IBM WebSphere Application Server Network Deployment, Version 5

Getting started

Note

Before using this information, be sure to read the general information under "Trademarks and service marks" on page v.

Compilation date: November 20, 2002

Contents

Trademarks and service marks

Chapter 1. Conceptual overview

Java 2 platform, Enterprise Edition (J2EE)

The J2EE platform is the standard for developing, deploying, and running enterprise applications. Read this document for a brief overview of key J2EE concepts, including the parts of the J2EE runtime environment and the J2EE application packaging and deployment. The ultimate source of J2EE information is the specification itself, available from the Sun Microsystems Web site (java.sun.com).

IBM® WebSphere® Application Server, Version 5 has completed the full J2EE certification test suite. The product supports all of the J2EE 1.3 APIs, and exceeds many with its extensions. You can check the list of J2EE-compatible configurations posted by Sun Microsystems at http://java.sun.com/j2ee/1.2_compatibility.html.

What is J2EE?

The acronym J2EE stands for Java™ 2 Platform, Enterprise Edition. It defines a standard that applies to all aspects of designing, 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. For more information, see Three-tier architectures.

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.

For information about the relationship of IBM WebSphere Application Server roles to J2EE roles, see ″User roles and activities″.

J2EE benefits

The J2EE standard empowers customers. Customers can compare J2EE offerings from vendors and know that they are comparing apples with apples. Comprehensive, independent Compatibility Test Suites ensure 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 the write-once, run-anywhere Java technology.
- Services providing integration with existing systems, including Java DataBase Connectivity (JDBC); Java Message Service (JMS); Java Interface Definition Language (Java IDL); JavaMail; and Java Transaction API (JTA and JTS) 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 more than 80 percent of the J2EE APIs.

Application components and their containers

The J2EE programming model has four types of application components, which reside in four types of containers in the application server:
- Enterprise JavaBeans™ components, residing in the EJB container
- Servlets and JavaServer Pages files, residing in the Web container
- Application clients, residing in the application client container
- Applets, residing in the applet container

For a thorough description of the components and containers, because this information also applies specifically to the IBM WebSphere Application Server, see ″Architectural features″.

J2EE containers provide the runtime support of the application components. 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.

Standard services

The J2EE platform provides components with a set of standard services that they can use to interact with each other. See the Sun Web page, http://java.sun.com/products/, for descriptions of each standard service.
- HTTP and HTTPS
- Java Transaction API (JTA)
- Remote Method Invocation/Internet Inter-ORB Protocol (RMI/IIOP)
- Java Interface Definition Language (Java IDL)
- Java DataBase Connectivity (JDBC)
- Java Message Service (JMS)
- Java Naming and Directory Interface (JNDI)
- JavaMail and JavaBeans Activation Framework (JAF)
- Java Transaction API (JTA and JTS)
- XML
- J2EE Connector Architecture
- Resource managers

J2EE packaging

Perhaps the most significant change introduced by the J2EE specification is how application components are packaged for deployment.

During a process called assembly, J2EE components are packaged into modules. Modules are then packaged into applications. Applications can be deployed on the application server. Each module and application contains a J2EE deployment descriptor. The deployment descriptor is an XML file providing instructions for deploying the application.

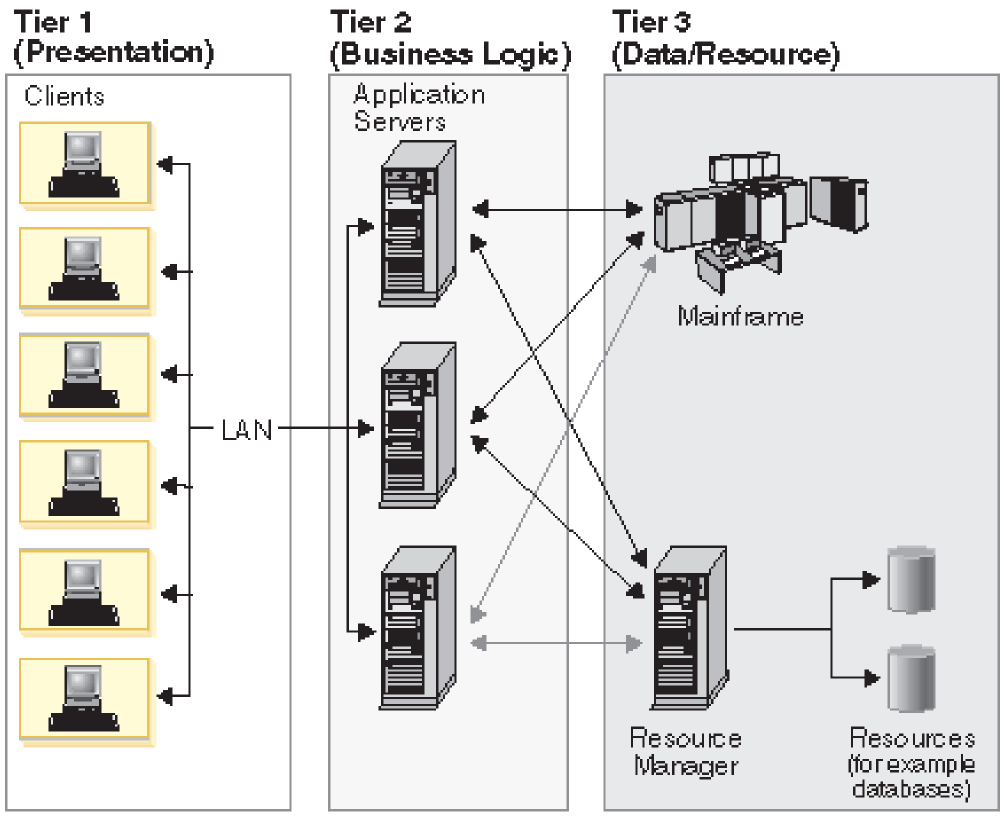
For more information, including IBM WebSphere Application Server specifics, see the *Assembling or packaging* topic (welc_assembling) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

Three-tier architectures

WebSphere Application Server provides the application logic layer in a three-tier architecture, enabling client components to interact with data resources and legacy applications.

Collectively, *three-tier architectures* are programming models that enable the distribution of application functionality across three independent systems, typically:
- Client components running on local workstations (tier one)
- Processes running on remote servers (tier two)

- A discrete collection of databases, resource managers, and mainframe applications (tier three)



First tier. Responsibility for presentation and user interaction resides with the first-tier components. These client components enable the user to interact with the second-tier processes in a secure and intuitive manner. WebSphere Application Server supports several client types.

Clients do not access the third-tier services directly. For example, a client component provides a form on which a customer orders products. The client component submits this order to the second-tier processes, which check the product databases and perform tasks needed for billing and shipping.

Second tier (application logic layer). The second-tier processes are commonly referred to as the application logic layer. These processes manage the business logic of the application, and are permitted access to the third-tier services. The application logic layer is where the bulk of the processing work occurs. Multiple client components are able to access the second-tier processes simultaneously, so this application logic layer must manage its own transactions.

Continuing with the previous example, if several customers attempt to place an order for the same item, of which only one remains, the application logic layer must determine who has the right to that item, update the database to reflect the purchase, and inform the other customers that the item is no longer available. Without an application logic layer, client components access the product database directly. The database is required to manage its own connections, typically locking out a record that is being accessed. A lock can occur simply when an item is placed into a shopping cart, preventing other customers from even considering it for purchase. Separating the second and third tiers reduces the load on the third-tier services, can improve overall network performance, and allows more eloquent connection management.

Third tier. The third-tier services are protected from direct access by the client components by residing within a secure network. Interaction must occur through the second-tier processes.

Communication among tiers. All three tiers must be able to communicate with each other. Open, standard protocols and exposed APIs simplify this communication. Client components then can be written in any programming language, such as Java or C++, and can run on any operating system, as long as they can speak with the application logic layer. Likewise, the databases in the third tier can be of any design, as long as the application layer can query and manipulate them. The key to this architecture is the application logic layer.

Architectural features

This document examines the major components within IBM WebSphere Application Server, including the application server and its containers; the HTTP server plug-in and embedded HTTP handling; and the virtual host configuration.

HTTP server

IBM WebSphere Application Server works with an HTTP server to handle requests for servlets and other dynamic content from Web applications. (The terms HTTP server and Web server are used interchangeably throughout the documentation.)

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

For more information, see the *Configuring Web server plug-ins* topic (trun_plugin) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

Application server

The application server collaborates with the Web server to return customized responses to client requests. Application code including servlets, JavaServer Pages (JSP) files, Enterprise JavaBeans (EJB) components, and their supporting classes run in an application server. In keeping with the J2EE component architecture, servlets and JSP files run in a Web container, and enterprise beans run in an EJB container. You can define multiple application servers, each running in its own Java Virtual Machine (JVM).

- **EJB container**

  The EJB container provides the runtime services needed to deploy and manage EJB components, from here on known as enterprise beans. It is a server process that handles requests for both session and entity beans.

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

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

For more information, see the *EJB containers* topic (cejb_ecnt) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

- **Web container**

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

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

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

  A Web container handles requests for servlets, JavaServer Pages (JSP) files, and other types of files that include server-side code. The Web container creates servlet instances, loads and unloads servlets, creates and manages request and response objects, and performs other servlet management tasks. WebSphere Application Serve provides Web server plug-ins for supported Web servers. These plug-ins pass servlet requests to Web containers.

- **Application client container**

  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 container handles Java application programs that accesses enterprise beans, Java Database Connectivity (JDBC), 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.

  For more information, see the *Application clients* topic (ccli_appclients) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

- **Applet container**

  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.

  The applet container handles Java applets embedded in a HyperText Markup Language (HTML) documents that reside on a client machine that is remote from the application server. With this type of client, the user accesses an enterprise bean in the application server through the Java applet in the HTML document.

For more information, see the *Application clients* topic (ccli_appclients) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

Embedded HTTP server. A nice product feature is the HTTP handling capability embedded within the application server, enabling an HTTP client to connect directly to the application server. Or, as described earlier, an HTTP client can connect to a Web server and the HTTP plug-in can forward the request to the application server.

Virtual host

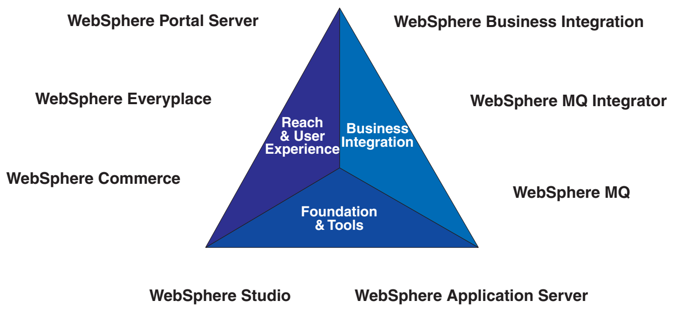
A virtual host is a configuration enabling a single host machine to resemble multiple host machines. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share the same physical machine.

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

For more information, see the *Virtual hosts* topic (crun_vhost) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

Product family overview

A variety of product lines complementing IBM WebSphere Application Server are described below. To investigate product availability and pricing, see the www.ibm.com Web pages for each product.



Foundation and tools

The following products collaborate to help you achieve scalability and productivity by growing and building e-business applications, rapidly and reliably.

- **WebSphere Application Server**

  Packaging changes in Version 5 make it easier than ever to upgrade as needed, as your business grows. See also WebSphere Application Server packages.

The product is available for several distributed operating systems, as well as specifically optimized for zSeries™ and iSeries™. For more information, see http://www.ibm.com/webservers/appserv/.

- **WebSphere MQ**

  This is software for exchanging information among more than 35 platforms with assured delivery. For more information, see http://www.ibm.com/software/ts/mqseries/.
- **WebSphere Studio**

  These are e-business professional tools based on a common workbench technology. Based on open standards, they exploit the WebSphere runtime environment. For more information, see http://www.ibm.com/software/webservers/studio/.

Reach and user experience

The following products collaborate to help you achieve customer loyalty by extending and personalizing user experiences.

- **WebSphere Portal**

  This is software for accessing widespread and diverse data sources from anywhere, anytime, by anyone you allow. For more information, see http://www.ibm.com/software/webservers/portal/.
- **WebSphere Everyplace™**

  This is software infrastructure to support mobile solutions, addressing the challenge of extending e-business applications to mobile devices. For more information, see http://www.ibm.com/software/pervasive/.
- **WebSphere Commerce**

  These are powerful solutions designed to handle the broad range of challenges encountered when selling in Business-to-Business (B2B) and Business-to-Consumer (B2C) environments. For more information, see http://www.ibm.com/software/webservers/commerce/.

Business integration

The following products collaborate to help you achieve business agility by integrating applications and automating business processes.

- **WebSphere Business Integration**

  This software creates the nimble infrastructure needed to support the business imperatives of your dynamic enterprise. For more information, see http://www.ibm.com/software/integration/.
- **WebSphere MQ Integrator**

  This software helps you to flexibly connect and integrate their assets within the enterprise and with trading partners. For more information, see http://www.ibm.com/software/integration/.

Product GUIs and tools

IBM WebSphere Application Server provides a variety of graphical and non-graphical user interfaces. The product GUIs are described, as well as some key scripting and command line tools. For a complete list of available commands, as well as more information about the commands referenced below, see the **Command line syntax** section of the **Quick reference** view in the IBM WebSphere

Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

Tools for installing, upgrading, and migrating

First Steps
First Steps is a desktop GUI from which you can start or stop the application server. It also provides access to the administrative console and the Application Assembly Tool. See *First Steps tool tips*.

Launch Pad
The Launch Pad is a graphical interface for launching the product installation. It also provides links to information you might need for installation. See *Using the LaunchPad to start the installation*.

Installation wizard
This graphical interface leads you through the process of installing the product.

Migration tools
Command line tools such as WASPreUpgrade and WASPostUpgrade are available to help you migrate from a previous product version.

Tools for developing applications

IBM WebSphere Application Server is not an application development tool, although it provides some Application Programming Interfaces (APIs) for improving the applications you deploy on the server. The WebSphere Studio Application Developer product line offers development environments and tools that complement each edition of the application server. For more information, see *Product family overview*.

Tools for assembling applications

As described in the *Assembling or packaging* topic (welc_assembling) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html), assembly is a necessary packaging and configuration step prior to deploying an application onto the server.

Application Assembly Tool
AAT is used to assemble enterprise applications for deployment. For more information, see the *Assembling applications* topic (uaatt_skel) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

Deployment tool
This is a command-line tool that the graphical Application Assembly Tool calls behind the scenes, to generate code for deployment. In specific cases, you might need to use it. See the *ejbdeploy tool* topic (raat_ejbdeploycmd) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

Application Client Resource Configuration Tool

This graphical interface helps you configure deployment descriptors that define the resources needed by application clients. For more information, see the *Deploying application clients* topic (ucli_tconfigclient) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html)..

clientUpgrade
Use this tool to migrate client JAR files from J2EE 1.2 to J2EE 1.3.

Text editor
While it is recommended that you use the graphical AAT to edit deployment descriptors, these XML documents can be opened in your favorite text editor.

Tools for deploying and administering

As described in the *Deploying* topic (welc_deploying) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html), deploying involves putting an application onto a particular server.

Systems administration tools
See the *Welcome to System Administration* topic (welc_configop) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html), for a description of the available systems administration tools, including the graphical WebSphere Administrative Console, the WSAdmin scripting client, and an assortment of special purpose command line tools.

A notable part of the WebSphere Administrative Console is the **Security Center**.

LaunchClient command
This command line tool deploys application clients. See the *launchClient tool* topic (rcli_javacmd) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

XML-SOAP administrative tool
This graphical interface helps you manage deployed Web services. See the *Administering deployed Web services (XML-SOAP administrative tool)* topic (twbs_adminwbs) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

UDDI user console
This is the graphical interface for interacting with the IBM WebSphere UDDI Registry. See the *UDDI user console* topic (twsu_uc) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

NameSpaceDump tool
This command line interface empties the IBM WebSphere Application Server name space for examination. See *dumpNameSpace tool* topic (rnam_dump_utility) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

Tools for monitoring and tuning

PMI Request Metrics in WebSphere Administrative Console
The product collects data by timing requests as they travel through components of the product. PMI Request Metrics, which is configurable through the administrative console, logs the time spent in major components, such as the Web container of the application server. These

data points are recorded in logs and can be written to Application Response Time (ARM) agents used by Tivoli® monitoring tools.

Tivoli Performance Viewer
This is a stand-alone program that monitors and helps analyze application server data. It is built on the Performance Monitoring Infrastructure (PMI) Client API, which also is exposed to third-party development tools. For more information, see the *Monitoring performance with Tivoli Performance Viewer (formerly Resource Analyzer)* topic (tprf_tpvmonitor) and the *Performance Monitoring Infrastructure* topic (cprf_pmidata) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

Tools for troubleshooting

Application Server Toolkit
Included with IBM WebSphere Application Server, but on a separately installable CD, this kit includes debugging functionality that is built on the Eclipse workbench. See the *Debugging with the Application Server Toolkit* topic (ctrb_debug) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

Collector tool
The Collector Tool gathers information about your WebSphere Application Server installation and packages it in a jar file that can be sent to IBM Customer Support to assist in problem determination and analysis. For more information, see the *Collector Tool* topic (ctrb_ct) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

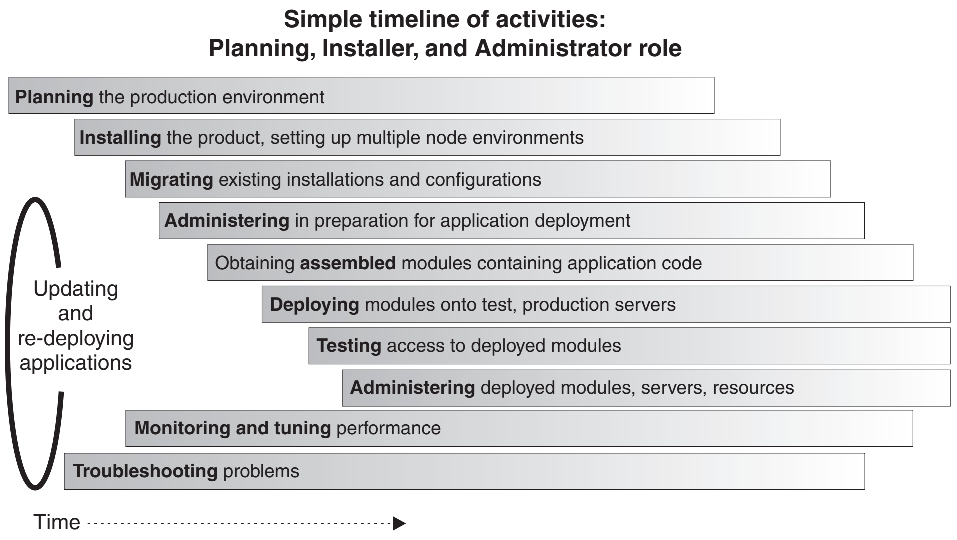
First Failure Data Capture (FFDC)
The First Failure Data Capture tool preserves the information generated from a processing failure and returns control to the affected engines. The captured data is saved in a log file for use in analyzing the problem. The First Failure Data Capture tool is intended primarily for use by IBM Service. For more information, see the *First Failure Data Capture tool* topic (ctrb_ffdc) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

Log Analyzer
The Log Analyzer takes one or more service or activity logs, merges all of the data, and displays the entries. Based on its symptom database, it analyzes and interprets the error conditions in the log entries to help you debug problems. Log Analyzer has a special feature enabling it to download the latest symptom database from the IBM Web site. For more information, see the *Log Analyzer* topic (ctrb_jfla) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

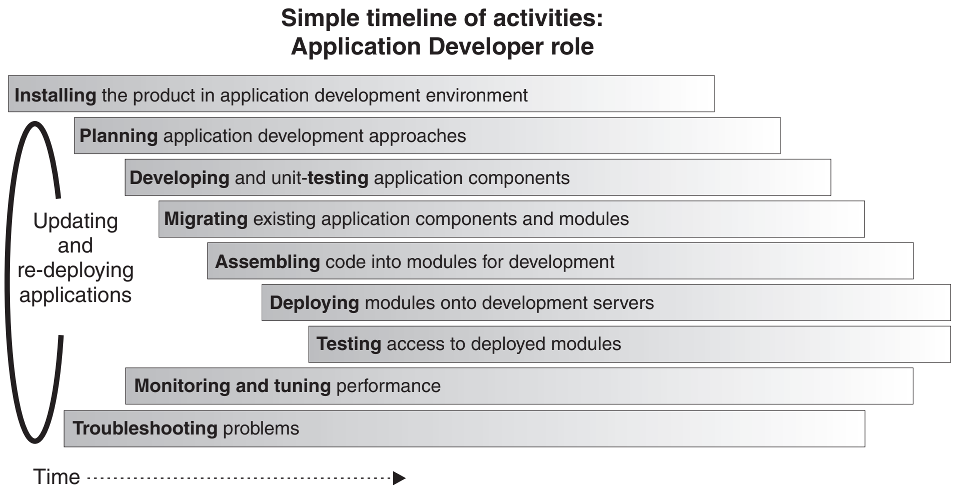
User roles and activities

Review the user role and activity descriptions to understand how someone with your organizational role might use WebSphere Application Server. If you are new to the product or Java 2 Platform, Enterprise Edition (J2EE), the simplified model presented here should help you with the basic flow of activities involved in seeing a J2EE application through to deployment.

Setting up and administering a production environment follows roughly these phases.


Simple timeline of activities:
Planning, Installer, and Administrator role
Planning the production environment
Installing the product, setting up multiple node environments
Migrating existing installations and configurations
Administering in preparation for application deployment
Obtaining **assembled** modules containing application code
Deploying modules onto test, production servers
Testing access to deployed modules
Administering deployed modules, servers, resources
Monitoring and tuning performance
Troubleshooting problems
Updating and re-deploying applications
Time

Using the application server in an application development environment follows roughly these phases.


Simple timeline of activities:
Application Developer role
Installing the product in application development environment
Planning application development approaches
Developing and unit-**testing** application components
Migrating existing application components and modules
Assembling code into modules for development
Deploying modules onto development servers
Testing access to deployed modules
Monitoring and tuning performance
Troubleshooting problems
Updating and re-deploying applications
Time

The product tools (described in *Product GUIs and tools*) and documentation are geared towards helping you with these activities. As you learn more, you will see ways to tailor the flow of activities to your specific needs.

