_ROOT_PATH}/bin

      ${DB2_ROOT_PATH}/function
   e. Implementation class name= COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource
4. Click **Apply**.
5. Select **DataSources**.

6. Click **New** to add a new CMP 2.0 DataSource with the following attributes:
   a. name = webbankds
   b. JNDI name = webbankds
   c. Use this datasource in CMP = checked
   d. Click on **Custom Properties**
   e. Select **databaseName** and enter WEBBANK as the value. Click **OK**.
   f. Click **New** and enter the following:
      i. Name=user
      ii. Value= the userid you granted database access to in step xxx
   g. Click **OK**.
   h. Click **New** and enter the following:
      i. Name=password
      ii. Value= the password for the userid you specified in the user property.
   i. Click **OK**.

Install the application

1. Install webbank.ear using **Applications > Install New Application**
   a. Browse to Deployed_WebbankV5.ear and select it. Click **Next**.
   b. Specify the following (all are the default):
      - Uncheck Generate default bindings
      - Prefixes=Do not specify unique prefix for beans
      - Override=Do not override existing bindings
      - EJB CMP 1.1 bindings=Do not default bindings for EJB 1.1 CMPs
      - Connection factory bindings=Do not default connection factory bindings
      - Virtual host=Default virtual host name for Web modules = default_host
   c. Click **Next**
   d. **Step 1: Provide options to perform the installation**.
      - Check Pre-compile JSP
      - Leave the directory to install the application blank. This installs application under the default location <WASROOT>/installedApps (in WAS5.0 base).
      - Check Distribute application
      - Uncheck Use Binary Configuration. This says to use the deployed configuration for WAS 5.0 that results from the settings we make during installation, e.g. resource bindings and are stored in the configuration repository. Selecting this option would tell WebSphere to instead use the raw deployment descriptors in the EAR - not what we want.
      - Uncheck Deploy EJBs. The EAR already has deployed EJB code, so no need to generated deployed code during install process.

- Application name = WebbankV5. The application name can be set to anything. It doesn't have to remain the same as the source EAR filename (minus the .ear extension). However there can't already be an application installed of this name. Every application must have unique name.
- Check Create MBeans for Resources (default)
- Reload interval - leave blank
- Click **Next**.

e. **Step 2 : Provide JNDI Names for Beans**

   The JNDI names are taken from the deployment descriptor and should be correct.

   - Click Next.

f. **Step 3: Provide default datasource mapping for modules containing 2.0 entity beans**

   - Put a checkmark next to the ‘webbankEntityEJBs’ EJB Module .
   - Expand Apply Multiple Mappings.
   - Specify existing JNDI resource name = eis/webbankds_CMP.
   - Click the Apply button next to the Resource JNDI name field.
   - Make sure that Specify Resource authorization = container for the module (default).
   - Click **Next.**

g. Step 4: Map datasources for all CMP 2.0 data beans

   - Ensure that each individual entity EJB uses the eis/webbankds_CMP datasource (using the same approach as in step 3 above).
   - Resource Authorization should be Per connection factory (default).
   - Click **Next**.

h. **Step 5: Map EJB References to beans**.

   - These mappings should be OK so click **Next**.

i. **Step 6 : Map virtual hosts for web modules**

   - Make sure that default_host is specified for the Web modules (default)
   - Click **Next**.

j. **Step 7 : Map modules to application servers**

   - We only have one server, server1, so we can take the defaults on this page.
   - Click **Next**.

k. **Step 8 : Ensure all unprotected 2.0 methods have the correct level of protection**

   - We aren’t using security at this point, so select “uncheck”.

- Click **Next**.

l. **Step 9: Summary**

- Click **Finish**.

The application will install and messages are displayed on the console.

m. Check configuration errors (see link at bottom of console). I have 6 errors that look like some kind of environment port/address errors.

n. If the installation is successful, click on **Save to Master Configuration**

o. Restart the application server. Servers>Application Servers. Check the box to the left of server1 and click Start.