Take advantage of task overviews. Task overviews are special sets of steps in this documentation set. Each outlines a feasible sequence of tasks for working with an area of product functionality, such as security. The tasks typically reflect the main activities, such as Migrating, Developing, Assembling, Deploying, and so on. Use task overviews to gain broad knowledge of the decisions and actions needed to accomplish your goals. From task overviews, you can drill down to more detailed sub-tasks.

For a list of available task overviews, see the **Task overviews** section of the **All topics by activity** view of the online IBM WebSphere Application Server, Version 5

InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

Mapping to J2EE roles. The following is the mapping of WebSphere Application Server roles and activities to the roles defined by the J2EE 1.3 specification.

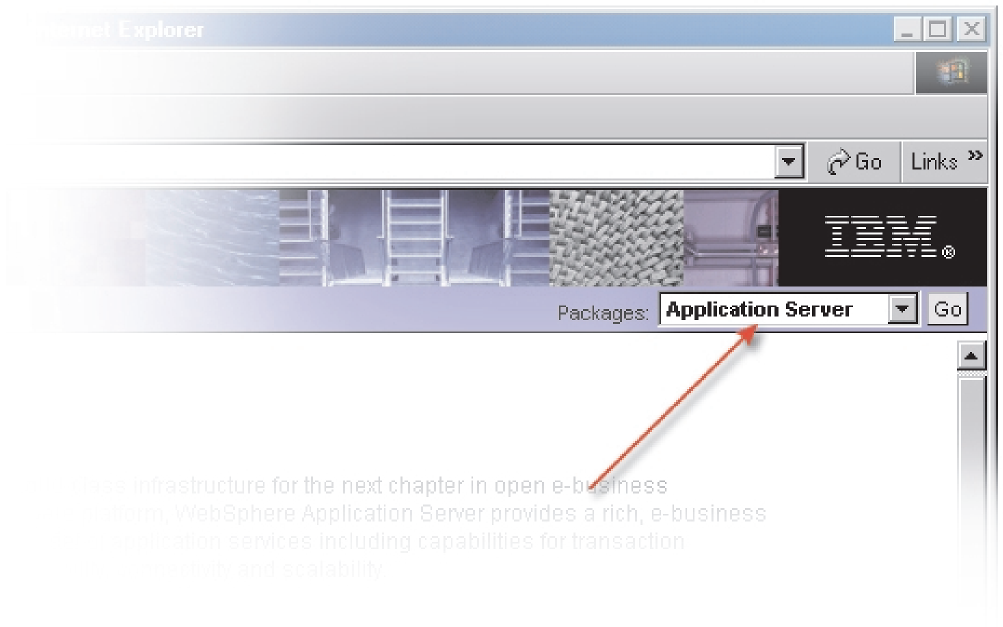
J2EE roles	Product roles	Product activities
Non-applicable	Planner, Installer, IT architect	Planning, Installing product environment
Application Component Provider	Developer / Programmer	Developing
Application Assembler	Developer / Programmer	Assembling
Deployer	Administrator (in production environment)	Deploying, Testing application deployment
System Administrator	Administrator	Administering
Non-applicable	All of the above	Migrating, Tuning, Troubleshooting
Tool Provider, J2EE Product Provider	WebSphere Application Server and product family	Non-applicable

InfoCenter

The InfoCenter displays the documentation for IBM WebSphere Application Server. The IBM WebSphere Application Server, Version 5 InfoCenter is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

Package selection

The product packaging structure is described in *WebSphere Application Server packages*. Use the **package selection** menu on the right-hand side of the InfoCenter banner to select documentation for a particular package of the product. Selecting a package resets the left-hand navigational views to show only the topics that pertain to that package.


Internet Explorer
Go
Links
IBM
Packages:
Application Server
Go
...ss infrastructure for the next chapter in open e-business
...form, WebSphere Application Server provides a rich, e-business
...application services including capabilities for transaction
...tivity and scalability

Navigational views

The navigation frame, located on the left side of the InfoCenter under the banner frame, houses the expandable table of contents. Navigational views include:

- **All topics by feature.** This hierarchical view organizes the topics according to the main technologies in the product domain, such as Web services and J2EE resources. This view is most suitable for reading topics in the sequence outlined by the back and next links included in each topic:

  ←

  →

- **All topics by activity.** This hierarchical view organizes topics according to tasks that might be familiar to you already from working with other software.

  The top-level categories in this view outline the main phases, from application development through deployment and updates. The organization in the next couple of levels of this view reflects that of the **All topics by feature** view.
- **Quick reference.** This fairly flattened view highlights reference information for quick lookup — such as confirming the syntax for a particular command line argument.

  The Quick reference view makes every reference topic accessible in two clicks, for quick retrieval. For example, you can look up syntax for the addNode command by clicking **Commands**, then clicking the name of the documentation topic — *addNode command* (rxml_addnode in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html). In contrast, the other navigational views require three or more clicks to view many of these topics, which tend to reside deeper in the document hierarchy.

This view includes sets of links to additional sources of information, outside of the InfoCenter. These links can include IBM Redbooks™, Best Practice documents, WebSphere Developer Domain topics, and a variety of others.

Information type indicators

Each document that is displayed in the right-hand frame of the InfoCenter is known as a *topic* or topic. Most topics belong primarily to one of the following *information types*:

	Concept	Concept topics answer the question, ″What is...?″ They explain what something is and why it behaves the way it does.
✓	Task	Task topics answer the question, ″How do I...?″ Some tasks lead you through a thought process, to arrive at a decision. Other tasks lead you through a detailed series of cursor clicks and keystrokes to produce a tangible result, such as updating and saving a configuration file. Some tasks lend themselves to sequential steps; others do not. When there are *many* valid sequences for performing a set of steps, topics in the InfoCenter typically present *one* of the valid sequences. This simplification helps novice users learn and complete tasks successfully. It is assumed that more advanced users will know or easily discover alternative sequences, adapting the documented sequence to their needs.

📖	Reference	Reference topics provide quick access to facts. For example, Javadoc is a type of reference information from which application developers look up method signatures for a specific Application Programming Interface (API). The Javadoc for this product is organized alphabetically by package name. As with a telephone directory, reference information is not intended to be read in large chunks. Reference topics — and the entries within reference topics — are organized for quick lookup, rather than to highlight their interrelationships. For example, two entries in the **Commands** section of the **Quick reference** view might have little in common, except that they both are valid commands that you can type at a command prompt. Based on alphabetization, the two otherwise unrelated items might be listed one directly after the other. It is anticipated that you might look up one of these commands on one occasion, perhaps needing the other one on entirely different occasion.
ℹ	Help file	These topics are the help files for the graphical user interfaces of the product. They are a mixture of concept, task, and reference information. One copy of these files is installed with the product, enabling you to view these topics by clicking a **Help** link or button. Another copy is included in the InfoCenter. The InfoCenter version of these files can contain updates that have not been included in the installed help files yet.

To use the product documentation effectively, you need **not** understand or rely upon the information types. Their primary purpose is to make the information more predictable and consistent, whether or not you notice their contribution. Because graphics are used to indicate each information type in the InfoCenter left-hand navigational views, the meaning of the graphics is explained.

Search

A full text document search is available from the banner of the online InfoCenter. For more information, see the *InfoCenter search help* topic (howtosearch) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).

To look for key words in topic titles only:
1. Open the site map from the InfoCenter banner.
2. Right-click at the top of the site map.
3. Use your browser **Find** capability to search the site map for key words or phrases.

Article IDs

To find the file name for a particular page in the InfoCenter, position your cursor over the icon found at the bottom of the page, next to the date stamp. The hover help is displayed, containing the file name. You can also search on this file name.

For example, if you position your cursor over the icon at the bottom of the *Launching scripting clients* topic (txml_launchscript) in the IBM WebSphere Application Server, Version 5 InfoCenter (at http://www-3.ibm.com/software/webservers/appserv/infocenter.html), the hover help is displayed. The file name is txml_launchscript. To help someone else to find that topic, you can send the person the file name, which the person can enter into the InfoCenter search window to locate the topic.

Bookmarks

Take advantage of your Web browser capabilities to bookmark (or add to *favorites*) a topic that is displayed in the right-hand frame of the InfoCenter.
1. Right-click anywhere in the right-hand frame displaying the InfoCenter topic you want to bookmark.
2. Select **Add Bookmark** if you are using Netscape Navigator. Select **Add to Favorites** if you are using Microsoft Internet Explorer.

Font size adjustment

A cascading style sheet (CSS) governs the appearance of the InfoCenter. However, the text size remains flexible. The text is displayed in the size indicated in your Web browser settings.

The instructions for changing the text or font size vary between browser brands and versions. Consult the browser product documentation for instructions. In Microsoft® Internet Explorer, look for the menu choice, **View > Text Size**. In Netscape browsers, look for the menu choice, **View > Increase Font Size**.

Browser support

The InfoCenter supports Netscape Navigator 6.0 and Microsoft Internet Explorer 6.0 and higher. In addition, JavaScript™ must be enabled in the browser. If you use a different or earlier version of a browser, your pages may format differently and some functions may not work correctly.

Printing

The PDF versions (available separately from InfoCenter) are recommended if you would like to print multiple topics with the intent of reading the topics in sequence, like a book.

Browser capabilities may be used to print single topics from the online InfoCenter. To print a topic displayed in the right-hand frame of the InfoCenter, first click anywhere in the right-hand frame to register that frame as the selected frame. Then use your browser to print the selected, right-hand frame.

The online version is subject to change frequently. Timestamps at the bottom of each topic help you keep track of when you printed the information.

Accessibility features

Accessibility features enable a user with a physical disability, such as restricted mobility or limited vision, to operate software products successfully.

The administrative console is the primary interface for interacting with the product. This console is displayed within a standard Web browser. By using an accessible Web browser, such as Microsoft Internet Explorer or Netscape Navigator, administrators are able to:

- Use screen-reader software and a digital speech synthesizer to hear what is displayed on the screen
- Use voice recognition software, such as IBM ViaVoice™, to enter data and to navigate the user interface
- Operate features by using the keyboard instead of the mouse

One can configure and administer product features by using standard text editors and scripted or command line interfaces instead of the graphical interfaces provided.

When appropriate, the documentation for specific product features contains additional information about the accessibility of the features.

Chapter 2. What is new in Version 5

IBM WebSphere Application Server, Version 5, offers a world-class infrastructure for the next chapter in open e-business platforms. As the foundation of the WebSphere software platform, WebSphere Application Server provides a rich, e-business application deployment environment with a complete set of application services including capabilities for transaction management, security, clustering, performance, availability, connectivity and scalability. Version 5 offer full J2EE specification support (Servlet 2.3, JSP 1.2, EJB 2.0, and others), as well as a variety of extensions.

Several new and improved Version 5 features are summarized here, with links to more information.

Planning, installation, and migration

- The product offers production-ready J2EE 1.3 standards and Web services support.
- The new packaging structure simplifies the conversion of standalone, single machine product installations into multiple node environments. To the base WebSphere Application Server package, the WebSphere Application Server Network Deployment package adds:
  - Clustering, including workload management (weighted bind policy), failover, distributed security, and distributed naming
  - Distributed systems administration, including single-system image, cluster creation and management, configuration and application distribution, and monitoring

  For more information, see *WebSphere Application Server packages*.
- The multiple node environments are created by *federating* multiple WebSphere Application Server installations into *cells*, each of which is managed by a *deployment manager* provided by a Network Deployment installation.

  For more information, see the administrative agents section of the *Welcome to System Administration* topic (welc_configop) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
- An essential new task for users of Network Deployment involves the addNode command. In summary, use this command to *federate* the configuration of a standalone WebSphere Application Server installation into a cell-wide configuration store for distributed management.

  Unlike Version 4, you do not install new software on the standalone server to make it part of the set of servers grouped under a single deployment manager. The set of servers is known as a *cell*.

  For more information, see Establishing multimachine environments.
- If migrating from Version 4 to a multiple machine environment including the Network Deployment package, it is best to migrate the Network Deployment node before adding other nodes to the central configuration. The Version 5 migration tools are available to help you.

  When you migrate to Version 5, the configuration from the previous version is split between configuration information relative to the Network Deployment

node (specifically, the deployment manager on that node) and the other nodes it manages. In particular, the following objects are migrated on a deployment manager configuration:
  - Clusters
  - Security
  - Virtual hosts

  Indeed, migration tools cannot be used on an WebSphere Application Server node that already belongs to a cell managed by a deployment manager. To be able to use the migration tools, remove the node from the cell, use the tools, and then federate the node into the cell again.
- Migration tools help you migrate existing configurations from certain previous versions during the installation process. They include both pre-upgrade and post-upgrade activities. For more information, see Migrating and coexisting.
- Multiple server instances and multiple product installations on one machine are now supported.

  For more information, see the Multiple deployment manager cells and Multiple application servers within a node topics.
- The servlet redirector and remote OSE mechanisms are no longer supported. Instead, HTTP forwards Web requests from your Web server to an HTTP server running inside the appropriate application server.
- WebSphere Edge Server functionality has been integrated into IBM WebSphere Application Server, including a Demilitarized Zone (DMZ) CD containing software for load-balancing and content caching. The edge of the DMZ includes:
  - HTTP routers
  - HTTP server
  - Caching proxy
  - Deployment manager

  For more information, see the *Configuring Edge Side Include caching* topic (tprf_esiedgecaching) in the InfoCenter.

New and improved WebSphere Samples Gallery

- Technology centered samples, including EJB, J2EE client, JMS, JSP, and Servlet samples.
- The **Plants by WebSphere** ″super sample,″ demonstrating multiple technologies used to build realistic applications.
- Java Petstore sample
- ANT-based build scripts enabling you to run, modify, rebuild, and run the samples again.
- Use of Cloudscape rather than DB2 for the samples requiring a database. Cloudscape has a smaller footprint.

For more information, see *Samples Gallery*.

Servers

- The product now has a single JVM runtime, including containers, naming, security, administration, resources, ORB, and HTTP engine

- The new, flexible packaging structure means there is just one application server code base, with add-ons available for scale and function. The application server now runs with the high performing:
  - IBM Developer Kit, Java Technology Edition, Version 1.3.1 used on AIX, Windows, and Linux operating systems
  - Java virtual machine from Sun on Solaris operating system

  For more information, see the *Using the JVM* topic (trun_jvm) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
- Workload management and clustering has improved:
  - Clusters now can be managed with browser based, scripting, and JMX administrative clients
  - The ability to configure member weights provides the administrator more control in tuning the system
  - A new load balancing algorithm has been incorporated to better optimize the distribution of client requests across cluster members
  - Enhanced failover support provides quicker failover in the case of network outages and other problems

  In Version 4, clusters were known as *server groups* or model and clones.

  For more information, see the *Balancing workloads with clusters* topic (trun_wlm) in the IBM WebSphere Application Server, Version 5 InfoCenter.
- In Version 3.5.x and Version 4, the default bootstrap setting was 900. Now the default is 2809. This does not cause problems in applications unless there is a direct reference to port 900. You can reconfigure the port setting to 900 if you migrate your previous configuration.

  For more information, see Port number settings in WebSphere Application Server versions.
- The system name space structure provided by the name server has changed significantly since the last release, including:
  - The Version 5 name space is distributed, meaning that objects are not all bound under a single context root as with previous versions.
  - The name space consists of partitions. Some partitions in the name space contain transient bindings and some partitions contain persistent bindings.

  These and other new naming features are summarized in the *New features for name space support* (rnam_new_features) topic in the InfoCenter.

Applications > EJB modules

- EJB persistence manager has been re-architected to support EJB 2.0 CMP scheme, which differs greatly from the EJB 1.1 scheme
- EJB persistence manager has improved in modularity, maintainability, and performance. Maintenance focuses on configuration of server components, with less need to regenerate deployed artifacts
- EJB 2.0 specification support, including:
  - Local and remote beans
  - Message-driven beans
  - Container-managed relationships
  - A portable finder query language

  - All other aspects of the specification
  - Programming model
  - Abstract and concrete entity beans
  - Local home and local entity interfaces
  - Container-managed association relationships
  - Dependent values
  - EJB query language
- Plus these features that add high performance persistence beyond EJB 2.0:
  - Changing semantic behavior
  - Entity bean inheritance
  - Optimistic concurrency control
  - Read-ahead
  - Intent mechanism support
  - Support for different types of backend access mechanisms
  - Procedural access
  - SQLJ
  - Data caching

  EJB specification extensions are described in the *WebSphere extensions to the Enterprise JavaBeans specification* topic (rejb_spcx) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
- Powerful new features enhance Container-managed persistence (CMP) entity bean performance, including:
  - Caching of bean data at several levels
  - Long lifetime caching, for beans that change only infrequently and thus remain read-only across many transactions
  - Read-ahead, which pre-loads groups and working-sets of beans in a single datastore operation by following selected bean relationships
  - Optimistic concurrency control, which minimizes the amount of time data is actually locked during updates and thus increases overall throughput in heavily-used applications
- CMP beans and bean-managed persistence (BMP) beans can share datastore connections, allowing access to related data by both kinds of beans when in the same transaction.
- CMP beans can inherit from one another. (In other words, they can subclass one another.) The application server will recognize this during bean deployment and at runtime will — for example — allow finders to return beans of that class or any subclass. Inheritance can be expressed in relational datastores in either *single-table* or *root-leaf* arrangements.

Applications > Web modules

- Servlet 2.3 with Filters and Events
- JSP 1.2 with XML Syntax
- Changes in autoRequestEncoding and autoResponseEncoding

  The web container no longer automatically sets request and response encodings and response content types. The programmer is expected to set these values using the methods available in the Servlet 2.3 API. If you want the application server to attempt to automatically set these values, set autoRequestEncoding=true in order to have the request encoding value set and set autoResponseEncoding=true in order to have the response encoding and content type set. These values can be found in the ibm-web-ext.xmi file for each web application.

For more information, see the *autoRequestEncoding and autoResponseEncoding* topic (cweb_autoreq) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

- *Filters* are Java classes that can be configured to operate on (filter) the request and response data of a requested resource.

  The resource to filter, and filter precedence is specified in the deployment descriptor information found in the web.xml file of a Web application. Initialization parameters for filters can also be specified in the web.xml. Filters can be chained and can be configured to work on a single resource or a group of resources. Typical usages for filters include logging filters, image conversion filters, encryption filters, and MIME-type filters (functionally equivalent to the old style servlet chaining).

  For more information, see the *Servlet filtering* topic (cweb_sfilt) in the IBM WebSphere Application Server, Version 5 InfoCenter.
- *Application lifecycle events* give the application developer greater control over interactions with ServletContext and HttpSession objects.

  Application event objects consist of application events and application listeners. Servlet context listeners are used to manage resources at an application level. Session listeners manage resources associated with a series of request from a single client. Listeners are available for lifecycle events and for attribute modification events. The listener developer creates a class that implements the javax listener interface corresponding to the desired listener functionality.

  For more information, see the *Application lifecycle listeners and events* topic (cweb_sctxl) in the InfoCenter.
- The HttpUtils class is deprecated in 2.3 and its methods are replaced by new methods in the request object. The HttpUtils class will still be available for use by servlet writers until a future servlet specification directs its complete removal.
- The product no longer requires the JSP-enabling servlet

  The *file serving enabled* check box in the *IBM extensions* tab of the Web module properties in the Application Assembly Tool controls this function. (It is selected by default.)

  Adding JSP files to the WAR file in the Application Assembly Tool, or adding them to the appropriate `application_name.war` directory of the installed Enterprise Application causes them to be served.

  Adding HTML files to the Web archive (WAR) file in the Application Assembly Tool, or adding them to the appropriate `application_name.war` directory of the installed Enterprise Application causes them to be served.
- New and improved features pertaining to **HTTP session support** include:
  – Multiple mechanisms for HTTP session state management, plus configuration options based on scalability and failover requirements from simple, single server environments to large, high-load clusters:
    - In memory
    - Persistent to database
    - Memory-to-memory

  - Replacement of the Session Manager object, which resided underneath each servlet engine in the WebSphere Application Server Version 4 topology. Its properties are now part of each application server.
  - Enhanced support for HTTP Session State failover, as described in the *Managing HTTP sessions* topic (tprs_sep1) in the InfoCenter.
  - With Version 5, a new option exists for saving HttpSession information for failure recovery purposes. In addition to a database, IBM WebSphere Application Server can save a HttpSession in more than one application server instance. Called ″in-memory session replication,″ this feature leverages the replication domain and replicator entry services provided in Network Deployment

Applications > Web services

- Web services enable businesses to connect applications to other business applications, to deliver business functions to a broader set of customers and partners, to interact with marketplaces more efficiently, and to create new business models dynamically. To that extent, the product provides four protocols that support Web services:
  - Web Services Description Language (WSDL), an XML-based description language that provides a way to catalog and describe services
  - Universal Discovery Description and Integration (UDDI), a global, platform-independent, open framework to enable businesses to discover each other, define their interaction, and share information in a global registry
  - Simple Object Access Protocol (SOAP), a lightweight protocol for exchange of information in a decentralized, distributed environment
  - eXtensible Markup Language (XML), which provides a common language for exchanging information.
- Enhanced Web Services, including WSIF, WS-Security, and a Technology Preview of JSR109. New and improved features in **Web services** support include:
  - An open source implementation of a Web Services Invocation Framework (WSIF), new in this release. It includes protocol isolation and dynamic invocation (no stubs)
  - The following new features added by WebSphere Application Server Network Deployment:
    - A private Universal Description, Discovery and Integration (UDDI) Registry, implementing Version 2.0 of the UDDI specification. See the *Migrating from the IBM WebSphere UDDI Registry on WebSphere Application Server 4.0* topic (twsu_migrate) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html, about migrating from the IBM WebSphere UDDI Registry PRPQ that ran in the previous product version.
    - A Web Services Gateway for providing gateway access to existing Web services. See the *Web Services Gateway - What is new in this release* topic (cwsg_new) in the IBM WebSphere Application Server, Version 5 InfoCenter, about the main differences between this version of the gateway and the Technical Preview version of the gateway that ran in the previous product version.
    - An additional Web services component, ″IBM WebSphere Web Services for J2EE Technology Preview.″ This is available for use with Version 5, as a separate download from: http://www7b.boulder.ibm.com/wsdd/downloads/techpreviews.html.

    The Web services technology preview supports emerging Java Web services standards like JAX-RPC and Web services for J2EE. It is recommended that new development efforts use the Web Services Technology Preview and follow these standards.
    - AXIS has improved performance and flexible architecture
  - Newly-enhanced Web services capabilities of WebSphere Studio (sold separately) for developing Web services and Web Services Gateway filters.

Applications > Application services

- Changes and improvements in **naming support** are described in the *New features for name space support* topic (rnam_new_features) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html. They include:
  - The way that the system binds objects into the name space has changed significantly.

    In previous product versions, all objects were bound relative to a single root context. Now they are bound to a context that is specific to the server associated with the object. This context is referred to as the server root context. Each server has its own server root context. An initial context can be any server root context. This means that jndiName values in deployment descriptors and lookup names in thin clients must be qualified when the object associated with the name is bound under a server root context different from the initial context.

    For more information, see the *Name space logical view* (cnam_name_space_partitions) and the *Lookup names support in deployment descriptors and thin clients* (rnam_names) topics in the IBM WebSphere Application Server, Version 5 InfoCenter.
  - In Version 3.5.x and Version 4.x, the Name Server runs in the same process as the administrative server. An administrative server is no longer running on every Version 5 installation. The Name Server configuration is included in the same configuration files as application servers. The Name Server runs in its own process.
- Changes and improvements in **dynamic caching** include:
  - Version 4 supported the configuration of dynamic servlet caching through the use of a `servletcache.xml` file. For migration purposes, this file is still supported by this release. In order to utilize the new and improved functionality of the dynamic cache service in this release, you must configure your cache policy using the new `cachespec.xml` format. For more information, see the *Configuring cacheable objects with the cachespec.xml file* topic (tprf_dynamiccacheconfig) in the InfoCenter.
  - Sophisticated dynamic network caching follows these directives, meaning explicit cache APIs are not needed:
    - Cache within the context of J2EE (such as Servlet, JSP, and EJB patterns)
    - Describe caching behavior in the form of XML cache policy files, providing a more flexible cache policy deployment descriptor

    Features of the dynamic caching engine include:
    - Disk overflow of cached objects through Java Object store/access (put and get)
    - Least Recently Used (LRU) management
    - XML cache policy management (such as the use of ID generation)

    - Invalidation management
    - Replication from one dynamic cache to another, using Replication Domains and their associated Replicator Entries
    - External cache support for caches such as:
      - IBM WebSphere Edge caching using Akamai ESI
      - IBM HTTP Server Fast Response Cache Accelerator (FRCA)

    Caching support includes:
    - Servlet and JSP results caching (same as Version 4.)
    - Command caching (new)
    - Pattern caching (new)
    - Web services caching (new)

    For more information, see the *Improving performance through the dynamic cache service* topic (tprf_dynamiccache) in the InfoCenter.
- Internationalization allows applications to become global by determining the client locale and changing all relevant attributes, such as currency, character sets, and so on.

  For more information, see the *Using the internationalization service* topic (tin_ep) in the InfoCenter.
- Classloaders are new and improved. For more information, see the *Classloading* topic (trun_classload) in the InfoCenter.
- User profile support is deprecated. For more information, see the *Managing user profiles* topic (tprs_sep2) in the InfoCenter.

Resources > Messaging

- Java Message Service (JMS) through embedded provider:
  – Supports point-to-point and publish/subscribe styles of messaging
  – Used for message-driven bean support
  – Integrated with transaction manager (JMS with XA)
  – Used for messaging within a cluster or cell
- Support for plugging in other JMS providers, including MQ Series
- Messaging and e-mail interfaces through JavaBeans Activation Framework (JAF), Remote Method Invocation over Internet InterORB Protocol (RMI/IIOP), JavaMail, and JMS with the help of IBM MQ Series
- Integrated Java Message Service, as described in the *Using asynchronous messaging* topic (tm_ep) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

Resources > Data access

- All connector access is through J2C.
  – JDBC access managed via J2 relational resource connector
  – Legacy JDBC support is provided
- Data access support provides a complete implementation of the JCA 1.0 specification, including support for:
  – Connection sharing

  This version fully supports the res-sharing-scope tag within the resource reference (resource-ref) element, so the product supports both shareable and unshareable connections.
  - Get/use/close and get/use/cache programming models for connection handles

    The product supports the Web Container. Both EJB and Web components can utilize the J2EE Connector Architecture.
  - XA, Local, and No Transaction models of resource adapters, including XA recovery
  - Security options A and C per the specification
  - Res-auth settings of either Application or Container.

    In Version 4, the res-auth setting was disregarded. That is, it was treated as if the value of res-auth was set to Application. If your existing applications had res-auth set to Container, you might get different behavior if you install them into the new environment without any changes.

  Applications must be packaged as J2EE 1.3 applications. For more information, see the *Migrating a version 4.0 data access application to version 5.0* topic (tdat_migdaapp) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
- Subpools were eliminated to provide better performance. You can no longer specify Pool and Subpool names. The Pool name is based on the data source or JNDI name of the connection factory.
- J2EE Connector Architecture or J2C is part of J2EE V1.3.

  J2C is similar to the Common Connector Framework (CCF) but is implemented for the Java platform. It provides specialized access to Enterprise Resource Planning (ERP) and mainframe systems such as CICS and IMS from IBM. As part of J2C, the product provides these components:
  - Common Client Interface API, which simplifies access to diverse back-end Enterprise Information Systems (EIS)
  - Resource Adapter, which enables the product to communicate with the back-end EIS. One-phase commit resource adapters are available for:
    - Host On-Demand
    - CICS
    - IMS
    - SAP
    - J.D. Edwards
    - PeopleSoft
    - Oracle Financial
  - Connection Factory, which connects an application to the Resource Adapter

Security

- Four administrative roles are now available for securing the administrative console. For more information, see the *Administrative console and naming service authorization* topic (csec_adminconsole) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
- Enhanced security features include:
  - JAAS

- CSIv2 interoperability
- Java2 security
- Support for third party Security Providers

For more information, see the *Welcome to Security* topic (welc_security) in the IBM WebSphere Application Server, Version 5 InfoCenter.

- Distributed Systems Management, Security, and Directory Support
- J2EE 1.3 security support, including JAAS programming model and CSIv2 for CORBA interoperability
- Web services security includes signatures and credential propagation
- Introducing support for third party security providers (prior to JSR 115)
- Version 4 users should consider migrating to the new UserRegistry interface, from the deprecated CustomRegistry interface that was introduced in Version 4.
- The Trust Association Interceptor interface remains backward compatible with that of Version 4
- The application login helper functions provided in Version 4 and prior releases are deprecated, but still supported.
- The login helper functions are replaced by the JAAS LoginContext and Subject based programming model in Version 5.

Environment

- Configurable plug-ins for popular Web servers

  The Web server (or HTTP server) plug-in enables communication between the HTTP server and the application server. It uses the industry-standard HTTP transport protocol for non-secure transports and HTTPS for secure transports.
- A single `install_root/config/cells/plugin-cfg.xml` file replaces the following `temp` directory files that told the HTTP server where to send Web requests in Version 4.0:
  - `rules.properties`
  - `hosts.properties`
  - `queues.properties`

  For more information, see the *Configuring Web server plug-ins* topic (trun_plugin) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
- For testing purposes, there are HTTP serving capabilities embedded within the product. For more information, see the *Configuring transports* topic (trun_plugin_transport) in the IBM WebSphere Application Server, Version 5 InfoCenter.
- New variable support. Variables are configuration properties that can be used to provide a parameter for any value in the system. For more information, see the *Variables* topic (crun_variable) in the InfoCenter.
- New shared library support, as described in the *Shared library files* topic (crun_sharedlib) in the InfoCenter.

System administration

- Terminology for distributed systems management:
  - A cell is a collection of machines that you are managing together
  - A node is a machine on which you are running an application server

  - A server is the Java virtual machine running the application server containing your applications
- New, scalable XML-based administrative infrastucture:
  - All configuration data is stored in XML for standard deployment descriptors, and XMI format for product-specific configuration documents. These documents are stored on each node. No relational database is required. See the *Working with server configuration files* topic (trun_data) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
  - WebSphere Common Configuration Model (WCCM) documented API is provided for manipulating product configuration files
  - Servers load directly off of documents.
  - Application binaries are managed as part of the configuration repository.
  - In clusters, the product manages synchronization of documents across machines. This feature is configurable, as described in the *File synchronization service settings* topic (urun_rsynchservice) in the IBM WebSphere Application Server, Version 5 InfoCenter.
- JMX support:
  - Multiple protocol support (SOAP by default, but also RMI/IIOP)
  - Support for alerts
  - Message routing between machines, providing cell-level view
  - Support for MBeans defined and registered by you
  - Runtime attributes and access to runtime operations, configurations, and performance data

  For more information, see the *Deploying and managing using programming* topic (txml_programming) in the InfoCenter.
- Scripting support:
  - Based on Bean Scripting Framework (BSF), supports multiple scripting languages
  - Parallel capabillity between scripting and Web-based administrative console (see below)
  - Interactive and script modes
  - Multiple connection styles (SOAP, RMI)
  - Remote administration support
  - Ability to access any MBean registered in any server in the cell
  - Runtime attributes and access to runtime operations, configurations, and performance data

  For more information, see the *Deploying and managing using scripting* topic (txml_script) in the InfoCenter.
- WebSphere Administrative Console:
  - Access to configuration data and to operations, runtime state
  - Multi-user support, with ability to customize based on user preferences. Coarse-grain administrative security control and filtering for roles such as Administrator, Monitor, Configurator, and Operator as described in the *Administrative console and naming service authorization* topic (csec_adminconsole) in the InfoCenter.
  - Filtering and search for collections

– Inclusion in all product packages — Version 4.0 Java-based console no longer available. Additional features are added as additional product packages are installed
– Struts and Tiles implementation
– Display of run time and configuration exceptions. You can toggle between them.

For more information, see the *Using the administrative console* topic (trun_console) in the InfoCenter.

- A stable of command line tools are now available for specific tasks. For more information, see the *Managing using command line tools* topic (txml_command) in the InfoCenter.

Monitoring and tuning performance

- The Performance Monitoring Infrastructure (PMI):
  – Is now integrated with Java Management eXtensions (JMX), as shown in the *Performance Monitoring Infrastructure* topic (cprf_pmidata) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
  – Has new counters including those for:
    - Dynamic caching
    - Workload management
    - Object Request Broker (ORB)
    - HTTP session size
    - JDBC time
    - CPU utilization

    For more information about data, see the *Performance data organization* topic (rprf_dataorg) in the IBM WebSphere Application Server, Version 5 InfoCenter.
  – Supports both the PmiClient interface and the JMX interface. The latter is supported through the AdminClient class described in the *Developing an administrative client program* topic (txml_develop) in the InfoCenter.
  – Continues to support the Version 4.0-style APIs, but the data hierarchy has been updated to match the Version 5 product structure
- Resource Analyzer has been rebranded as Tivoli Performance Viewer and bundled with WebSphere Application Server. New features include logging and replay in XML format, and CPU utilization.

  For more information, see the *Monitoring performance with Tivoli Performance Viewer (formerly Resource Analyzer)* topic (tprf_tpvmonitor) in the InfoCenter.
- Request metrics is instrumentation that tracks the time spent by selected requests in each WebSphere Application Server component in the system. The data can be written to a log file or sent to an ARM agent.

  For more information, see the *Measuring data requests (Performance Monitoring Infrastructure Request Metrics)* topic (tprf_requestmetrics) in the InfoCenter.
- Performance features include:
  – Dynamic, multi-tier caching, which is set up per node or application server using XML files and is most effective for non-user specific output such as mutual fund prices
  – Dynamic reloading of enterprise beans

- JNDI caching, which improves performance by caching expensive lookups. See the *JNDI caching* topic (cnam_naming_caching) in the InfoCenter.
    - Caching of dynamic content, such as servlets and JSP files, to improve throughput
- The product can be tuned from the WebSphere Administrative Console.

Troubleshooting

- Enhanced Problem Determination features, including FFDC (First Failure Data Capture):
    - Allows for collection of data based on the first failure in the system
    - Filters out expected or recurring exceptions to reduce overhead in collecting data
    - Passes data to an analysis engine that searches a knowledge base of information about common errors, including their possible causes and solutions

  For more information, see the *Working with troubleshooting tools* topic (ttrb_trbtls) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
- RAS Collector Tool gathers information to send to IBM Service personnel. For more information, see the *Collector Tool* topic (ctrb_ct) in the IBM WebSphere Application Server, Version 5 InfoCenter.
- Improved messages. To view message documentation, click **Messages** in the **Quick reference** view of the InfoCenter.

Chapter 3. Installing WebSphere Application Server Network Deployment

Perform the following tasks to install the IBM WebSphere Application Server Network Deployment product on your machine.

1. Plan to install an e-business network.

   This task helps you plan for installing an e-business network. It also describes interoperability considerations.
2. Install the product.

   This task helps you prepare your machine for installation and explains the different types of installation available to you. It includes information on:
   - Using the LaunchPad
   - Deciding whether to migrate applications and the configuration from a previous version
   - Deciding whether to coexist with a previous version
   - Using the silent installation method

   This task walks you through using the installation wizard. The installation wizard lets you migrate applications and configurations from a previous version of WebSphere Application Server or coexist with the previous version, choose a typical or custom installation, and perform some initial configuration.
3. Verify the WebSphere Application Server installation.
   a. Use the First Steps tool to run the Installation Verification Test.

      The First Steps tool starts automatically at the end of the installation.
   b. Identify and correct any problems using the troubleshooting procedure.
4. (Optional) Examine the results of migration, or prepare to perform a manual migration.

   This task describes exactly what is migrated during the automatic migration. It also describes how to perform a manual migration using the migration tools.
5. (Optional) Prepare to migrate from an unsupported operating system.

   This task describes a basic procedure to follow when migrating from one operating system to another, taking into account the possibility that you might have to format a drive, for example.
6. (Optional) Configure WebSphere Application Server after migration.

   This task explains how to check migrated application and configuration information, to understand and configure exactly what you migrated.
   - If you migrate from Version 3.x, examine the applications you are moving. Make any necessary changes to the applications, which were converted to Java 2 Platform, Enterprise Edition (J2EE) platform applications. The migration tools create the initial J2EE enterprise applications, based on Version 3.5.x configurations.
   - If you migrate from Version 4.x, there is little to review. J2EE 1.2 EAR files in Version 4 work in Version 5 of WebSphere Application Server, which also supports J2EE 1.3.
7. Set up a multinode environment.

This task describes how to set up multiple nodes into a group, or cell, of application servers, with centralized configuration under the control of the deployment manager node and its distributed node agents.

8. (Optional) Set up Version 3.5.x and Version 5 coexistence.

   This task describes running WebSphere Application Server Version 3.5.5 and later editions with Version 5, on the same machine.
9. (Optional) Set up Version 4.0.x and Version 5 coexistence.

   This task describes running WebSphere Application Server Version 4.0.2 and later editions with Version 5, on the same machine.
10. (Optional) Set up Version 5 coexistence.

    This task describes running multiple WebSphere Application Server Version 5 installations on the same machine.
11. (Optional) Federate multiple Version 5 installation instances.

    This step describes how to add multiple installation instances to a deployment manager cell.
12. (Optional) Automatically restart WebSphere Application Server server processes.

    This task describes how to set up WebSphere Application Server Versions 5 server processes to be monitored and restarted by the operating system.
13. (Optional) Create multiple Version 5 configuration instances on one machine.

    This task describes creating multiple configuration instances from one installation.
14. (Optional) Create servers in coexistence or multiple instance environments.

    This task describes creating multiple servers in a coexistence or multiple instance environment.
15. (Optional) Change HTTP transport ports.

    This task describes changing HTTP transport port settings to avoid conflicts in a coexistence or multiple configuration instances environment.

Included in this *Getting Started* guide is a description of how WebSphere Application Server stores product version and history information in a series of XML files in the installation root directory.

If you must uninstall the product, follow the steps in the Uninstalling WebSphere Application Server topic. If you must delete all files, including files that the uninstall wizard does not delete, perform a manual uninstall as described in the topic.

An Installation: Resources for learning section at the end of this document contains links to the Web sites mentioned throughout the document, and to others as well.

There are several WebSphere Application Server packages available.

WebSphere Application Server packages

The WebSphere Application Server family of interoperable products provides a next-generation application server on an industry-standard foundation. The IBM WebSphere Application Server family is divided into four packages for Version 5, to better address requirements you might have. Each edition addresses a distinct set of scenarios and needs. WebSphere Application Server includes:

- **WebSphere Application Server Express** (not pictured below)

This edition is a lightweight server for static content, servlets, and JSP pages, but does not support enterprise beans.

- **WebSphere Application Server**

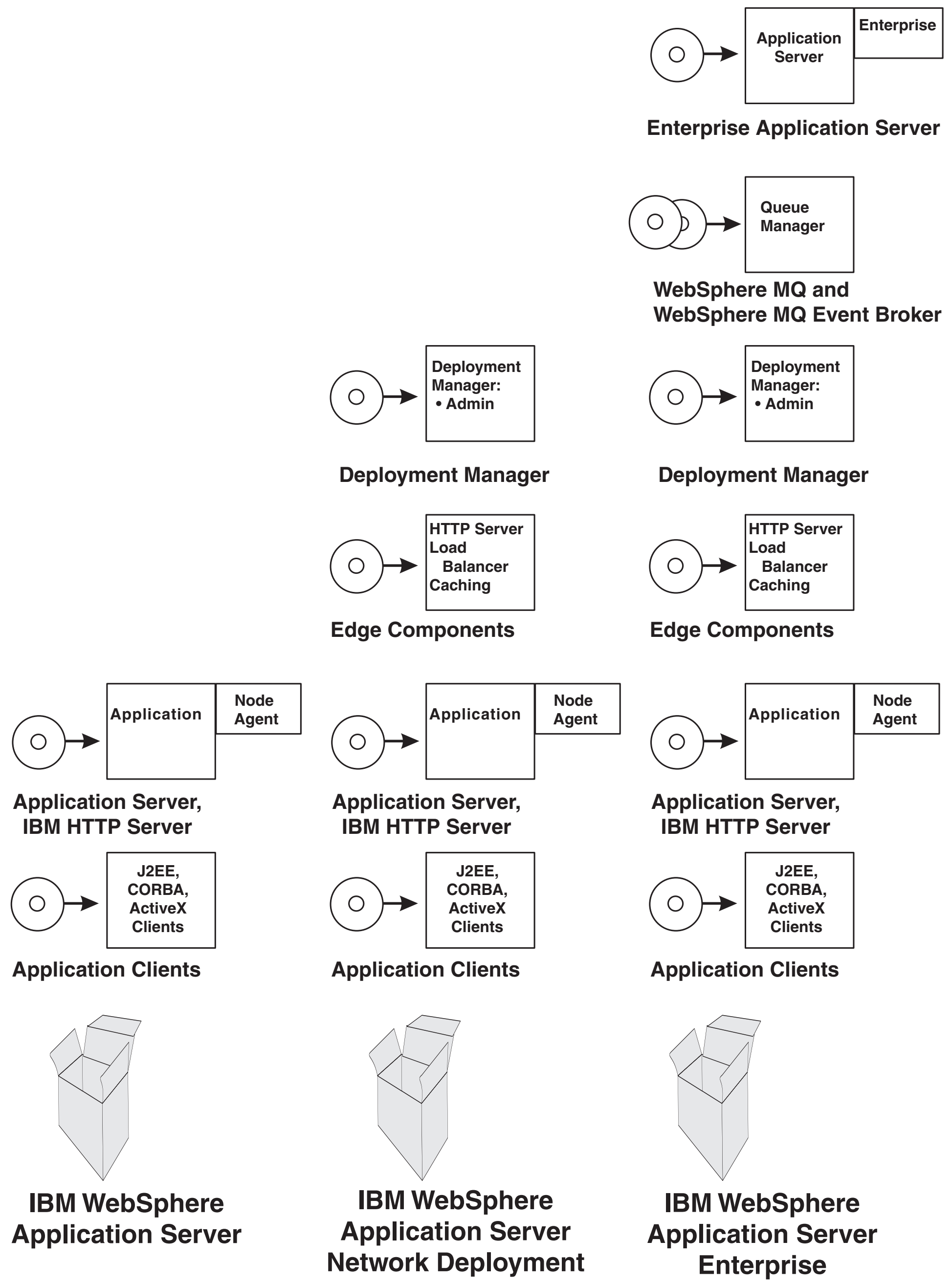
  This edition addresses the basic programming and execution needs of desktop developers and single-server production scenarios. The execution environment for this edition addresses standards-based programming for Web and component-based programming, as well as Web services.

  The administration model for this edition presumes a single-server environment - no clustering for failover or workload balancing, nor centralized administration of multiple server instances. However, you can add a standalone node to a centrally administered network (the cell) at any time after installing the next product, which controls the cell.
- **WebSphere Application Server Network Deployment**

  This edition addresses application server execution in departmental computing scenarios. It provides centralized administration of multiple server instances, as well as basic clustering and caching support.
- **WebSphere Application Server Enterprise**

  This edition addresses large information technology (I/T) production scenarios for applications that are designed according to the core, standards-based programming model of Java 2 Platform, Enterprise Edition (J2EE) and Web services. It supports large-scale clustering, caching, content distribution, and dynamic workload management to help gain efficient utilization of shared resources in the I/T computing center. It also addresses complex programming requirements for integrating applications and lines of business, by introducing super-standard programming functions for business process management, integration adapters, rules management, and message transformations.

Installation images in WebSphere Application Server packages



Application Server
Enterprise
Enterprise Application Server
Queue Manager
WebSphere MQ and WebSphere MQ Event Broker
Deployment Manager: • Admin
Deployment Manager
Deployment Manager: • Admin
Deployment Manager
HTTP Server Load Balancer Caching
Edge Components
HTTP Server Load Balancer Caching
Edge Components
Application
Node Agent
Application Server, IBM HTTP Server
Application
Node Agent
Application Server, IBM HTTP Server
Application
Node Agent
Application Server, IBM HTTP Server
J2EE, CORBA, ActiveX Clients
Application Clients
J2EE, CORBA, ActiveX Clients
Application Clients
J2EE, CORBA, ActiveX Clients
Application Clients
IBM WebSphere Application Server
IBM WebSphere Application Server Network Deployment
IBM WebSphere Application Server Enterprise

The WebSphere software platform for e-business starts with a foundation formed from Web application serving and integration. The WebSphere Application Server family lets you quickly, reliably and flexibly enable your business for the Web. It provides the core software to deploy, integrate and manage your e-business applications. WebSphere Application Server supports custom-built applications, based on integrated WebSphere software platform products, or on other third-party products. Such applications can range from dynamic Web presentations to sophisticated transaction processing systems.