Bring-up sequence (after a reboot, etc)

Web Server

Deployment Manager

- ► cd C:\websphere\deploymentmanager\bin
- ► startManager

Base

- ► cd C:\websphere\appserver\bin
- ► startNode
- ► startServer server1
- ► startServer jmsserver

Author Comment: When do I need to start jmsserver?

Shutdown sequence

Installing WebSphere MQ

WebSphere MQ can be installed with Network deployment or at a later time. This section will discuss the procedure for installing webSphere MQ after the installation.

A

Webbank Application Overview

To illustrate how to work with the WebSphere Application Server, we use an application called Webbank. This application, which was originally developed for teaching purposes, lets you make transfers between a branch and a customer account and check accounts balance. It is a full J2EE 1.3 application, using servlets, JSP, and entreprise Java Beans, as depicted in the Figure 3 below. It also comprises two J2EE fat clients.

In this appendix, we describe what the webbank application does and how it was written. We also explain in more details some of the critical design points of this application.

0.1 Webbank Design Overview

This application has been written following the Best Practices for WebSphere development and design, namely:

- Usage of the Model-View-Controller architecture.
- Session facades: entity beans are always accessed through session beans.
- Usage of sendRedirect to prevent multiple executions of a transfer (the infamous “reload problem”)

- Usage of Helper Classes to:
  - shield clients (i.e. servlets) from the complexity of using EJBs.
  - cache EJB home lookups
- Use of Java Script for basic form validation

OID GOES HERE

This application also uses the WebSphere Application Server tracing/logging mechanisms (JRas API).

0.2 Webbank Functionality Overview

There are multiple ways to use the Webbank Application, the same functionality being available via web or via J2EE clients. You can basically do two operations:

- View the branch and customer accounts balance
- Transfer money between accounts.

0.2.1 Presentation Layer Overview

You can reach the webbank application by calling `http://<hostname>/webbank/index.html`. If you want to make a transfer, you have two choices: use the “traditional” implementation or a Struts-based implementation. Both do exactly the same thing, and are meant to illustrate how to use various technologies.

0.2.2 Transfer Servlet Overview

The Transfer servlet gathers the transfer information from an HTTP form, and invokes the makeTransfer(TransferDO) method of the TransferHelper class. The TransferHelper class, which is implemented as a singleton, makes an access to the Transfer session bean (stateful session EJB) transparently to the servlet. It also caches the home of this EJB. The TransferDO class is a value object, a JavaBean used to transport the data coming from the HTTP form back to the enterprise Java beans layer. The BranchAccount entity bean represents a branch account, its primary key is the branch office name (such as Sophia). The Customer entity bean represents a customer account, its primary key is composed of the customer name (such as Isabelle) and the customer account ID (such as A1). Both entity beans are persistent into a database called WEBBANK.

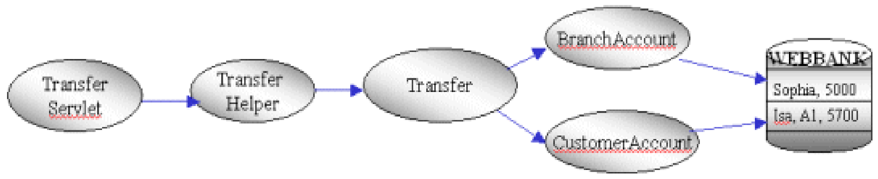


Figure 0-1 Transfer Servlet Invocation Flow

0.2.3 Using the Consultation Servlet

The Consultation servlet gathers accounts information from a form, and simply displays the balance of each account.

0.2.4 Using the J2EE clients

Information on fat clients goes here...

0.3 Webbank Struts Implementation

The webbank sample has also been used to show how to use the Struts framework. There are two ways you can invoke the webbank application. If you call webbank.html, then you will use the traditional way, that is HMTL calling a servlet, calling a JSP. If you call webbank.jsp, then you will use the Struts framework.