IBM WebSphere Application Server, WebSphere Application Server Network Deployment, and WebSphere Application Server Enterprise are interoperable building blocks that build upon each other.

- The **Clients** installation image contains support for Java (J2EE) Application client 2, CORBA and ActiveX client run times.
- The **WebSphere Application Server** (base) product installation image contains the core application server runtime, a native JMS provider (the Embedded Messaging feature), IBM HTTP Server, IBM Developer Kit, IBM Cloudscape, XML and XSL parsers, the Application Assembly Tool (AAT), the deployment tool, the node agent for communicating to the deployment manager when it is part of a cell, and the external adapter library for proxy caching enablement.
- The **Deployment Manager** product installation image contains the deployment manager configured for use in departmental production computing scenarios.
- The **Edge components** installation image contains IBM HTTP Server, and edge of network support for the Load Balancer (Dispatcher) and Caching Proxy (edge caching), as well as support for network authentication and single sign-on.
- The **Enterprise** installation image contains programming model extensions to the core application server, such as Business Rule Beans and Business Policy Management, and deployment manager extensions for administering functions included in the programming model extensions. It also contains the **WebSphere Application Server** (base) product installation image, to support an *umbrella installation*, where both products are installed in the same installation procedure.
- The **WebSphere MQ and WebSphere MQ Event Broker** installation images provide the non-embedded full-function MQ Series Queue Manager for reliable, dynamically load balanced asynchronous messaging for more than 35 platforms.

Also included with all packages are the Data Direct Technologies JDBC Drivers and the Application Server Toolkit CD-ROMs.

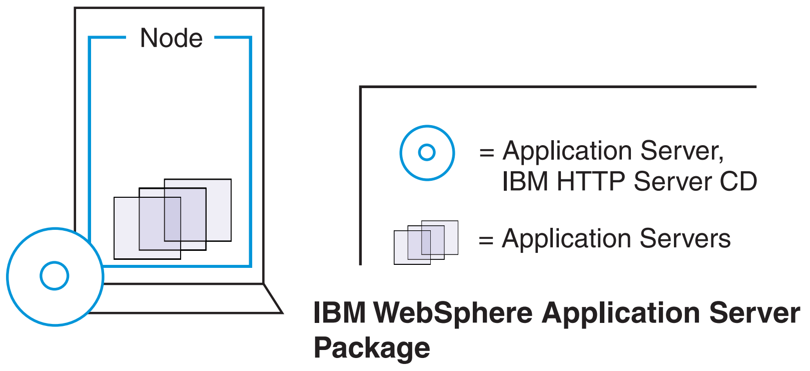
In addition, the WebSphere Application Server Network Deployment and Enterprise packages include the IBM Directory Server and the IBM DB2 Universal Database Enterprise Edition products.

The ultimate goal of selecting a WebSphere Application Server package is to create an e-business network.

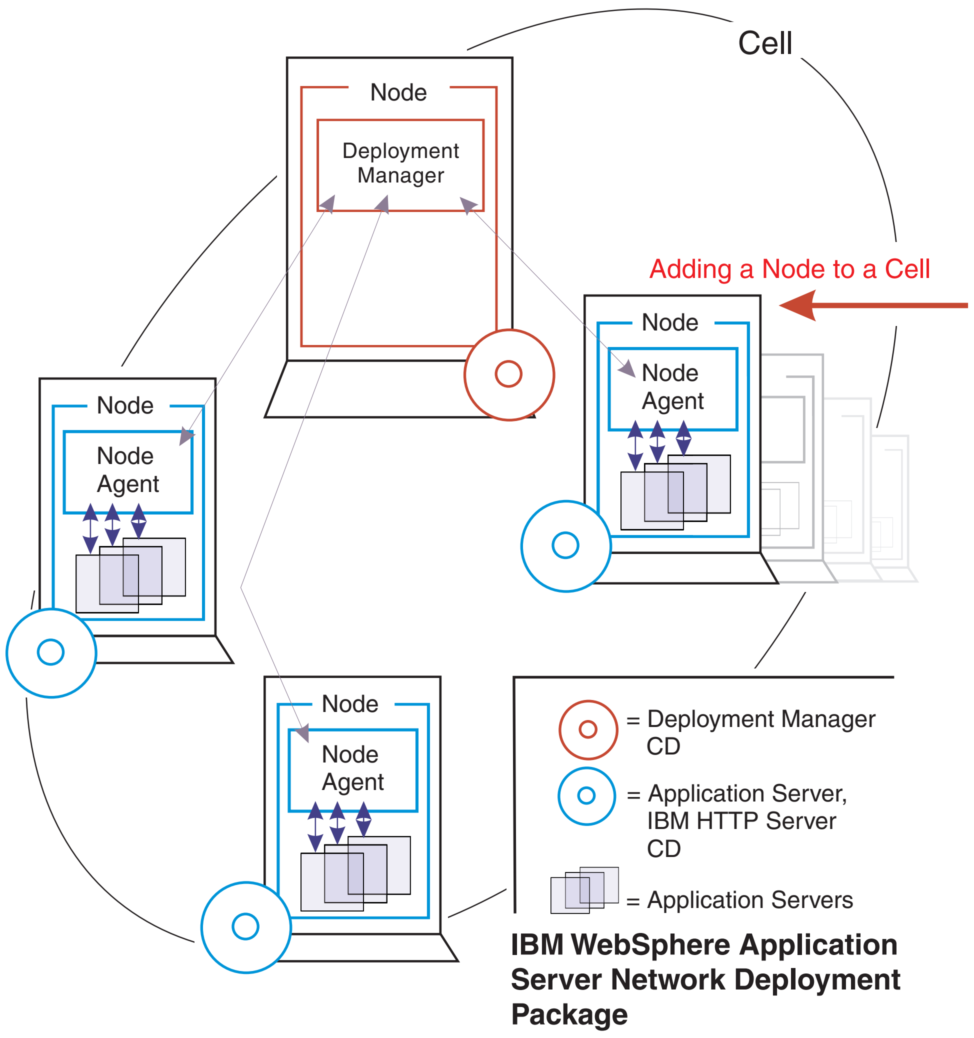
Planning to install an e-business network

Version 5 of the WebSphere Application Server family provides flexible configurations and deployment options for hosting Java 2 Platform, Enterprise Edition (J2EE) applications. The WebSphere Application Server package provides all required components:

- The WebSphere Application Server
- A Web server
- A J2EE client


Node
= Application Server, IBM HTTP Server CD
= Application Servers
IBM WebSphere Application Server Package

The WebSphere Application Server Network Deployment package includes the core package and provides a smooth transition to deploying and managing applications in a distributed network environment. It includes productivity and scalability enabling components (Edge Components). The deployment manager server lets you easily federate single nodes to its group, which it manages as a single image or *cell*, by using node agents on each node.



Tasks in the installation planning process include:

1. Plan the general scope of your network.

   Review single machine topologies and how to establish a multimachine environment to learn about WebSphere Application Server, Version 5 product family scalability. With one command you can move an application server node from a standalone environment into a managed group of server processes, all part of a single-image cell, under the centralized control of the Version 5 deployment manager. Or, you can move an application server back to a standalone environment just as easily. You can start small and *scale up* to larger integrated topologies easily, with the seamless architecture and packaging of the WebSphere Application Server, Version 5 product family.
2. ″Setting up a multinode environment″

   Determine which scalable and reliable e-business environment is a best fit for your requirements as you learn where to install the typical components of a WebSphere Application Server environment. For example, decide whether to install WebSphere Application Server on the same server as the Web server, or whether to create a cluster of application servers on one or several machines.

   Decide which WebSphere Application Server package you need. Understanding scalability factors and picking the right design can help you start with the correct package, and can help you achieve your business model objectives for e-business in a timely and cost-efficient manner.
3. Review common topologies for WebSphere Application Server Edge Components, as described in the InfoCenter for WebSphere Application Server Edge components at http://www-3.ibm.com/software/webservers/appserv/ecinfocenter.html.

   After planning how to install the WebSphere Application Server product, you can plan the installation of Edge Components, which are included in the Network Deployment package and therefore, in the Enterprise package. Edge Components include the Caching Proxy and the Load Balancer component set, which includes components such as the Dispatcher.
4. **(Optional)** Review the latest research in e-business patterns at the developerWorks Web site at http://www.ibm.com/developerworks/patterns/index.html. You can find additional information on proven patterns of machine and product configurations, from thousands of actual IBM experiences.
5. Plan for interoperability and coexistence.

   Plan to have WebSphere Application Server interoperate with your other e-business systems, including other versions of WebSphere Application Server. *Interoperability* provides a communication mechanism for WebSphere Application Server nodes that are at different versions. *Coexistence* is another term in common use in this InfoCenter. It describes multiple versions or instances running on the same machine, at the same time.

   Interoperability support enhances migration scenarios with more configuration options. It often is convenient or practical to interoperate during the migration of a configuration from an earlier application server version to a later one when some machines are at the earlier version and some at the later one. The mixed environment of machines and application components at different software version levels requires interoperability and coexistence.

   It is often impractical, or even physically impossible, to migrate all the machines and applications within an enterprise at the same time. Understanding multiversion interoperability and coexistence is therefore an essential part of a migration between version levels.

   a. Plan to run WebSphere Application Server across platforms.

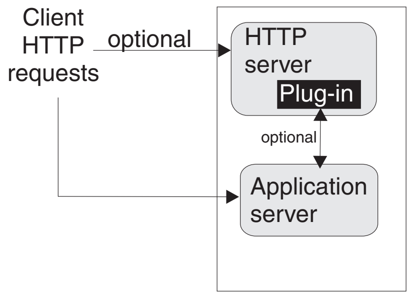
Support of multiple versions is provided on all operating system platforms supported by WebSphere Application Server, Version 5.

b. Plan to run WebSphere Application Server across versions.

   WebSphere Application Server, Version 5 is generally interoperable with WebSphere Application Server Versions 3.5.x and 4.0.x, although each version has specific requirements. However, the ability to run different versions of an application server in a configuration does not let you include Version 5 application servers in an existing administrative domain, or let you include Version 3.5.x or Version 4.0.x application servers in a Version 5 cell.

Single server topology

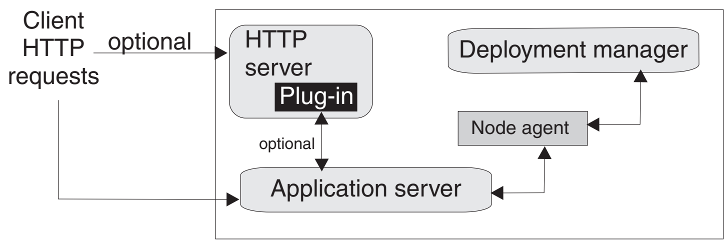
The following illustrations show examples of single server topologies. Each WebSphere Application Server product can run in a single server environment. The most common topology is a standalone base WebSphere Application Server product.



The WebSphere Application Server base product runs on a single machine. You can install the product in a standalone configuration or as part of a cell in a multimachine configuration. The standalone configuration is typically for developer desktops or standalone production computing, which involve a single application server instance operating independently of any other applications.

You can install an application server installation image on any supported machine. You can use the application server to host one or more applications, configuring it through the administrative console.

The Network Deployment product can also run on a standalone machine or in a multimachine configuration, as described in other topology topics. The following illustration shows a typical developer environment for a Network Deployment product in a standalone configuration. This configuration is not recommended for a production environment unless you have a machine with the capacity for both products.

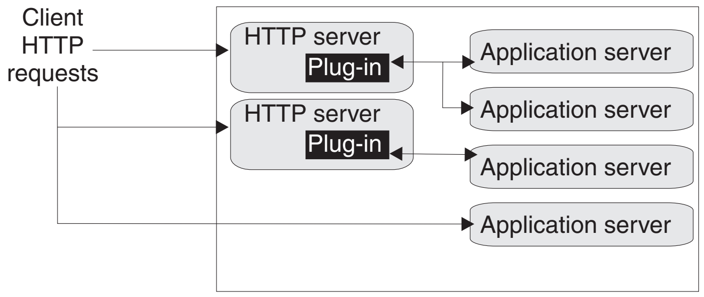


Multiple instances on one machine

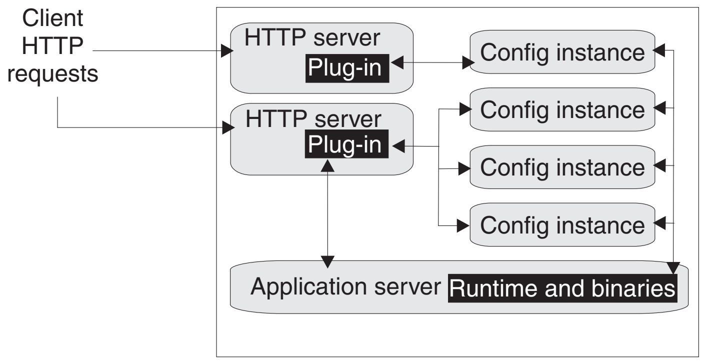
There are three main topologies for the base WebSphere Application Server product:

- A single installation, as described above
- Multiple installations in a coexistence environment
- A single installation with multiple configuration instances

You can install the base WebSphere Application Server product multiple times in separate directories. Each resulting installation instance is a fully functional application server. The following illustration shows an example of such a configuration.



You can also install the base WebSphere Application Server product one time and use the **wsinstance** command to create multiple configuration instances. Configuration instances are fully functional application servers that share the run-time and command binaries of the initial product installation. The following illustration shows an example of such a configuration.



Establishing multimachine environments

Setting up a Network Deployment environment involves several steps performed on each of the computers that comprise the cell.

1. Secure your environment before installation, as described in the online*Securing your environment before installation* (rsec_preinstall) topic, in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
2. (Optional) Start the administrative server from an earlier version of WebSphere Application Server.

If you intend to migrate the applications and configuration from the earlier version, start the administrative server so that the installation wizard can export the configuration as it performs the automated migration that you can select in the next step.

Start the administrative server of WebSphere Application Server Standard Edition (Version 3.5.x) or WebSphere Application Server Advanced Edition (Versions 3.5.x or 4.0.x).

It is not necessary to start the administrative server for WebSphere Application Server Advanced Single Server Edition, Version 4. The migration tools use the XMI configuration files directly.

3. Install the base WebSphere Application Server product on each system that is to host an application server node.

   You must install the base WebSphere Application Server product on a machine for the machine to become a node in the cell. The installation procedure is the same for a node that is to be federated into a cell as it is for a standalone application server. You can install the base WebSphere Application Server product more than once on a single machine. Coexistence is supported. There are several ways to federate a standalone application server installation instance into the deployment manager cell.

   You can also install the base WebSphere Application Server product once and create multiple configuration instances on the machine, using the **wsinstance** command. You can also install the Network Deployment product once and create multiple configuration instances of the deployment manager. Each deployment manager configuration instance can federate standalone base WebSphere Application Server product installation instances, but a deployment manager cannot federate base product configuration instances.

   Migrate applications, security settings, and the remaining configuration from WebSphere Application Server, Version 3.5.x and later, or WebSphere Application Server, Version 4.0.x and later. You can also choose to coexist with WebSphere Application Server, Versions 3.5.5 and later, or Versions 4.0.2 and later. Both migration and coexistence are described in the installation procedure for the base WebSphere Application Server product, which is available from the InfoCenter for the WebSphere Application Server product.

   Stop the administrative server from the earlier version before you perform the installation verification test, which starts the new server. This will avoid potential port conflicts.

4. Install the WebSphere Application Server Network Deployment product on a machine. This system now hosts the deployment manager.

   Only one system hosts the deployment manager. As it federates base WebSphere Application Server nodes, it expands the cell that it manages. Although you can install a base WebSphere Application Server on the same machine as the deployment manager, it is not generally done unless you have a machine with the capacity to host both products. The deployment manager is the central administrative manager. It does not install the base WebSphere Application Server product on other machines. You must do that separately. The only functions supported in the Network Deployment installation are the deployment manager and its associated administrative programs.

   Migrate applications, security settings, and the remaining configuration from WebSphere Application Server, Version 3.5.x and later, or WebSphere Application Server, Version 4.0.x and later. You can also choose to coexist with WebSphere Application Server, Versions 3.5.5 and later, or Versions 4.0.2 and later. This is described in the installation procedure for the WebSphere Application Server Network Deployment product.

5. Start the deployment manager process. One way is using the **startManager** command line tool as described in the InfoCenter in the ″startManager command ″ (rxml_startmanager) topic.

   Run the startManager command line tool from the `/bin` directory of the deployment manager installation.

   There are two methods to use to start the deployment manager. You can start it as a monitored process, which restarts automatically if a failure occurs. You can also start it as an unmonitored process, which is what the **startManager** command does. For production systems, running the deployment manager as a monitored process is recommended.
6. Run the **addNode** command on every node that you plan to federate into the cell.

   The addNode command line utility (rxml_addnode in the InfoCenter) incorporates a base WebSphere Application Server product node into a deployment manager cell. You must run this tool on every system that you plan to make part of a Network Deployment cell. There are several parameters for the **addNode** command, but the most important are includeapps, host name, and the SOAP Connector Address port. The SOAP connector address port identifies the deployment manager that instantiates the node agent.

   Alternatively, you can use the administrative console of the deployment manager (trun_console in the InfoCenter) to add running application server nodes to the cell.
7. Enable the appropriate level of security after the installation is complete.

   See the *Securing your environment after installation* (rsec_postinstall) topic in the InfoCenter.
8. Develop and unit test new applications.

   Load existing applications into your development environment and debug them. See the *Developing* (welc_developing) topic in the InfoCenter.
9. Assemble code into a main application module or EAR file.

   See the *Assembling or packaging* (welc_assembling) topic in the InfoCenter.
10. Start all servers in the test environment.

    See the *Starting servers* (trun_svr_start) topic in the InfoCenter.
11. Deploy your applications in the test environment.

    See the *Deploying* (welc_deploying) topic in the InfoCenter.
12. Test all applications thoroughly.

    See the *Testing* (welc_testing) topic in the InfoCenter.

    Follow normal test procedures as you move the test environment into production. Review the information in the ″Migrating″ (welc_migrating in the InfoCenter) topic to understand what you must look for. In particular, review the table at the end of the topic that links you to specific recommendations and practices. You must configure your migrating applications to ensure that they migrate in the manner that you intend.
13. Prepare and monitor the environment into which you deploy applications.

    See the *Administering* (welc_managing) topic in the InfoCenter.
14. Adjust application code, configurations, and system settings to improve performance.

    See the *Tuning* (welc_tuning) topic in the InfoCenter.
15. Fix any known problems.

See the *Troubleshooting or problem determination* (welc_troubleshooting) topic in the InfoCenter.

16. Set up your production system by configuring all server processes to be monitored by their operating systems.

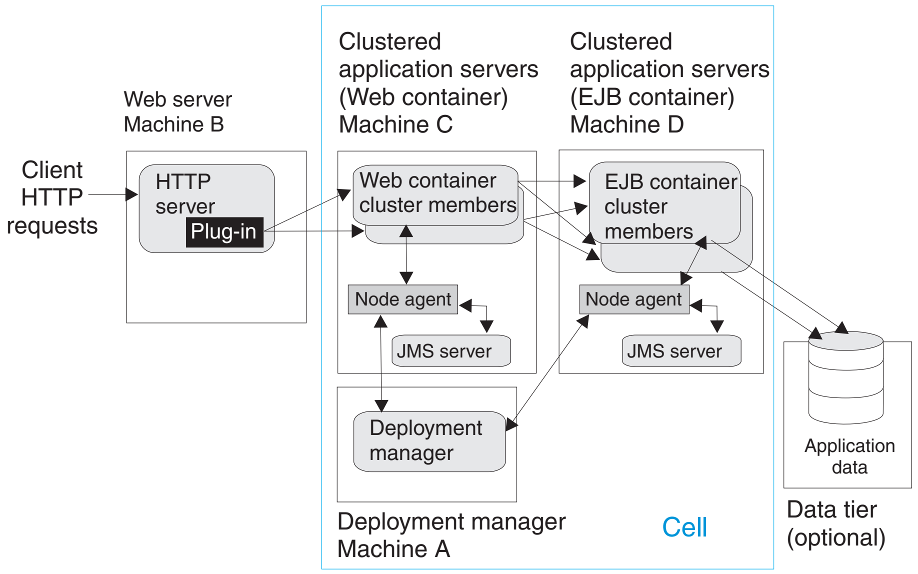
You now have an understanding of what it takes to create a working WebSphere Application Server cell.

You can use the administrative console (trun_console in the InfoCenter) or other administrative tools (welc_configop) to observe and control the incorporated nodes, and the resources on those nodes. The console provides a central location for configuring, monitoring, and controlling all of the application servers on the various nodes within the cell.

Setting up a multinode environment

A multiple node environment places WebSphere Application Server processes on separate physical machines, under the central management of the deployment manager process, which groups application servers into its managed ***cell***. This example topology shows how you can have a tier of application servers within the cell in addition to traditional tiers that can contain the Web server, databases, enterprise information systems, and other types of persistent storage.

The following illustration shows an example of multiple nodes and multiple tiers.


Client HTTP requests
Web server Machine B
HTTP server
Plug-in
Clustered application servers (Web container) Machine C
Web container cluster members
Node agent
JMS server
Clustered application servers (EJB container) Machine D
EJB container cluster members
Node agent
JMS server
Deployment manager
Deployment manager Machine A
Cell
Application data
Data tier (optional)

In this example, Web container cluster members (machine C) are closer (in a network sense) to the HTTP server (machine B), which improves their response to client requests. Application server processes that run enterprise beans (machine D) are closer in network terms to application data, which is represented in an application by entity beans and stored in a database on the data tier.

Clustered application servers on Machines C and D help maximize resource use on each machine. Although the example shows two cluster members in each node, you can have as many as a machine can support. Cluster members in a multi-tiered topology provide process redundancy and use memory more efficiently than in similar topologies that host only single instances of application servers. Additional resources on the machines can improve application throughput and performance.

Introducing firewalls between each pair of tiers can provide the same level of security for entity beans as for application data.

A multi-tiered topology within the cell eliminates local Java Virtual Machine (JVM) optimizations that occur when the same cluster member runs both the Web container and the EJB container. This topology also introduces network latency, which tends to slow system performance. Although multi-tiered topologies provide more redundancy for application server processes, they also introduce more possible points of failure. The level of redundancy can also complicate maintenance.

The deployment manager on machine A manages the configuration of cluster members and other application server processes on machines C and D. The deployment manager coordinates all application server processes through node agent processes, which run on each node.

Setting up environments, such as those shown in the illustrations, involves performing several steps on each computer comprising the deployment manager cell.

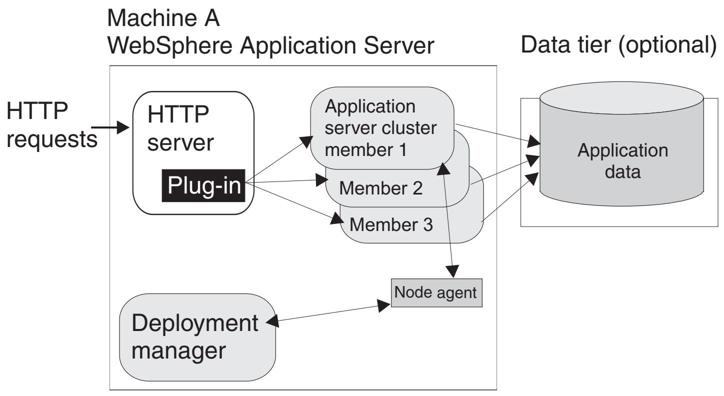
1. Install the Network Deployment product on machine A, to make it the deployment manager node, using the product CD-ROM labelled, **Deployment Manager**, from the product package. Launch the **Install.exe** program (**install.sh** on Linux/390, **install** on other Linux or UNIX platforms) on the platform root directory.

   Machine A is the system for the deployment manager server (dmgr), which provides a centralized administrative console for the entire group, or cell, of application servers that it controls. Installing the Network Deployment product does not install the base application servers. The Network Deployment installation installs only the deployment manager process and its associated administrative programs. Install the Network Deployment product on only one computer system from the set that is to make up your administrative cell.
2. Install the base WebSphere Application Server product on machines C and D, to make them application server nodes, using the product CD-ROM labelled, **Application Server, IBM HTTP Server**, from the product package. Launch the **Install.exe** program (**install.sh** on Linux/390, **install** on other Linux or UNIX platforms) on the platform root directory.

   Aside from the one computer where you installed the deployment manager node, you must install the base WebSphere Application Server product on other computers that are to comprise the cell. The installation process is the same for an application server (server1) that you federate into a cell as it is for a standalone application server.
3. Start the deployment manager process. The startManager command line tool is one way.
4. Run the addNode command line tool on every node that you intend to incorporate into the cell.

Vertical scaling topology

Vertical scaling refers to setting up multiple application servers on one machine, usually by creating cluster members.



This topology illustrates a simple vertical scaling example, with multiple cluster members of an application server on Machine A. You can also implement vertical scaling on more than one machine in a configuration. Combine vertical scaling with other topologies to boost performance and throughput.

Typical use

Vertical scaling offers the following advantages:

- **Increased processing power efficiency.** 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 process associated with each application server process. This correspondence avoids concurrency limitations and lets the application server use the full processing power of the machine.
- **Load balancing.** Vertical scaling topologies can use the WebSphere Application Server workload management facility.
- **Process failover.** A vertical scaling topology also provides failover support among application server cluster members. If one application server instance goes offline, the other instances on the machine continue to process client requests.

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.

Instructions

To set up a vertical scaling topology, use the administrative console to configure a set of application server cluster members that reside on the same machine.

It is recommended that you plan vertical scaling configurations ahead of time. However, because vertical scaling does not require any special installation steps, you can implement vertical scaling whenever it is needed.

While you are deciding how many cluster members to create on a machine, take these factors into account:

- **The design of the application.** Applications that use more components require more memory, limiting the number of cluster members you can run on a machine.

- **The hardware environment.** Vertical scaling works best with plenty of memory and processing power. Eventually there is a point of diminishing returns on any machine, where the overhead of running more cluster members cancels out the benefits of adding them.

The best way to ensure good performance in a vertical scaling configuration is to tune a single instance of an application server for throughput and performance, then incrementally add cluster members. Test performance and throughput as you add each cluster member. Always monitor memory use when you are configuring a vertical scaling topology, so you do not exceed available physical memory on the machine.

Multimachine topology concepts

Multiple machine environments extend basic single machine WebSphere Application Server configurations by distributing the application server over multiple machines, increasing the overall processing power from one machine to contributions from all machines in the configuration.

The flow of data in a WebSphere Application Server environment starts with a Web server receiving requests and routing them to the application server for processing. A WebSphere Application Server node stores administrative configuration data in XML files. A database can hold application data for applications that require a place to store data, such as user session information. There are also one or more administrative clients, such as the administrative console, for manipulating configuration data.

Some of the reasons for creating WebSphere Application Server applications that run on multiple machine systems include:

- **Scalability**. Adding more machines increases processing power, which scales up the system to handle a higher client load than that provided by a basic, single machine configuration. Scalability pertains to the capability of a system to adapt readily to a greater or lesser intensity of use, volume, or demand. For example, a scalable system can efficiently adapt to work with larger or smaller networks performing tasks of varying complexity. Ideally, it is possible to handle any given load by adding more servers and machines, assuming each additional machine processes its fair share of client requests. Each machine should process a share of the total system load that is proportional to the processing power of the machine.

  Consider these primary scalability factors when adopting a WebSphere Application Server topology:

  – **Security**. Address certain security concerns by physically separating the Web server from the application server, by using firewalls.

  – **Performance**. Maximize performance by ensuring the response time for transactions is as short as possible. You can use two general topologies to improve transaction performance:

    - *Vertical scaling*, in which you create additional application server processes on a single physical machine. The *Vertical scaling sample topology* topic describes a physical implementation for vertical scaling.

    - *Horizontal scaling*, in which you create additional application server processes on multiple physical machines to take advantage of the additional processing power available on each machine. You can also use WebSphere Application Server Edge Components, such as the Caching Proxy Edge component, and the Load Balancer component set (which includes the Dispatcher component), to implement horizontal scaling. The *Horizontal scaling with cluster members sample topology* and *Horizontal scaling* *with Network Dispatcher sample topology* topics describe physical implementations for horizontal scaling.

– **Throughput**. Add application server clusters to scale vertically or horizontally. Application server clusters can increase the number of concurrent transactions that the application can perform, to help process as many transactions as possible within a given time period.

– **Availability and failover support**. Avoid a single point of failure and maximize system availability by ensuring that the topology has 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 becomes a point of failure.

  A Dispatcher server performs intelligent load balancing to determine where to send a TCP/IP request. It can direct client HTTP requests to available Web servers, bypassing any that are offline. Another server can back up the Dispatcher server, to eliminate it as a single point of failure. Workload management of application servers and administrative servers also improves availability and failover support.

  Failover support distributes client requests to the remaining servers, which ensures continued client access without significant interruptions. (In practice, failover is not entirely transparent to clients.)

  WebSphere Application Server supports these methods of ensuring availability with multiple machines and applications:

  - **Multiple tiers** The components of an application (the Web server, application servers, databases, and so forth) are physically separated on different machines.
  - **HTTP server separation** The Web (HTTP) server is located on a different physical machine than the application server. You can redirect requests to application servers through a variety of methods.
  - **Demilitarized zone (DMZ)** Firewalls create demilitarized zone machines, which are isolated from both the public Internet and other machines in the configuration. The DMZ scales security processing, which improves security and throughput in the application environment.
  - **Multiple cells** Multiple WebSphere Application Server cells provide failover for clustering application servers and deploying applications.
  - **Multiple applications** Multiple application instances can be on the same physical machine, or on more than one machine in the cell.

– **Maintainability**. Understand that the topology affects the ease with which you can update system hardware and software. For instance, using multiple WebSphere Application Server deployment manager cells or horizontal scaling can make a system easier to maintain because you can take individual machines offline without interrupting other machines, running the application.

  Maintainability sometimes 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.

– **Maintaining session state between client HTTP requests**. Consider that session state is important for stateful applications, or for applications that run on multiple machines or application server instances. You can share a session between multiple application server processes (cluster members) by saving the session state to a database. In addition, configuring a network dispatcher affects how the session state is maintained. This consideration does not apply if your application runs on a single application server instance or is completely stateless.

- **Shared data access**. Placing backend resources, such as databases, on different machines provides ease of use when sharing these resources.
- **Fault isolation**. Providing more robust failover support through a configuration that includes a degree of fault isolation, reduces the potential for failure of one server to affect other servers. Configurations that provide simple failover support are concerned only with individual server failures that have no effect on the performance of other servers. However, in some situations, a malfunctioning server can create problems for other servers that are otherwise functioning normally. For example, a failing server can consume more than its share of system and database resources, preventing other servers from gaining adequate access to these resources. You can configure WebSphere Application Server to provide fault isolation between different parts of a system.
- **Dynamic changes to configurations**. Modifying the system configuration without interrupting its operation enhances the manageability and flexibility of the system. For instance, administrators can add or remove cluster members to handle variations in the client load, change server characteristics and propagate the changes to its cluster members, temporarily stop servers for maintenance, and so forth.
- **Mixed application server versions**. Migrating a few machines and applications at one time is possible on certain multiple machine configurations. In addition to Version 5 application servers, your server configuration can include earlier version application servers, where you currently deploy all applications you intend to migrate. You can migrate applications to Version 5 and deploy them in stages. You can also easily upgrade system hardware and software. When combined with the ability to make dynamic changes to the configuration, you can use a configuration of mixed application server versions to upgrade an application or machine without any interruption of service.

  Note: The ability to run different versions of an application server in a configuration does not let you include Version 5 application servers in an existing administrative domain, nor does it let you include Version 3.5.x or Version 4.0.x application servers in a Version 5 cell.

You can configure multiple machines to add processing power, improve security, maximize availability, and balance workloads. WebSphere Application Server Network Deployment and WebSphere Application Server Edge Components provide clusters, workload management, and the Dispatcher to implement configurations that address these issues. These scaling techniques are generally combined to maximize benefits and minimize problems associated with multiple machine systems.

Here is an overview of various ways you can scale up the basic, single machine WebSphere Application Server system to meet the needs of your organization. It is not intended as an exhaustive discussion of WebSphere Application Server configurations.

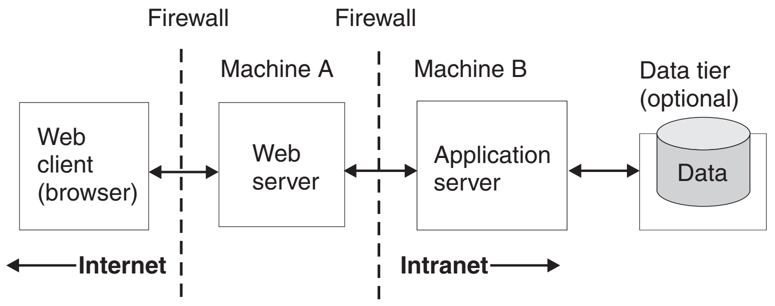
- **Clusters**. Clusters are identical multiple application server copies. A cluster member is one application server in a cluster. The configuration of each cluster member is based upon the application server that you copied to create the cluster member. You can create all cluster members on the same physical machine, or on different machines. Using cluster members can improve the performance of a server, simplify its administration, and enable the use of workload management.

- **Workload management (WLM)**. WLM enables both load balancing and failover, improving the reliability and scalability of applications deployed on the WebSphere Application Server server. Incoming processing requests from clients are transparently distributed among the cluster members of an application server.
- **Dispatcher**. The Dispatcher transparently redirects incoming HTTP requests from Web clients to a set of Web servers. Although the clients behave as if they are communicating directly with a given Web server, the Dispatcher is actually intercepting all requests and distributing them among all the available Web servers in the cluster. The Dispatcher can provide scalability, load balancing, and failover for Web servers.

Keep in mind that the techniques described above are not mutually exclusive. You can combine their basic elements in various ways.

Firewalls and demilitarized zone configurations: *Firewalls* protect backend resources, such as databases in multiple machine systems. You can also use firewalls to protect application servers and Web servers from unauthorized outside access.

A *demilitarized zone* (DMZ) configuration involves multiple firewalls that add layers of security between the Internet and critical data and business logic. A wide variety of topologies are appropriate for a DMZ environment. Although WebSphere Application Server provides great flexibility in configuring DMZ topologies, the basic locations of elements in a simple DMZ topology follow:



The main purpose of a DMZ configuration is to protect the business logic and data in the environment from unauthorized access. A typical DMZ configuration includes:

- An outer firewall between the public Internet and the Web server or servers processing the requests originating on the company Web site.
- An inner firewall between the Web server and the application servers to which it is forwarding requests. Company data also resides behind the inner firewall.

The area between the two firewalls gives the DMZ configuration its name. Additional firewalls can further safeguard access to databases holding administrative and application data.

Comparison of DMZ configurations

Somehow, requests for applications that WebSphere Application Server manages must get from the Web server to the application servers, passing through firewalls. You can implement DMZ configurations for a wide variety of multi-tiered systems. WebSphere Application Server offers many configuration choices for accomplishing this goal. The following table summarizes benefits of each DMZ configuration option supported by the product. Criteria for each topology are described after the table.

An **X** represents an advantage.

Benefit (X) or statistic	Remote HTTP	Reverse proxy
Compatible with product security	X	X
Avoids data access from DMZ	X	X
Supports Network Address Translation (NAT)	X	X
Avoids DMZ protocol switch		X
Allows encrypted link between Web server and application server	X	Depends on Web server
Avoids single point of failure	X	
Minimum firewall holes	One per application server, plus one if WebSphere Application Server security is used on the Web server machine	One

- **Works with product security.** WebSphere Application Server security protects applications and their components, by enforcing authorization and authentication policies. Configuration options compatible with product security are desirable because they do not necessitate alternative security solutions.
- **Avoids critical business data in the DMZ.** A DMZ configuration protects application logic and data, by creating a buffer between the public Internet Web site and the internal intranet, where application servers and the data tier reside. Desirable DMZ topologies do not have databases or application servers with critical business data in the DMZ.
- **Supports Network Address Translation (NAT).** 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.
- **Avoids the DMZ protocol switch.** The Web server sends HTTP requests to application servers behind firewalls. It is simplest to open an HTTP port in the firewall to let the requests through. Configurations that require switching to another protocol, such as IIOP, and opening firewall ports corresponding to the protocol, are less desirable. They are often more complex to set up, and the protocol switching overhead can impact performance.
- **Allows an encrypted link between Web server and application server.** Configurations that support encryption of communication between the Web server and application server reduce the risk that attackers are able to obtain secure information by *sniffing* packets sent between the Web server and application server. A performance penalty usually accompanies such encryption.
- **Avoids a single point of failure.** A point of failure exists when one process or machine depends on another process or machine. A single point of failure is

especially undesirable because if the point fails, the whole system becomes 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.

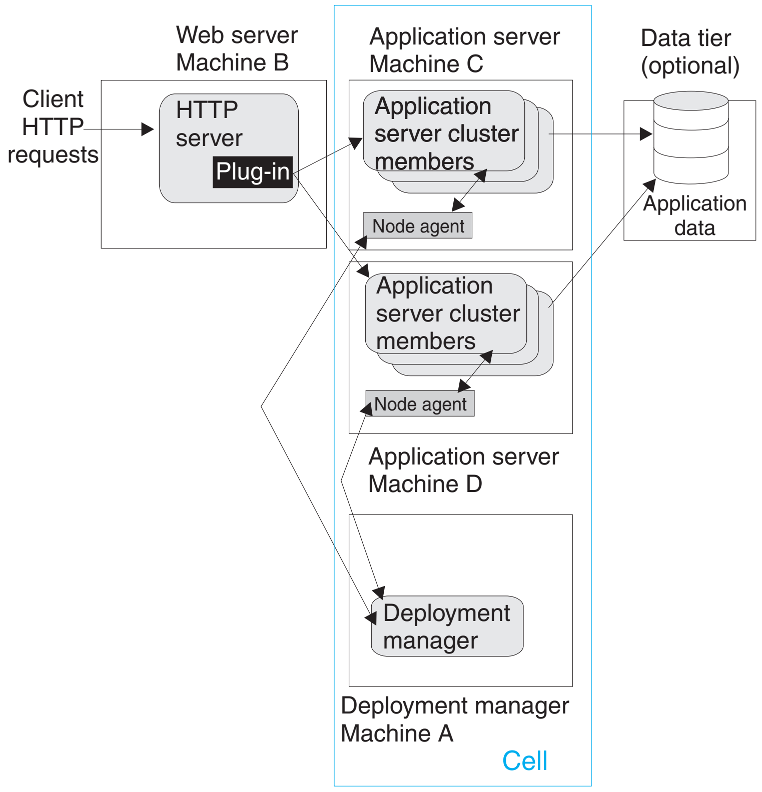
- **Minimizes the number of firewall holes.** Configurations that minimize the number of firewall ports are desirable because each additional firewall port leaves the firewall more vulnerable to attackers.

Some solutions are faster than others, in terms of the number of client requests they can process per unit of time. 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. To learn about the necessary maintenance for a topology, review the instructions for setting up and maintaining that topology. Of course, if you can automate the necessary administrative steps through command line clients and scripting, this might not concern you.

Horizontal scaling topology

Horizontal scaling exists when there are members of an application server cluster on multiple physical machines. Having cluster members on several machines lets a single application span the machines, yet still present a single system image.

The following figure shows an example of horizontal scaling.



In this example, the Web server on Machine B distributes requests to clustered application servers on Machines C and D. Cluster members on Machines C and D are created in the same cluster.

You can combine a network dispatcher to distribute client HTTP requests with clustering, to reap the benefits of both types of horizontal scaling. The *Network dispatcher topology* topic describes this system configuration.

Typical use

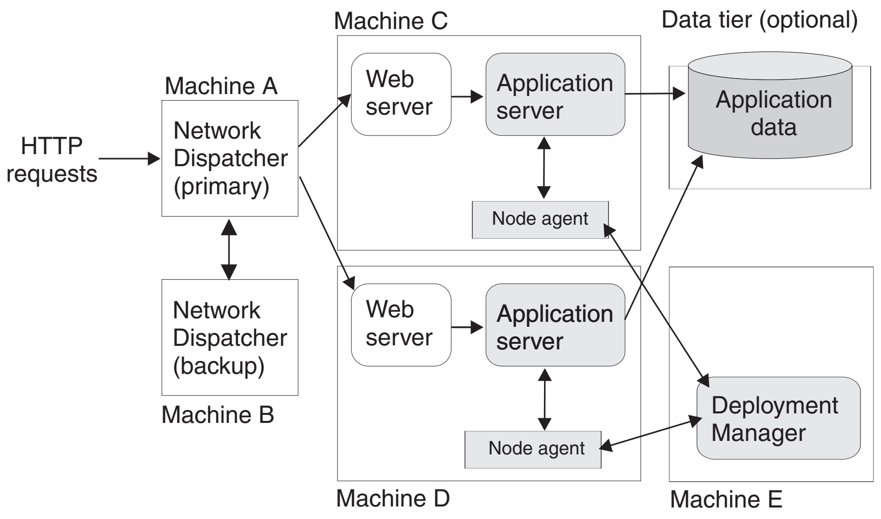
Horizontal scaling provides the increased throughput of vertical scaling topologies but also provides failover support. This topology lets you handle application server process failure and hardware failure without significant interruption to client service. You can also use horizontal scaling to optimize the distribution of client requests through mechanisms, such as workload management or remote HTTP transport.

Dispatcher

The Dispatcher component is part of the Load Balancer component set of the IBM WebSphere Application Server Edge Components product. The Edge components are included in WebSphere Application Server Network Deployment. The Dispatcher performs intelligent load balancing by using server availability, capability, workload, and other criteria you can define, to determine where to send a TCP/IP request. You can use the Dispatcher to distribute HTTP requests among application server instances that are running on multiple physical machines.

A simple Dispatcher topology

The following figure illustrates a simple horizontal scaling configuration that uses the Dispatcher to distribute requests among application servers on different machines.

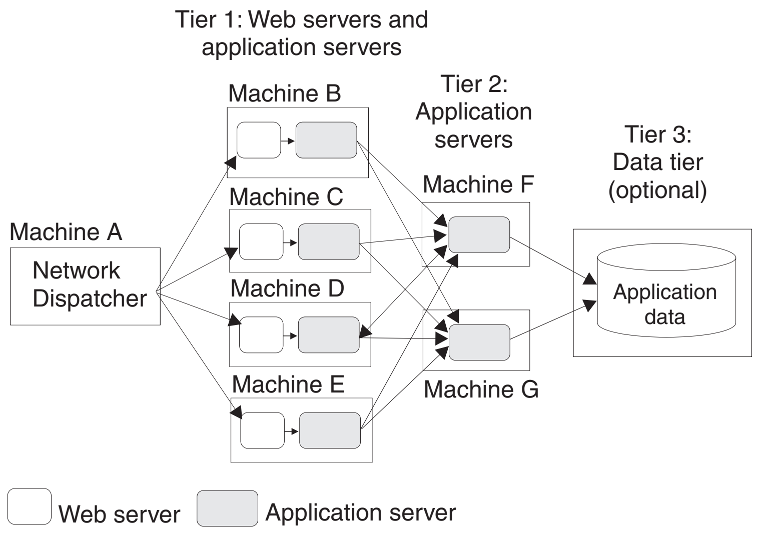


You normally configure a backup node for the Dispatcher machine, to eliminate it as a single point of failure. In this example, you can set up the backup Dispatcher node (Machine B) to take over if the primary Dispatcher node (Machine A) fails.

You can cluster application servers in this example from the same model, or configure them independently.

A more complex Dispatcher topology

The next figure shows a Dispatcher that distributes requests among several machines containing clustered Web servers and application servers. Backups are not shown.



The first tier Web server machines host servlet-based applications. The second tier application servers contain mostly enterprise beans that access application data and execute business logic. This approach lets you employ numerous, less powerful machines on the first tier and fewer but more powerful machines on the second tier.