0.4 Webbank Clerk Application

This application is new in the V5 handbook. It has been developed to demonstrate the benefit of message-driven beans (MDB). Each time a transfer over a certain amount has been done, the clerk responsible for this account will see a log when starting the monitoring application. This application has been developed with WebSphere Studio Application Developer version 5.0 EA.

The MDB waits for messages on the Webbank queue. Whenever a message arrives, the application analyzes it and logs the message in a LOGTRANS table (via straight JDBC).

0.5 Detailed Enterprise Java beans Information

0.5.1 Why using a stateful Transfer session EJB?

Stateful EJBs are rarely used in applications. Most of the time, you want to use stateless EJBs for scalability and performance. Using a stateful EJB for this application find his roots in its original purpose: teaching EJB transactions. We used a stateful EJB to be able to use bean-managed transactions and drive the transaction from a graphical client. When you use the J2EE Transfer client, you see that you can begin/rollback/commit a user transaction from the GUI. This is possible **only** by using a stateful EJB. We thought we would leave this implementation for information/teaching purposes, plus this allows to have all types of EJBs in the application. However, we would not have implemented the Transfer EJB as a stateful EJB in a real application. A stateless EJB would be much better, since there is no need for a client to handle transactions boundaries.

If you want to use the stateless version of the Transfer EJB rather than the stateful version, all you need to do is updating the Tranfer EJB reference binding (ejb/Transfer) defined in the Webbank web module and theTransfer J2EE client. This binding should be set to webbank/ejb/TransferStateless.

0.6 Installing the Webbank Application

0.6.1 Instructions for DB2

All entity EJBs will be persistent in a database called WEBBANK. Make sure you are logged in with an NT ID which is less than 8 characters long, and that has DBA rights, for example *db2admi*n. Once the database has been successfully created, you can import the schema definition for the entity EJBs using the Table.ddl file that you will find under c:\webbank\install, then populate some data using the webbank.cli file.

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. Create the WEBBANK tables
   ```
   db2 connect to webbank user <username> using <password>
   db2 -tvf Table.ddl
   ```
4. Populate the Webbank tables with some records
   ```
   db2 -tvf webbank.cli
   ```

0.6.2 Setup for Cloudscape

0.6.3 Building Webbank with ANT

Appendix B.

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 1088.

- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx

Other resources

These publications are also relevant as further information sources:

- *Java 2 Platform, Enterprise Edition Management Specification,* by Hans Hrasna, available at `http://java.sun.com/j2ee/tools/management`
- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx

Referenced Web sites

These Web sites are also relevant as further information sources:

- Description1

  `http://????????.???.???/`
- Description2

  `http://????????.???.???/`
- Description3

  `http://????????.???.???/`

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set**

Redbooks

(1.5" spine)
1.5"<-> 1.998"
789 <->1051 pages

Redbooks

WebSphere Application Server Version 5.0

(1.0" spine)
0.875"<->1.498"
460 <-> 788 pages

Redbooks

WebSphere Application Server Version 5.0 Handbook

(0.5" spine)
0.475"<->0.875"
250 <-> 459 pages

Redbooks

WebSphere Application Server Version 5.0 Handbook

(0.2"spine)
0.17"<->0.473"
90<->249 pages

(0.1"spine)
0.1"<->0.169"
53<->89 pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5” spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set**

Redbooks

(2.5” spine)
2.5”<->nnn.n”
1315<-> nnnn pages

Redbooks

(2.0” spine)
2.0” <-> 2.498”
1052 <-> 1314 pages

WebSphere Application Server Version 5.0 Handbook

Redbooks

This redbook provides readers with the knowledge needed to implement WebSphere Application Server V5.0, Advanced Edition runtime environment, to package and deploy Web applications, and to perform ongoing management of the WebSphere environment. This project will update SG24-6176, WebSphere V4.0 Handbook, currently available at http://www.redbooks.ibm.com/abstracts/sg246176.html.

SG24-6195-00 ISBN