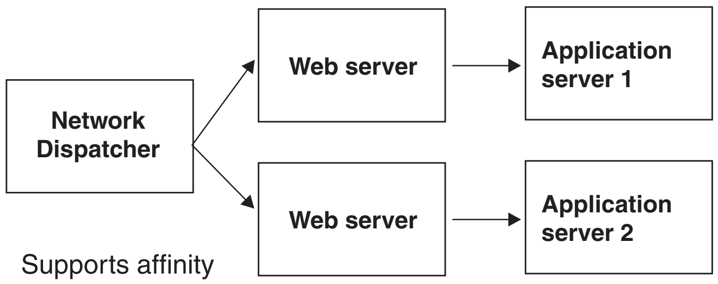
Using the Dispatcher with firewalls

You can also use a load balancing product, such as the Dispatcher with demilitarized zone (DMZ) topologies. The Dispatcher simplifies creating a DMZ topology with two firewalls. One firewall protects the Web server from the public Web site. The second protects backend systems from the Web server in the DMZ, by using proxy services.

The Dispatcher machine is placed between the outside firewall and the cluster of Web servers that it serves. The outside firewall provides filtering to allow only HTTP and HTTPS traffic. The firewall to the backend systems (DBMS, CICS, SAP, and so on) handles non-HTTP protocols, such as IIOP and JDBC. The WebSphere Application Server can reside in the DMZ or on the same side of the firewall as the data tier.

The Dispatcher and session affinity

In a Dispatcher or similarly configured topology, you must associate a Web server with a separate, rather than clustered, application server to preserve affinity between the servers.



Discussion

Adding a mechanism for the Dispatcher to distribute HTTP requests provides these advantages:

- Improves server performance, by distributing incoming TCP/IP requests - in this case, HTTP requests, among a cluster of servers.
- Increases the number of connected users.
- Eliminates the Web server as a single point of failure. You can also use this mechanism in combination with WebSphere Application Server workload management, to eliminate the application server as a single point of failure.
- Improves throughput, by letting multiple servers and CPUs handle the client workload.

Instructions

Set up machines containing Web servers and application servers for the topology you plan to implement.

Place the Dispatcher, or another load balancing product, in front of the Web server machines. See the Edge Component documentation for the Dispatcher, or the other load balancing product documentation. Instructions vary per product.

The load balancing product communicates with the Web server, which in turn communicates with application servers. The configuration involves setting up communications between the load balancing product and the Web server.

It does not matter to the Dispatcher whether the Web server is routing requests to an application server or processing them itself. Therefore, it is not necessary to perform any special configuration to make the load balancing product and application servers aware of one another. This lack of configuration is true with the Dispatcher, based on testing with IBM WebSphere Application Server. Results can vary with other load balancing products.

Web server separation

Web server separation is a topology that physically separates the Web (HTTP) server from the application servers, placing the Web server on a different machine in the configuration. Compared to a configuration where the Web server and the application servers are located on the same physical server, separating the Web server can improve application performance, provide better fault isolation, and enhance security. These topologies are often used with firewalls to create a secure demilitarized zone (DMZ) surrounding the Web server.

WebSphere Application Server provides these alternatives for physically separating the Web server from the application server:

- HTTP transport configurations
- Reverse proxy or IP forwarding configurations

These system topologies are described in detail in other topics.

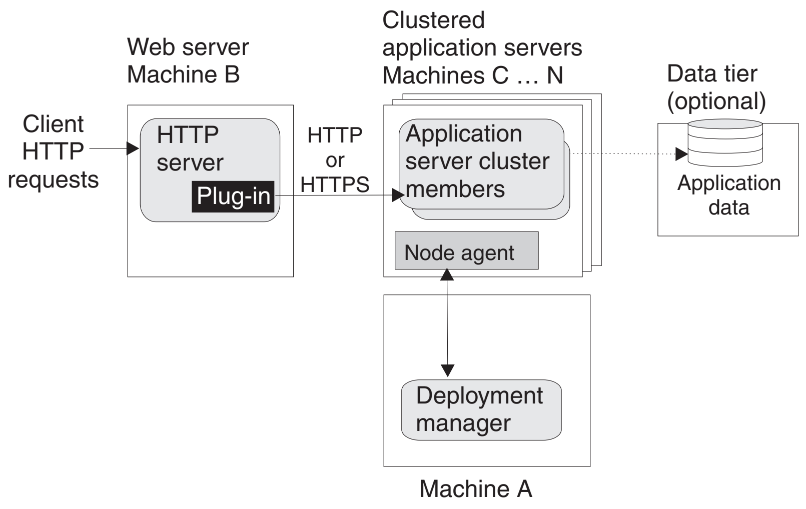
The following table summarizes advantages and disadvantages of each configuration. Criteria are explained after the table.

Topology	Secure Socket Layer	Database password required?	Workload manage-ment	Network Address Translation	Perfor-mance	Admin-istration
HTTP server separation	Yes	No	Yes	Yes	High	Manual
Reverse proxy	Yes	No	No	Yes	High	Manual

- **Secure Sockets Layer** Supports Secure Sockets Layer (SSL) security.
- **Database password required?** Requires a stored database user ID and password on the machine used by the database processes.
- **Workload management** Uses the workload management service to balance client workloads.
- **Network Address Translation** Supports Network Address Translation (NAT) firewalls. NAT firewalls receive packets for one IP address, translate the headers of the packets, and send the packets to a second IP address.
- **Performance** Compares the relative performance of each of these configurations.
- **Administration** Specifies whether to administer the configuration manually or through the administrative console.

These criteria give you a basis to compare the relative difficulty of administering each configuration.

HTTP transport: WebSphere Application Server can use the *HTTP protocol* to route requests from a Web server to application servers on remote machines.



In the diagram, Machine B hosts the Web server and receives HTTP requests from clients. The Web server forwards the requests to the application server on Machine C, by using the HTTP or HTTPS protocol.

Variations on this configuration include vertical scaling of the application servers. You can add application server machines (D, E, ... N), to the configuration to implement horizontal scaling.

The HTTP transport supports Network Address Translation (NAT) firewalls.

Load balancing support

The HTTP transport is fully integrated with WebSphere Application Server workload management and the clustering facility. HTTP transport balances loads within a cluster.

- **Load balancing between application servers.** You can configure the HTTP transport to forward requests from each URL to a different application server and its cluster members, enabling manual load balancing. For instance, you can forward URLs that generate a large number of requests to application servers on more powerful machines.
- **Load balancing among cluster members.** The HTTP transport automatically distributes requests among members of a cluster that is defined to respond to a single URL. The method for selecting which cluster member handles a particular request combines a round-robin selection policy with server affinity.

  If session persistence is not enabled, which is the default, requests are distributed among all available cluster members using a strict round-robin policy. Each cluster member gets the next request in turn. The only exception is when a cluster member is added or restarted. See the failover support information later in this topic for details.

  If session persistence is enabled, that is, session clustering and server affinity are enabled, requests are distributed as follows:

  – The HTTP transport distributes the first request of each session and all requests that are not associated with a session, as if session persistence is not enabled. They are distributed using a round-robin policy, except when cluster members are added or restarted.

  – The HTTP transport attempts to distribute all requests associated with a particular session to the same cluster member. Different sessions are assigned to different cluster members of the application server.

    Be aware that there is no guarantee that the same cluster member is used for all requests within a session. You cannot always maintain session affinity in situations where the number of available cluster members changes during the lifetime of a session. The Session Manager session clustering facility ensures that session state is not lost if requests are switched to another cluster member during a session. In any case, applications that require available session information across multiple client invocations must store session information in a database.

Failover support

The HTTP transport automatically handles failover and changes in the number of available cluster members.

- If a cluster member is stopped or unexpectedly fails, all subsequent requests are distributed among the remaining cluster members. The unavailable cluster member is skipped.

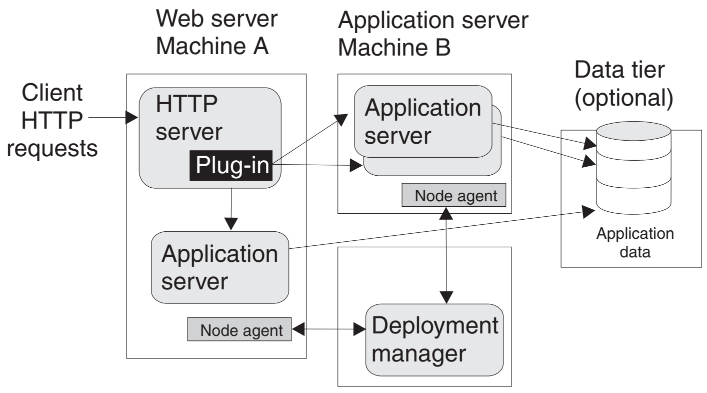
- If a cluster member is added or restarted, the system automatically begins to distribute requests to it. The next several requests are dispatched to that cluster member before HTTP resumes its normal methods for distributing requests to the cluster members of an application server, based on whether session persistence is enabled. See the load balancing support information for details.

Typical use

HTTP transport has the following advantages:
- Supports load balancing and failover.
- Requires no requirements for database access through the firewall.
- Supports WebSphere Application Server security.
- Supports Secure Sockets Layer (SSL) encryption for communications between the Web server and the application server.
- Supports Network Address Translation (NAT) firewalls.
- Supports relatively fast performance.

The HTTP transport has the disadvantage of requiring at least one firewall port, more if multiple application server cluster members are configured, or WebSphere Application Server security is used on the machine hosting the Web server.

A variation of the HTTP transport topology occurs when an instance of the application server runs on the machine that hosts the Web server. In this configuration, an instance of an application server runs on the same machine as the Web server.



Such configurations can direct client requests to additional application server cluster members on other machines. This example redirects client requests to both the application server instance running on Machine A and the cluster members running on Machine B. You can administer all application server instances from the deployment manager node, which can exist on Machine A, Machine B, or another machine. The deployment manager communicates with node agents on each machine, to coordinate configuration changes.

Typical use

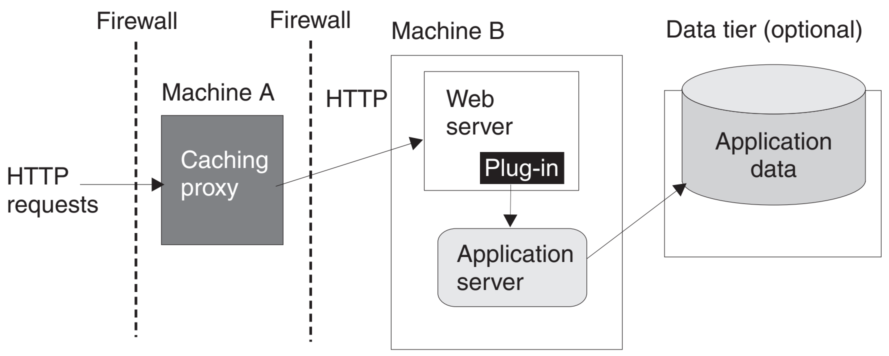
This topology is recommended only in situations where hardware limitations prevent you from hosting the Web server on a dedicated machine.

In many production environments, one set of servers is configured to run Web servers and another set of servers is configured to run application servers. This configuration lets you add capacity in a production environment. You might also use this topology to more fully replicate a production configuration in a test environment. This topology provides a means of load distribution between a machine hosting both the Web server and application server, and machines hosting just the application server.

You can also use this topology to distribute the workload in situations where there are a limited number of machines.

Reverse proxy (IP forwarding): *Reverse proxy*, or *IP-forwarding* topologies use a reverse proxy server, such as the Caching Proxy in WebSphere Application Server Edge Components, to receive incoming HTTP requests and forward them to a Web server. The Web server forwards the requests to the application servers for actual processing. The reverse proxy returns completed requests to the client, hiding the originating Web server.

The following figure shows a simple reverse proxy 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 such requests to an HTTP server that resides on the same machine as WebSphere Application Server. After the requests are fulfilled, they are returned through the reverse proxy to the client, hiding the originating Web server.

Typical use

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 Application Server 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 Secure Socket Layer (SSL) - port 443.

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

- The reverse proxy server does not need database access through the firewall.

- The reverse proxy configuration supports WebSphere Application Server security and NAT firewalls.
- The basic reverse proxy configuration is well known and tested in the industry, resulting in less customer confusion than other DMZ configurations.
- The reverse proxy configuration is reliable and its performance is relatively fast.
- The reverse proxy configuration eliminates protocol switching, by using the HTTP protocol for all forwarded requests.
- The reverse proxy configuration does not affect the configuration and maintenance of an application deployed on WebSphere Application Server.
- The reverse proxy server uses only one HTTP firewall port for requests and responses.

The reverse proxy configuration is also a disadvantage in some environments where security policies prohibit using the same port or protocol for inbound and outbound traffic across a firewall.

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

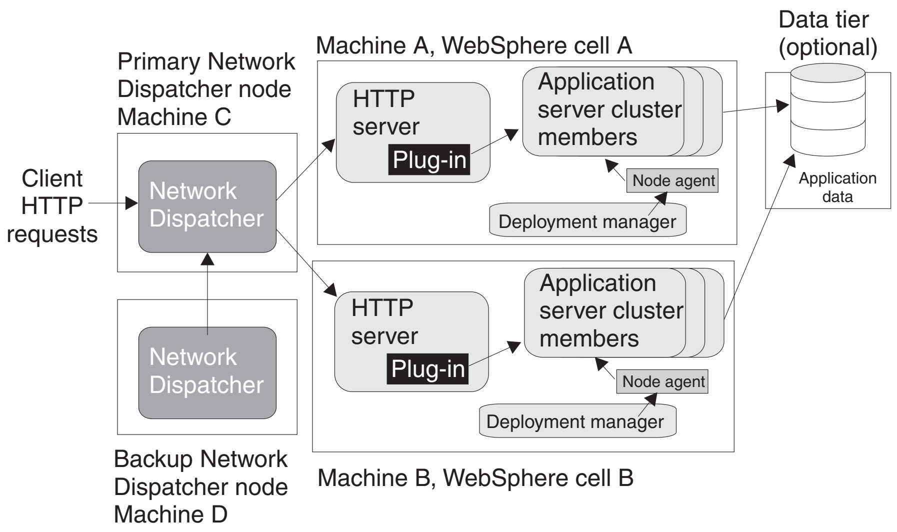
- The presence of a reverse proxy server in a DMZ is not suitable for some environments.
- The reverse proxy configuration requires more hardware and software than similar topologies that do not include a reverse proxy server, which makes it more complicated to configure and maintain.
- The reverse proxy server does not participate in WebSphere Application Server workload management.

Instructions

Implementation specifics are determined by the reverse proxy server. Refer to the documentation for the product you are using. No additional WebSphere Application Server administration is required for the reverse proxy server, although you might need it for other elements of the reverse proxy topology.

Multiple deployment manager cells

The following figure shows an example of how you can implement an application over multiple WebSphere Application Server deployment manager cells.


Data tier (optional)
Machine A, WebSphere cell A
Primary Network Dispatcher node Machine C
HTTP server
Plug-in
Application server cluster members
Node agent
Application data
Client HTTP requests
Network Dispatcher
Deployment manager
HTTP server
Plug-in
Application server cluster members
Node agent
Network Dispatcher
Deployment manager
Backup Network Dispatcher node Machine D
Machine B, WebSphere cell B

The example application runs simultaneously in two cells, each hosted on a different physical machine (Machines A and B). Network Dispatcher is used to distribute incoming HTTP requests between the two cells, presenting a single image of the application to clients. A backup Network Dispatcher node provides failover support.

Installing enterprise applications on the cluster in each deployment manager node ensures that identical versions of the application run in each cluster member. However, you administer each cell independently. Each cell has its own set of XML configuration files.

You can also run a different version of the application in each cell cluster. Because the cells are isolated from one another, you can also run different versions of the WebSphere Application Server software in each cell. For example, you can have a Version 5 cell and a Version 4 domain interoperating in your network.

Typical use

Topologies that incorporate more than one cell have 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 isolation is helpful in a variety of situations:
  - When rolling out a new application or a revision of an existing application.

    You can bring the new application or revision online in one cell, and test it in a live situation while other cells continue to handle client requests with the production version of the application.
  - When deploying a new version of the WebSphere Application Server software.

    You can bring the new version into production, and test it in a live situation without interrupting service.
  - When applying fixes or patches to the WebSphere Application Server software.

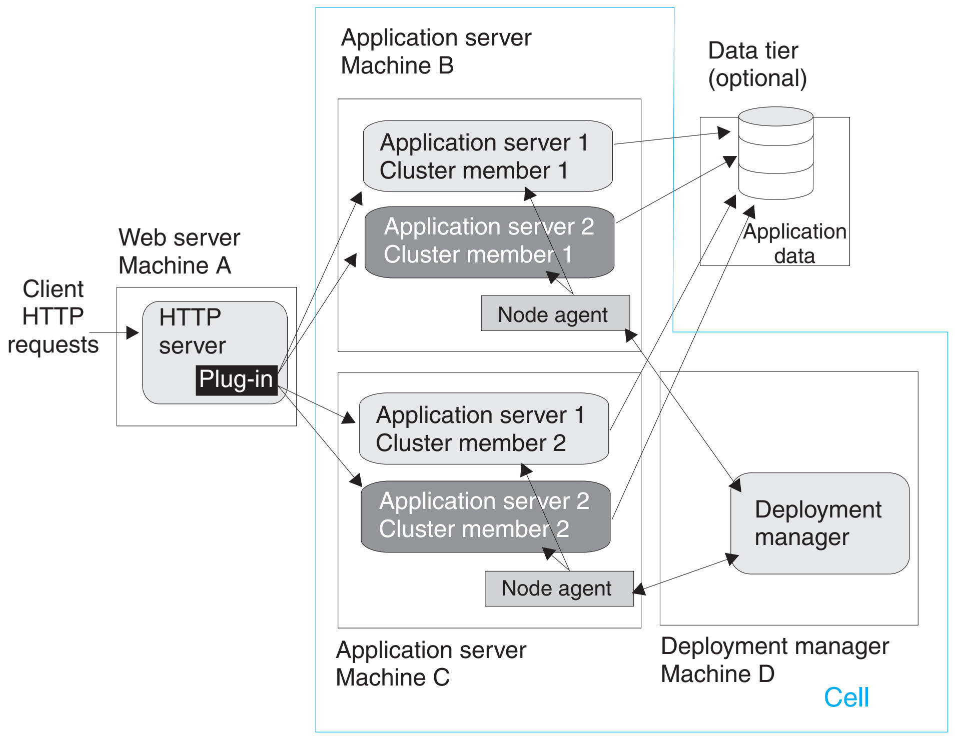
    You can take each cell offline to upgrade it, without interrupting the application.

  If an unforeseen problem occurs with the new software, using multiple cells can prevent an outage to an entire site. You can also rollback to a previous software version more quickly. You can handle hardware and software upgrades on a cell-by-cell basis during off-peak hours.

Using multiple cells has several drawbacks:
- Deployment is more complicated than for a single administrative cell.
- Multiple cells require more administration effort because each cell is administered independently. Use scripts to standardize and automate common administrative tasks to reduce this problem.

Multiple application servers within a node

The following figure shows a topology in which cluster members of more than one application server are hosted on a physical node.



Application server Machine B
Data tier (optional)
Application server 1 Cluster member 1
Application server 2 Cluster member 1
Node agent
Application data
Web server Machine A
Client HTTP requests
HTTP server
Plug-in
Application server 1 Cluster member 2
Application server 2 Cluster member 2
Node agent
Deployment manager
Application server Machine C
Deployment manager Machine D
Cell

The example topology is a variation of the basic horizontal scaling topology. Cluster members are not hosted on just a single machine, but are distributed throughout all of the machines in the system. In this example, a cluster member is hosted on Machines B and C. Machine A serves as the Web server for the application and distributes client requests to the application server cluster members on each node.

Typical use

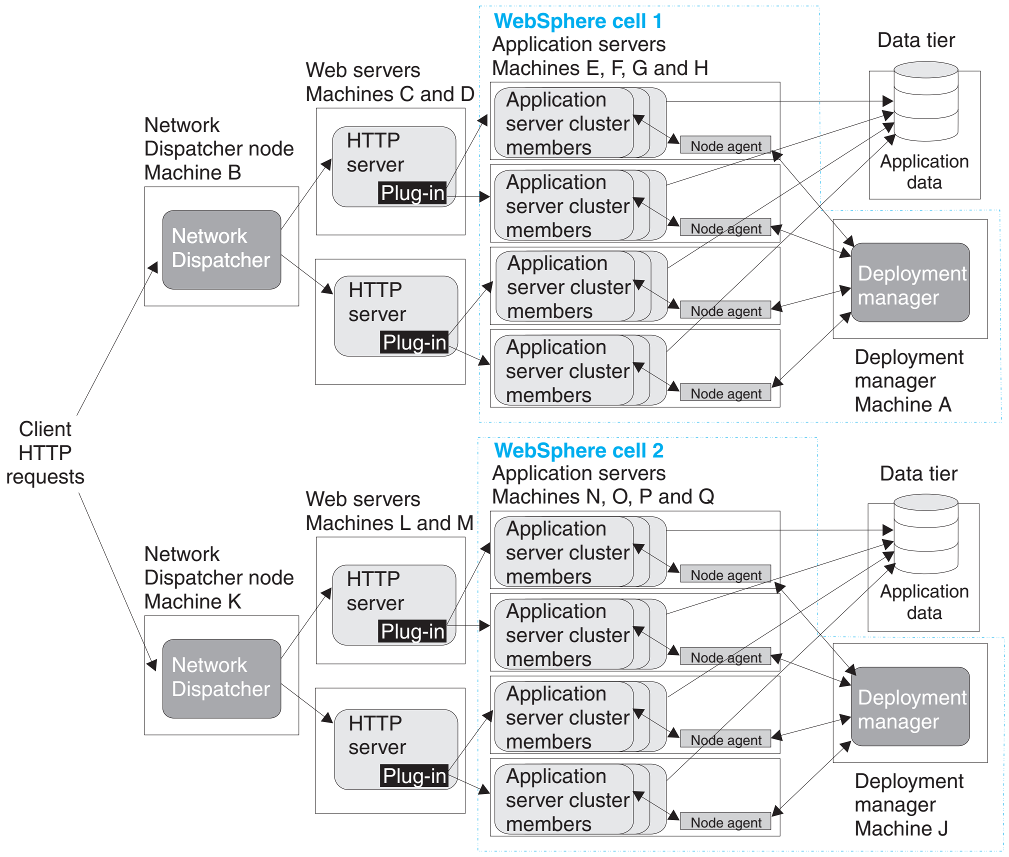
Hosting cluster members of multiple application servers within a node provides the following benefits:

- Improved throughput. Clustering an application server enables it to handle more client requests simultaneously.
- Improved performance. Hosting cluster members on multiple machines enables each cluster member to use machine processing resources.
- Hardware failover. Hosting cluster members on multiple nodes isolates hardware failures and provides failover support. Redirect client requests to the application server cluster members on other nodes if one node goes offline.
- Application server failover. Hosting cluster members on multiple nodes also isolates application software failures and provides failover support if a cluster member stops running. Redirect client requests to cluster members on other nodes.
- Process isolation. If one application server process fails, its cluster members on the other nodes are unaffected.

A drawback of this topology is more complex maintenance. You must maintain cluster members of each application server on multiple machines.

Putting it all together - a combined topology

An example of a topology that combines the best elements of the other topologies discussed in this section is shown in the following figure.



This topology combines elements of several different basic topologies:

- Two deployment manager cells
- A deployment manager node to manage each cell (Machine A in cell 1; Machine J in cell 2)
- Two Network Dispatcher nodes (Machine B in cell 1; Machine K in cell 2)
- Two HTTP servers for each cell (Machines C and D in cell 1; Machines L and M in cell 2)
- Four application server nodes for each cell (Machines E, F, G, and H in cell 1; Machines N, O, P, and Q in cell 2)
- The use of cluster members for both vertical and horizontal scaling. In the example topology, each node hosts three cluster members; in practice, the number of cluster members is limited by the computing resources of each node.
- A data tier for each cell.

Typical use

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

- Having more than one Network Dispatcher node, HTTP server, and application 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. You can handle hardware and software upgrades on a cell-by-cell basis during off-peak hours.
- Horizontal scaling uses both clustering and a Network Dispatcher 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 one large cell. There is less interprocess communication in a smaller cell, which lets you devote more resources to processing client requests.
  - Using cluster members to vertically scale application servers on each node, which makes more efficient use of machine resources.
- Applications with this topology can use several workload management techniques. In this example, you can perform workload management through one or both of the following:
  - Using the WebSphere Application Server Network Deployment workload management (WLM) facility to distribute work among application server cluster members.
  - Using Network Dispatcher to distribute client HTTP requests to each Web server.

  For example, an application can manage workloads at the Web server level with Network Dispatcher and at the application server level with WebSphere Application Server 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 situation occurs, the active HTTP sessions are lost for half of the clients. The system can still process HTTP requests, although its performance is degraded.

The combined topology has several drawbacks:
- Deployment is more complicated. WebSphere Application Server software and application files must deploy in each cell, which is not the case for applications that run only in a single cell.
- Multiple cells require more administration effort because each cell is administered independently. Reduce this problem by using scripting to standardize and automate common administrative tasks.

Running WebSphere Application Server across versions

WebSphere Application Server, Version 5 is generally interoperable with WebSphere Application Server, Versions 3.5.x and 4.0.x. However, there are specific requirements to address for each version. Make the following changes to support interoperability between versions.

1. Apply required e-fixes.

E-fixes to apply for Version 3.5.x

E-fix	Version 3.5.3	Version 3.5.4	Version 3.5.5	Version 3.5.6
PQ51387	Apply	Apply		
PQ60074	Apply	Apply	Apply	Apply
PQ60335			Apply	
PQ63548	Apply	Apply	Apply	Apply

E-fixes to apply for Version 4.0.x

E-fix	Version 4.0.1	Version 4.0.2	Version 4.0.3
PQ60074	Apply	Apply	
PQ60336	Apply	Apply	
PQ63548	Apply	Apply	Apply

All e-fixes are available on the IBM WebSphere Application Server Support page.

E-fix PQ51387:
A naming client e-fix that supports Version 3.5.x naming client access to the Version 5 name server.

E-fix PQ60074:
An Object Request Broker (ORB) e-fix that supports Version 5 naming client access to the Version 3.5.x or 4.0.x name server. A down-level client has no problem accessing a Version 5 name server, even using corbaloc.

E-fix PQ60335:
An ORB e-fix to reconcile `java.math.BigDecimal` and other class differences in IBM Software Development Kits 122 and 131.

Note: This e-fix does not apply to IBM Software Development Kits on the Solaris operating system.

E-fix PQ60336:
An ORB e-fix to reconcile `java.math.BigDecimal` and other class differences in IBM Software Development Kits 130 and 131.

Note: This e-fix does not apply to IBM Software Development Kits on the Solaris operating system.

E-fix PQ63548:
An e-fix to correct problems that might occur when passing embedded valueTypes between WebSphere Application Server releases.

The best solution is to upgrade all your installations to the latest release and PTF levels, such as Versions 3.5.7 or 4.0.4, which do not require this fix. If this solution is not possible, apply the e-fix to your version.

Symptoms include `org.omg.CORBA.MARSHAL` exceptions when passing embedded valueTypes across the versions. Sometimes, other symptoms might mask `org.omg.CORBA.MARSHAL` exceptions, which makes them difficult to identify.

If symptoms reoccur in spite of the e-fix, re-export existing IORs.

2. Follow required guidelines.

Guidelines to apply for Version 3.5.x and Version 4.0.x

Guideline	Version 3.5.x	Version 4.0.x
1	Apply	
2	Apply	Apply
3	Apply	
4	Apply	
5	Apply	Apply
6	Apply	Apply

Guideline 1:
Use the context of the lowest common denominator when interoperating at the naming level. For example, always use the 3.5.x context com.ibm.ejs.ns.jndi.CNInitialContextFactory when a client or server is at 3.5.x. For later versions, use the current com.ibm.websphere.naming.WsnInitialContextFactory context.

Guideline 2:
Make required naming changes to support Version 3.5.x or 4.0.x client access to Version 5 enterprise beans. This issue is new, introduced by Version 5. There are several ways to make it work, such as:

- Updating the `namebindings.xml` file if you do not use the Version 5 migration tools, but have installed Version 3.5.x or 4.0.x applications on Version 5. To allow Version 3.5.x or 4.0.x client access to the applications, add additional information to the bind information in the Version 5 namespace to make the old JNDI names work. Add the information to the `namebindings.xml` configuration file at the cell level using the administrative console.

  Note: Applications that you migrate to Version 5 during installation, or that you manually migrate using the WASPreUpgrade and WASPostUpgrade migration tools, already have this update.

  After federating an application server node into a deployment manager cell, you cannot use the migration tools. To use these tools again, remove the node from the cell, use the tools, and add the node to the cell again.
- Calling Version 5 enterprise beans directly using the naming path that includes the server on which the enterprise beans are running, such as `cell/node/server1/some/ejb/name`, for example.
- Using the Version 4.0.x client `java:/comp` location to find Version 5 enterprise beans. (You cannot use the command from a Version 3.5.x client.)

Guideline 3:
Ensure that programs performing a JNDI lookup of the `UserTransaction` interface, use an InitialContext that resolves to a local implementation of the interface. Also ensure that such programs use a JNDI location appropriate for the enterprise bean version.

Prior to the EJB 1.1 Specification, the JNDI location of the UserTransaction interface was not specified. Earlier versions up to and including Version 3.5.x do not use the EJB 1.1 Specification. They bind the UserTransaction interface to a JNDI location of `jta/usertransaction`.

Version 4, and later releases, bind the UserTransaction interface at the location defined by the EJB 1.1 Specification, which is `java:comp/UserTransaction`.

Version 5 no longer provides the earlier `jta/usertransaction` binding within Web and EJB containers to applications at a J2EE level of 1.3 or later, to enforce use of the newer UserTransaction interface. For example, EJB 2.0 applications can use only the `java:comp/UserTransaction` location.

Guideline 4:
Be aware of limitations when calling WorkLoad Management (WLM)-enabled enterprise beans.

Some clients cannot call WLM-enabled enterprise beans on remote clusters when there is a local WLM-enabled enterprise bean of the same name. If there is a cluster local to the client with the same enterprise bean as the remote cluster that the client is trying to call, the client ends up talking to the local cluster. The following table lists supported combinations of clients calling WLM-enabled enterprise beans on remote application servers.

All clients at Version:	Server at Version:	Supported interoperability
3.5.6	5	Yes
4.02, 4.03	5	Yes
5	3.5.x	No
5	4.02, 4.03	Yes

Guideline 5:
Be aware of administrative console limitations.

You cannot use the administrative interfaces for Version 5 to administer a Version 3.5.x or 4.0.x administrative server. Likewise, you cannot use a Version 3.5.x or Version 4.0.x administrative console to administer a Version 5 environment. If you use the administrative console on a remote machine to administer Version 3.5.x or 4.0.x WebSphere Application Server domains, migrating any of the nodes or domains to Version 5 renders the remote administration console ineffective for administering any Version 5 environment.

Guideline 6:
Use Secure Socket Layer Version 3 (SSLv3) when interoperating with Version 3.5.x for secure connections. You cannot use Common Secure Interoperability Version 2 (CSIv2) for interoperability, because Versions 3.5.x and 4.0.x do not support CSIv2.

This information is dynamic and might be augmented by information in Technical Articles that are available on the **IBM WebSphere Developer Domain** at http://www7b.software.ibm.com/wsdd/. Be sure to check the site for the latest information.

Installing the product

This topic describes installing the Network Deployment product, using the installation image on the product CD-ROM labelled, **Deployment Manager**. There is a CD-ROM in the Network Deployment product package that you can use to install the base WebSphere Application Server product. It is labelled, **Application Server, IBM HTTP Server**. Open this topic in the InfoCenter for the base WebSphere Application Server product to learn how to install the base WebSphere Application Server product.

The **Deployment Manager** CD-ROM is available in both the Network Deployment product package and the Enterprise product package. Use this InfoCenter topic to install the Network Deployment product, regardless of which package the CD-ROM comes from. Open this topic in the InfoCenter for the Enterprise product to learn how to install the Enterprise product and extend the multimachine environment. The CD-ROM for the Enterprise product is labelled, **Enterprise**.

This topic is available in Adobe PDF format, on the product CD-ROMs, as well as online in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html. When possible, access the most current version of this information by selecting the InfoCenter version in your language.

Although it is not supported or recommended, you can install this product as a non-root user on a UNIX-based operating system, or from a user ID that is not part of the Administrator group on a Windows NT or Windows 2000 platform. However, there are certain components, such as the embedded messaging feature that require you to install as root or as part of the Administrator group.

The LaunchPad tool lets you access the product overview, `readme.html` file, and installation guides. After starting the LaunchPad, click its **Install the product** option to walk through the installation wizard.

The installation wizard:
- Automatically checks prerequisites
- Looks for a previous WebSphere Application Server installation, to determine whether to display the migration and coexistence panel during the installation

This section contains the following topics:
- Using the LaunchPad to start the installation
- Installing with the installation wizard GUI
- Installing silently
  - Tailoring the options response file
  - responsefile
- Platform-specific tips for installing and migrating

Using the LaunchPad to start the installation

Use the LaunchPad to access a product overview, the ReadMe file, and installation guides, and to install the product.

1. Start the LaunchPad.

   On Windows NT and Windows 2000 systems, insert the product CD to automatically run the LaunchPad.

You can also start the LaunchPad manually:

- On Windows NT and Windows 2000 platforms, run the `LaunchPad.bat` command.
- Mount the CD-ROM drive on a UNIX-based system, if necessary. This procedure varies per platform. Consult your operating system documentation for instructions on mounting and dismounting CD-ROM drives. After accessing data on the CD, run the `LaunchPad.sh` shell script.

The LaunchPad program is in the *`operating-system platform`* directory on the product CD.

2. Select a language for the LaunchPad.
3. Use the LaunchPad to access the product overview, the ReadMe file, and installation guides.
4. Click **Install the product** to launch the installation wizard.

Installing with the installation wizard GUI

1. Start the installation.

   The default installation method is to click **Install the product** on the LaunchPad tool to launch the InstallShield for MultiPlatforms installation wizard. This action launches the installation wizard GUI.

   You also have the option of starting the installation wizard from the product CD-ROM. The installation program is in the *`operating-system platform`* directory on the product CD-ROM.

   - On Windows NT and Windows 2000 platforms, run the **Install.exe** command.
   - On Linux/390 platforms, run the **install.sh** command.
   - On other Linux platforms and UNIX-based platforms, run the **install** command.

   You can also perform a silent installation using the `-options responsefile` parameter, which causes the installation wizard to read your responses from the options response file, instead of from the interactive GUI. You must tailor the response file before installation. After tailoring the file, you must issue the command to silently install. Silent installation is particularly useful if you install the product often.

   The rest of this procedure assumes that you are using the installation wizard. There are corresponding entries in the response file for every prompt that is described as part of the wizard. Review the description of the response file for more information. Comments in the file describe how to tailor its options.
2. Select a language for the wizard GUI and click **Next**.

   The installation wizard opens and a welcome page appears.
3. Click **Next** to continue.

   The license agreement appears for you to read.
4. Click the radio button beside the **I accept the terms in the license agreement** message if you agree to the license agreement and click **Next** to continue.

   As the WebSphere Application Server Network Deployment product version changes, its prerequisites and corequisites change.

   The Network Deployment product simplifies migrating product prerequisites, by providing the option to install a complimentary Java 2 SDK on your supported operating system. You can uninstall back-level prerequisites and let the installation wizard install current versions.

Although the installation wizard checks for prerequisite operating system patches, review the prerequisites on the IBM WebSphere Application Server supported hardware, software, and APIs Web site at http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html. You can also open the page by selecting the prerequisites option from the LaunchPad. Refer to the documentation for non-IBM prerequisite and corequisite products to learn how to migrate to the supported version.

Earlier versions of the WebSphere Application Server product required you to edit prerequisites checking, to correct situations where operating system patch levels, the Web server software level, or the database level were more recent that what was defined in the `prereq.properties` file. This was necessary before you could continue the installation of the earlier product. Passing the prerequisites test is no longer required to install the WebSphere Application Server product because Version 5 does *not* store configuration data in a database. It stores the information in XML configuration files on the node. The WebSphere Application Server product does not require a database during installation.

The `*platform_specific_directory*waspc/prereqChecker.xml` file on the product CD-ROM checks for the presence of a supported operating system and required patches. For migration, coexistence, and other reasons, the `prereqChecker.XML` file also checks for existing versions of WebSphere Application Server, as well as for existing Version 5 and Version 5 feature installations.

If the wizard finds a previous version of WebSphere Application Server, it prompts you to migrate applications and the configuration from the previous version, or to coexist with it. If it finds more than one previous version, the installation wizard lists them for you to select which one to migrate.

5. Choose whether to migrate applications and the configuration from a previous version, or to coexist with another version, to do both, or to do neither, and click **Next** to continue.

   Note: You can also perform a silent migration or configure for coexistence during a silent installation.

   Refer to the ″Installing silently″ topic for a description of performing a silent installation, including options you can specify.

   The migration prompt appears only when the installation wizard detects a previous version. The coexistence prompt appears when the installation wizard detects any other installation, including another Version 5 installation.

   If you choose to coexist, the wizard displays a port selection panel, where you can select non-conflicting ports. For example, you can change the HTTP transport port for coexistence, from 9081 (one more than the default Version 5 port number) to 9085 or higher, to avoid potential conflicts with port numbers that previous versions of WebSphere Application Server commonly use.

   In some cases, such as when installing a non-English version, the installation wizard might not detect a previous version. You can force the migration and coexistence panel to appear, by starting the installation with an option on the **Install.exe** or **install** command. For example, on UNIX machines, use this command:

   ```
   ./install -W previousVersionDetectedBean.previousVersionDetected="true"
   ```

   You can also force the appearance of the coexistence panel to change conflicting port number assignments. For example, the AIX WebSM system management server listens on port 9090 by default. To avoid a conflict with the WebSphere HTTPS Administrative Console Secure Port (HTTP_TRANSPORT_ADMIN) assignment, which is also 9090 by default, force the coexistence panel to appear using this command:

```
./install -W coexistenceOptionsBean.showCoexistence="true"
```

If you choose neither the migration option nor the coexistence option, you can run Version 5 and the earlier version, but not at the same time. Although it is possible that both might coexist without port conflicts, you can ensure that both versions run together by selecting the coexistence option and checking for conflicting port assignments.

The migration panel lists all known previous releases that it can identify. If you highlight each release that is shown, the **select previous version** text boxes show the location of the previous product. Select the product that you intend to migrate. If you do not see the previous version that you intend to migrate, click **Select previous version** to enter a location and configuration file name if you are migrating a WebSphere Application Server Advanced Single Server Edition, Version 4.x installation. (The configuration file name is valid only for WebSphere Application Server Advanced Single Server Edition, Version 4.x.) Otherwise, you must start the Administrative Server of the previous version so that the installation wizard can export the configuration.

Although you might select migration at this point in the installation process, the actual migration does not begin until after the Version 5 installation is complete. At that time, if the WASPreUpgrade tool fails, the installation wizard does not call the WASPostUpgrade tool to complete the migration but instead, displays the `WASPreUpgrade.log` and `WASPostUpgrade.log` log files for you to diagnose the problem. After fixing the problem, such as starting the administrative server of a previous release, you can start the migration again as described in the ″Migrating and coexisting″ topic.

6. Select features to install and click **Next** to continue.

   A description of each feature appears at the bottom of the panel when you roll the cursor over the feature. Choose from these features:

   Deployment manager

   Installs the product run time. It provides high performance and scalability across your deployment environment. It includes multiserver administration, server clustering, load balancing and workload management for hosting highly available e-business applications.

   Web services

   Installs the UDDI registry and the Web services gateway to provide Web services support for your e-business applications.

   See the Developing and managing Web services topic for more information.

   UDDI Registry

   Installs a Version 2 compliant universal description, discovery, and identification (UDDI) registry, accessible from the UDDI registry User Console application, or from SOAP or EJB interfaces.

   See the IBM WebSphere UDDI Registry topic for more information.

Web services gateway

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.

See the Enabling Web services through the IBM Web Services Gateway topic for more information.

Embedded messaging client

Includes the client necessary for the administration of WebSphere MQ Queues and the mapping of JMS resources into the deployment manager JNDI namespace.

Embedded messaging feature considerations

Before you install the embedded messaging component on UNIX platforms, you must define the operating system groups **mqm** and **mqbrkrs**, and the user IDs needed for embedded WebSphere messaging. For more information, see Installing WebSphere embedded messaging as the JMS provider. On AIX Version 4.3.3 and AIX Version 5, you must install Java130.rte.lib Version 1.3.0 to ensure that the embedded messaging feature installs correctly.

You have a choice if you already have WebSphere MQ Version 5.3 installed:

- You can install only the embedded messaging client feature on a machine that already has WebSphere MQ Version 5.3.

  To use WebSphere MQ Version 5.3 as the JMS provider, install the **embedded messaging client** feature. Installing and using the WebSphere Application Server embedded messaging client feature is recommended for use with either the base WebSphere Application Server product embedded messaging server feature or the full WebSphere MQ Version 5.3 product server feature. WebSphere Application Server messaging applications can use the WebSphere MQ Version 5.3 product as the JMS provider. Using the client feature, however, requires that you already have installed the WebSphere MQ Version 5.3 feature, Java Messaging.
- You can install the embedded messaging client feature on a machine that already has WebSphere MQ Version 5.3.

  To install the embedded messaging client feature when WebSphere MQ Version 5.3 is already installed, upgrade WebSphere MQ Version 5.3:
  - Apply the CSD01 update to the original WebSphere MQ Version 5.3 release, or move to the WebSphere MQ Version 5.3 refresh release (which includes CSD01).
  - Install the WebSphere MQ Version 5.3 feature, **Java Messaging**, which the WebSphere Application Server embedded messaging client feature requires.

If you have not installed WebSphere MQ Version 5.3 with the required MQ feature, the installation of the IBM WebSphere Application Server embedded messaging client feature is unsuccessful because of prerequisite check errors.

For information about installing the WebSphere MQ Version 5.3 product, or migrating to WebSphere MQ Version 5.3 from an earlier release, refer to the appropriate WebSphere MQ *Quick Beginnings* book, as follows:

- *WebSphere MQ for Windows, V5.3 Quick Beginnings*, GC34-6073
- *WebSphere MQ for AIX, V5.3 Quick Beginnings*, GC34-6076
- *WebSphere MQ for Solaris, V5.3 Quick Beginnings*, GC34-6075
- *WebSphere MQ for Linux for Intel and Linux for zSeries, V5.3 Quick Beginnings*, GC34-6078

These books are available at the WebSphere MQ messaging platform-specific books Web page at http://www-3.ibm.com/software/ts/mqseries/library/ manualsa/manuals/platspecific.html.

Adding additional features at a later time or uninstalling features

You can add features to an existing installation, by running the installation wizard again. An installation wizard panel lets you select whether to add features to the existing installation, or perform a new installation to another directory.

When adding features, previously installed features are checked and grayed out with the term **(Installed)** at the end of the feature name.

You can run the uninstall program to remove all installed features.

7. Specify a destination directory. Click **Next** to continue.

   Ensure there is adequate space available in the target directory. On an AIX system, also ensure that there is adequate space in the `/tmp` directory. When installing on a Solaris system, do not use a double-byte character set (DBCS) name for the directory.

   If you select the embedded messaging client feature and there are missing prerequisites, the installation wizard displays the `mq_prereq.log` error log and takes you back to the installation type panel. Choose **Custom** installation and deselect the embedded messaging client feature to continue. The `mq_prereq.log` file is in the system temp directory.

8. Specify node information and click **Next**.

   Specify the node name, cell name, and host name or IP address for the deployment manager. Although the wizard inserts the machine name (of the installation platform) as the node name, you can specify any name. The node name is an arbitrary WebSphere Application Server-specific name that must be unique within a cell.

   The host name is the network name for the physical machine on which the node is installed. It must resolve to a physical network node on the server. When there are multiple network cards in the server, for example, the host name or IP address must resolve to one of the network cards. Remote WebSphere Application Server nodes use the host name to connect to, and communicate with, this node. It is extremely important to select a host name that other machines can reach within your network.

   The value you use for the host name can be the fully qualified DNS hostname, the short host name, or even a numeric IP address. A numeric IP address has the advantage of not requiring name resolution through DNS. A remote node can connect to the node you name with a numeric IP address without DNS being available. The disadvantage is that the numeric IP address is fixed. You must change this setting in the configuration whenever you change the machine IP address. Therefore, do not use a numeric IP address if you use DHCP, or if you change IP addresses regularly. You also cannot use the node if the numeric IP host name is disconnected from the network.

A fully qualified domain hostname is usually the best choice. Advantages are that such host names are flexible enough to support DHCP systems and can run disconnected from the network. Remote nodes usually resolve the address. A disadvantage is that a remote node depends on DNS availability to connect to either a fully qualified or a short domain host name.

9. **(Optional)** Create a Windows NT or Windows 2000 service, or plan to create a UNIX monitored process. Click **Next**.

   Processes started by a **startServer** command are not running as monitored processes, regardless of how you have configured them.

   Run WebSphere Application Server Network Deployment as a service on Windows NT or Windows 2000 systems only by clicking the check box. Clicking the check box on this panel configures a manually started service. You can also create UNIX monitored processes after the installation is complete. Processes started by a **startServer** (or **startNode** or **startManager**) command are not running as monitored processes, regardless of how you have configured them.

   For example, you can configure the deployment manager server as a WebSphere Application Server Windows service. However, if you start the deployment manager instance using the **startManager** command, Windows does not monitor or restart the deployment manager because it was not started as a Windows service. The same is true on UNIX-based platforms. You must start the server process with a shell script based on the example `rc.was` file, to have the server running as a monitored process.

   To perform this installation task, your local Windows user ID must belong to the Administrator group and have the advanced user rights *Act as part of the operating system* and *Log on as a service*. (If your Windows user ID belongs to the Administrator group, the installation wizard grants it the advanced user rights.) You can also create other Windows NT or Windows 2000 services after the installation is complete, to start other server processes.

   Although the installation wizard creates Windows NT or Windows 2000 services only, you can create a shell script that performs a similar function on UNIX-based platforms. Use the `rc.was` example shell script, in the *`install_root`*`/bin` directory, to create the UNIX equivalent of a Windows service. Create a new shell script for each process that the UNIX-based system is to monitor and restart. Edit the `inittab` table of the operating system, to add an entry for each shell script you have created. This causes the UNIX-based system to call each shell script whenever the system reboots.

   Similar to a Windows service, each shell script monitors and restarts an individual WebSphere Application Server server process in a standalone environment. Comments in the header of the `rc.was` file provide instructions for identifying a WebSphere Application Server process, and show a sample `inittab` entry line for adding each copy of the script.

10. Review the summary information and click **Next** to install the product code or **Back** to change your specifications.

    When the installation is complete, the wizard displays the `install_root\logs\mq_install.log` installation log if you selected the embedded messaging feature and there are errors with its installation.

11. Review the `mq_install.log` installation log if it appears. Click **Next** to continue.

The wizard displays the registration panel.

12. Click **Next** to register the product, or deselect the check box and click **Next** to register at a later time.

    The installation wizard starts the First Steps tool.

13. Click **Finish** to close the installation wizard.

The installation wizard configures the product. It is not necessary to perform further configuration at this time.

Installing silently

Use this procedure to perform a silent installation.

1. Tailor the response file for WebSphere Application Server Network Deployment.

   You must tailor the options response file exactly for the installation to be successful.

2. Issue one of these commands to use your custom response file:

   Windows NT or Windows 2000
   : **Install.exe -options myoptionsfile**

   Linux/390 platforms
   : **install.sh -options ./myoptionsfile**

   Other Linux and UNIX-based platforms
   : **install -options ./myoptionsfile**

   You can find the sample options response file, `responsefile`, in the *`operating-system platform`* directory on the product CD.

3. **(Optional)** Reboot your machine in response to the prompt that might appear on Windows NT or Windows 2000 platforms.

   If you install the embedded messaging feature, `-P mqSeriesBean.active="true"`, certain conditions, such as a locked file, might require you to reboot. You have the option of rebooting immediately, after which the installation program resumes the installation at the point it left off. You can also defer rebooting to a convenient time, such as after the overall installation is complete.

You now understand how to install silently, using the response file. Examine the `log.txt` file for lines similar to these:

```
com.ibm.ws.install.actions.WSAdminInstallEarsAction, msg1, Installing ear with:
util\launcher.exe "C:\WebSphere\DeploymentManager\bin\wsadmin.bat" -conntype NONE
-c "$AdminApp install \\"C:/WebSphere/DeploymentManager/installableApps/adminconsole.
ear\\" {-node NodeName -cell CellName -server dmgr
-copy.sessionmgr.servername dmgr -appname adminconsole}"
```

This is an indicator that you have successfully installed the product.

Tailoring the options response file

Before using the **install -options ./myoptionsfile** command, for example, to invoke a silent installation, you must tailor the response file to add your selections.

Tailor the options response file precisely to enable the installation program to use it.

1. Locate the sample options response file. The file is named *responsefile* in the *`operating-system platform`* directory on the product CD-ROM.

2. Copy the file to preserve it in its original form. For example, copy it as *myOptionsFile*.
3. Edit the copy in your flat file editor of choice, on the target operating system. Read the directions within the response file to choose appropriate values.

   Note: Prepare for a silent installation on an AIX platform by using UNIX line-end characters (0x0D0A) to terminate each line of the options response file, as described in the Platform-specific tips for installing and migrating topic.
4. Make the first non-commented option **-silent** to have a silent installation.
5. Include custom option responses that reflect parameters for your system.

   Read the directions within the response file to choose appropriate values.
6. Save the file.

Usage scenario

Edit the version of the file that ships with the IBM WebSphere Application Server Network Deployment product.

```
# ********************************************
#
# Response file for WebSphere Application Server
# 5.0 Network Deployment Install

# Please follow the comments to use the response file
# and understand the various options.  You must carefully
# complete or change the various values. If the values are not completed
# properly, the install may be unsuccessful.
#
# IMPORTANT: ALL VALUES MUST BE ENCLOSED IN DOUBLE QUOTES ( "" ).
#
# *******************************************
# Below is the beginning of the response file that needs to be
# filled in by the user.
# *******************************************

# *******************************************
# The below value specifies silent install. This value
# indicates that the install will be silent. If you wish not to install
# silently, just delete this value.
# *******************************************

-silent


# *******************************************
# WebSphere Application Server Network Deployment Install Location
#
# Please specify the destination directory for the WebSphere
# Application Server Network Deployment installation.   You will need to change
# this for UNIX platforms. As an example for AIX, the value
# may be "/usr/WebSphere/DeploymentManager"
# *******************************************

-P wasBean.installLocation="C:\WebSphere\DeploymentManager"


# *******************************************
# Below are the features that you may choose to install.
# Set the following values to "true" or "false," depending upon whether
# you want to install the following features or not.
# *******************************************

# *******************************************
# Install Deployment Manager
# ********************************************

-P serverBean.active="true"


# ********************************************
#
# Begin Features for Web Services
#
# ********************************************

# *********
# Install Web Services
#
# This feature is for installation of Web Services.  If you
# want to install the next two features, you must set this to "true"
# else set this to "false."
# *********

-P webServicesBean.active="true"

# *********
# Install UDDI Registry
# *********

-P uddiBean.active="true"

# *********
# Install the Web Services Gateway
# *********

-P webServicesGatewayBean.active="true"

# ********************************************
#
# End Features for Web Services
#
# ********************************************


# ********************************************
# Install Embedded Messaging Client
# ********************************************

-P mqFeatureBean.active="true"

# ********************************************
# Embedded Messaging Client Install Location
#
# If you choose to install Embedded Messaging Client above,
# please specify an install location below for Windows platforms only.
# The directory may not be configured by the user for UNIX platforms
# as it is predetermined.
# ********************************************

-P mqFeatureBean.installLocation="C:\Program Files\IBM\WebSphere MQ"


# ************************************************************
# **         Support for Silent Coexistence
# **
# ** NOTE: You must uncomment and modify the properties in
# ** this section for silent coexistence to work properly.
# **
# ************************************************************
# ************************************************************
# Tell the installer that you want to perform coexistence
# ************************************************************

#-W coexistenceOptionsBean.doCoexistence="true"

# ************************************************************
# Set this property if you want to modify the default IHS
# and IHS Admin ports
# ************************************************************

#-W coexistencePanelBean.useIhs="true"

# ************************************************************
# The new value for the Bootstrap Port
# ************************************************************

#-W coexistencePanelBean.bootstrapPort="2810"

# ************************************************************
# The new values for the IHS and IHS Admin ports
# NOTE: These values are only used if
# coexistencePanelBean.useIhs is set to "true"
# ************************************************************

#-W coexistencePanelBean.ihsPort="81"
#-W coexistencePanelBean.ihsAdminPort="8009"

# ************************************************************
# The new values for the HTTP and HTTPs transports.
# ************************************************************

#-W coexistencePanelBean.httpTransportPort="9086"
#-W coexistencePanelBean.httpsTransportPort="9044"

# ************************************************************
# Thew new values for the admin console an secure admin
# console ports.
# ************************************************************

#-W coexistencePanelBean.adminConsolePort="9091"
#-W coexistencePanelBean.secureAdminConsolePort="9444"

# ************************************************************
# The new values for the csivServerAuthListener and
# the csivMultiAuthListener ports.
# NOTE: You can usually leave these set to 0
# ************************************************************

#-W coexistencePanelBean.csivServerAuthListenerAddr="0"
#-W coexistencePanelBean.csivMultiAuthListenerAddr="0"

# ************************************************************
# The new value for the sasSSLServerAuth port.
# ************************************************************

#-W coexistencePanelBean.sasSSLServerAuthAddr="0"

# ************************************************************
# The new values for the JMS Server Direct Address,
# JMS Server Security, and JMS Server QueuedAddress ports
# ************************************************************

#-W coexistencePanelBean.jmsServerDirectAddress="5569"
#-W coexistencePanelBean.jmsServerSecurityPort="5567"
#-W coexistencePanelBean.jmsServerQueuedAddress="5568"
```

```
# ************************************************************
# The new value for the soap connector address port
# ************************************************************

#-W coexistencePanelBean.soapConnectorAddress="8881"

# ************************************************************
# The new value for the DRS Client Address port
# ************************************************************

#-W coexistencePanelBean.drsClientAddress="7874"


# ************************************************************
# **         Support for Silent Migration
# **
# ** NOTE: You must uncomment and modify EVERY property
# ** in this section for silent migration to work properly.
# **
# ************************************************************

# ************************************************************
# The installer must be informed that you wish to operate on
# a previous version, so you must 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"

# *************************************************************
# Specify the location where the previous version is installed.
# *************************************************************

# -W previousVersionPanelBean.selectedVersionInstallLocation="/opt/WebSphere/AppServer"

# *************************************************************
# Specify 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/config/server-cfg.xml"

# *************************************************************
# Specify the version number of the previous version: 4.0 4.0.1 3.5 etc...
# *************************************************************
# -W previousVersionPanelBean.previousVersionSelected="4.0"

# ***********************************************************
# Uncomment the below line to indicate that you wish to
# migrate the previous version.
# ***********************************************************

# -W previousVersionPanelBean.migrationSelected="true"

# ***********************************************************
# Specify the directory where migration will backup
# information about the previous version.
# ***********************************************************

# -W migrationInformationPanelBean.migrationBackupDir="/tmp/migrationbackup"

# ***********************************************************
# Specify the directory where migration logs will be stored.
# ***********************************************************

# -W migrationInformationPanelBean.migrationLogfileDir="/tmp/migrationlogs"

# ******************************************
# Enter a node name, host name, and cell name for this
# installation.  The node name is used for administration,
# and must be unique within its group of nodes (cell).
# The host name is the DNS name or IP address for this computer.
# The cell name is a logical name for a group of nodes.
#
# You must replace the "nodenameManager" with the node name that
# you want the default node name to be (must be unique).
# Please change the nodename to your machine name in the
# "nodenameManager" default.
# ******************************************

-W nodeNameBean.nodeName="nodenameManager"

# *********
# You may replace the "nodenameNetwork" with the cell name
# that you want the create cell name to be.  Please change the
# nodename to your machine name in the "nodenameNetwork" default.
# *********

-W nodeNameBean.cellName="nodenameNetwork"

# *********
# You must replace "hostNameOrIPAddress" with either the hostname
# or IP address for the cell manager.
# *********

-W nodeNameBean.hostName="hostNameOrIPAddress"


# ******************************************
# Begin Installing Services
#
# The following options are to install Services for Websphere
# Network Deployment on Windows.  Using Services, you can start and
# stop services, and configure startup and recovery actions.
# You can ignore these or comment them out for other Operating Systems.
# ******************************************

-W serviceSettingsWizardBean.active="true"

# *********
# Install the WebSphere Network Deployment service
# *********
```

```
-W serviceSettingsWizardBean.wasChoice="true"

# *********
# If you chose to install a service above, then you must
# specify the User Name and Password which are required to
# install the Services. The current user must be admin or must
# have admin authority to install a Service. Also the username
# which is given here must have "Log On as a Service " authority
# for the service to run properly.
# *********

# *********
# 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
#
# *******************************************


# *******************************************
# Change the path to the prerequisite checker configuration
# file only if a new file has been provided.  This can be a
# relative path or an absolute path.  Make sure both the
# prereqChecker.xml and prereqChecker.dtd files are present at the provided path.
# *******************************************

-W osLevelCheckActionBean.configFilePath="waspc/prereqChecker.xml"


# *******************************************
# Product Registration Tool
#
# To launch the Product Registration Tool, please
# change the value to "true." This is only for
# GUI install.
# *******************************************

-W launchPRTBean.active="false"


# *******************************************
# First Steps
#
# If you would the First Steps to display at the end
# of the installation, please change the value to "true."
# *******************************************

-W firstStepsSequenceBean.active="false"
```

The response file

Silent installation is an installation method that reads all choices from the options response file, `responsefile`, which you must tailor before using.

Location

The sample options response file, `responsefile`, is on the product CD in the *`operating-system platform`* directory.

Usage notes

- This file is not a read-only file.
- Edit this file directly with your flat file editor of choice, such as WordPad on a Windows 2000 platform.
- This file is not updated by any product components.
- The file must exist to perform a silent installation. The installation program reads this file to determine installation option values when you install silently.
- Save the file in a location you can identify when you specify the fully qualified path as part of the installation command.

Platform-specific tips for installing and migrating

This topic is a collection of platform-specific tips that can help you install and migrate the base WebSphere Application Server product and the Network Deployment product.

This topic is divided into sections to make it easier for you to find applicable tips.

All platforms

- **Migration tools are in the migration subdirectory on the WebSphere Application Server, Version 5 CD-ROM.**

  A `migration` subdirectory on the CD-ROM installation image contains the WASPreUpgrade command line migration tool. It is intended for scenarios where you might save the currently installed configuration manually before installing the Version 5 product. One example of this situation is where you must upgrade the operating system as part of the Version 5 installation. You could migrate the earlier version, copy the migrated files in the backup directory to another system, update the operating system, restore the migrated files in their backup directory, install Version 5, and complete the migration.

  You can also use the migration directory on the CD-ROM to back up a Version 5 configuration in the event of an operating system upgrade. After the upgrade, you could restore the Version 5 configuration using the WASPostUpgrade tool.
- **Deselect the embedded messaging option during Network Deployment installation.**

  If you have already installed the embedded messaging client sub-feature on the machine, deselect the embedded messaging feature during Network Deployment installation. If you select it, a prerequisite check error states that the embedded messaging feature is already installed.
- **License files can contain bad characters in certain languages.**

  Use the graphical installation interface to avoid this problem.
- **InvalidExecutableException while starting jmsserver**

  You might get an exception while starting the jmsserver when you install Network Deployment first and then install the base WebSphere Application Server product and its embedded messaging feature on the same node. The error message is recorded in the *`install_root`*`/logs/jmsserver/SystemOut.log` file:

```
[9/5/02 14:35:37:818 EDT] 36349b90 JMSService
E MSGS0001E: Starting the Server failed with exception: com.ibm.ws.process.exception.
InvalidExecutableException: Error creating new process.
  002: No such file or directory
```

In addition, although `mq_install.log` files might appear to have run without error, the `createMQ.nodeName_jmsserver.log` file contains I/O Exceptions. The exceptions are due to a corrupted installation of the embedded messaging feature caused by installing Network Deployment before the base WebSphere Application Server product. The workaround is to uninstall both products, reinstall the base WebSphere Application Server product, and then reinstall the Network Deployment product.

Windows NT or Windows 2000 systems

- **If you are installing by downloading WebSphere Application Server from IBM, use another unzip product, such as WINZIP, instead of the PKWARE pkunzip utility to decompress the product archive.** Do not use the PKWARE pkunzip utility to decompress downloadable product archive. If you are using the downloadable archive file to install the WebSphere Application Server product, the pkunzip utility might not decompress the download image correctly. Use another utility (such as WinZip) to unzip the image.
- **The installation hangs or there are font problems that are often fixed by renaming the vpd.properties file.**

  Installing WebSphere Application Server on some international computers might result in installation failure or font problems. To work around this problem, follow these steps:

  1. Locate the `vpd.properties` file in the operating system installation directory.

     For example, `c:\windows` or `c:\winnt`.
  2. Rename this file to `vpd.properties_old`.

  The installation process recreates the file after the installation completes.
- **The WebSphere Application Server service fails to start.**

  During the installation of the base WebSphere Application Server product into the default path, `C:\Program Files\WebSphere\AppServer`, the installation wizard creates a log called **`C:\program`**.

  The presence of the `C:\program` file can cause any services that are installed under `C:\Program Files` to experience a start failure. It can also cause other programs that use Microsoft Windows Installer to install improperly.

  The workaround is to rename the file `C:\program` to another name, such as **`C:\mqlog`**, for example, after the WebSphere Application Server installation, before starting any services or installing any programs that use `C:\Program Files\` in the installation path.
- **Prepare for Adaptive Fast Path Architecture (AFPA) driver availability when migrating from IBM HTTP Server Version 1.3.19.x, or earlier.**

  The AFPA driver controls the fast response cache accelerator function, which is also known as the *cache accelerator*. The version of IBM HTTP Server installed with WebSphere Application Server shares the AFPA driver with any coexisting IBM HTTP Server. Uninstalling a coexisting Version 1.3.19.x (or earlier) IBM HTTP Server also uninstalls the common AFPA driver.

  When configured to use an AFPA driver that is no longer present, IBM HTTP Server generates errors as it starts and there is no response improvement from the cache accelerator. For example:

  ```
  [error] (9) Bad file descriptor: Afpa Device Driver open failed.
  ```

You can either restore the driver or disable the cache accelerator configuration. Restore the driver by reinstalling the later version of IBM HTTP Server after you uninstall the earlier version. To verify that the AFPA driver is installed and working, see if it is listed in device manager under `Non-Plug and Play Drivers`. Be sure to select the `Show hidden devices` option in the device manager view on Windows 2000 systems.

You can disable AFPA, by commenting out the following directives in the `httpd.conf` configuration file:

```
AfpaEnable
AfpaCache on
AfpaLogFile "C:\Program Files\IBM HTTP Server\log\afpalog" V-ECLF
```

- **Reboot the machine after uninstalling the embedded messaging feature.**

  If you install and then subsequently manually uninstall the embedded messaging feature on a Windows 2000 or Windows NT machine, you must reboot the machine before reinstalling the embedded messaging feature.

All UNIX-based operating systems platforms

- **Move all core dump files from the var/sadm/pkg directory.**

  The InstallShield for MultiPlatforms (ISMP) installation wizard iterates through all the directories in /var/sadm/pkg, assuming that each entry it finds is a directory. It then tries to open the `pkginfo` file underneath the directory. The ISMP wizard fails when it cannot find an entry under the core file. Move the core file out of the directory to avoid the problem.

- **Start the JMS server prior to running some Samples on UNIX platforms.**

  To run Samples that use JMS APIs, you must manually start the JMS server (`jmsserver`) before running the samples on UNIX platforms.

  To start the JMS server, complete the following steps:

  1. Open the administrative console: `http://localhost:9090/admin`.
  2. Expand **Servers > Application Servers > server1 > Server Components > JMS Servers**.
  3. Change the `Initial State` from **`STOP`** to **`START`**.
  4. Save the configuration and log out of the administrative console.
  5. Stop and restart the application server from the command line. For example,

     `stopServer.sh server1 followed by startServer.sh server1`

- **Use an option on the install or uninstall command to identify a temporary directory other than the default /tmp directory.**

  If the tmp disk does not have a large enough allocation, this message appears:

```
Error writing file =  There may not be enough temporary disk space.
Try using -is:tempdir to use a temporary directory on a partition with more disk space.
```

  Use the `-is:tempdir` installation option to specify a different temporary disk. For example, the following command uses the `/swap` file system as a temporary disk during installation:

  ```
  ./install -is:tempdir /swap
  ```

Solaris tips and techniques

- **Update the base WebSphere Application Server Version 5 product, or the Network Deployment product, to work with Java 2 SDK Version 1.4.1.**

The base WebSphere Application Server product, and the Network Deployment product, provide the Java 2 SDK Version 1.3 product as an internal component. The WebSphere Application Server products can also run using Java 2 SDK Version 1.4.

If you would like to experiment with using Java 2 SDK Version 1.4 with a WebSphere Application Server, Version 5 product, use the following unofficial procedure to update the internal SDK component:

1. Obtain the SDK 1.4 product. You can download the SDK from the Web site of Sun Microsystems, Inc. The *Installation: Resources for Learning* topic has a link to the site.
2. Install the SDK 1.4 product into a new subdirectory of `$WAS_HOME`, which is the root directory for the WebSphere Application Server product. The SDK installation program names the new subdirectory `$WAS_HOME/j2sdk1_4`. Installation instructions accompany the SDK 1.4 product, and are also available on the Web site of Sun Microsystems, Inc.
3. Create the following new subdirectory: `$WAS_HOME/ j2sdk1_4/ibm_bin`.
4. Copy all files from the `$WAS_HOME/java/ibm_bin` directory to the new `$WAS_HOME/ j2sdk1_4/ibm_bin` directory.
5. If the `$WAS_HOME/java/ibm_lib` directory exists, create the following new subdirectory: `$WAS_HOME/ j2sdk1_4/ibm_lib`
6. Copy the following files from `$WAS_HOME/java/ibm_lib/` directory to `$WAS_HOME/ j2sdk1_4/ibm_lib/` directory:
   - `ir.idl`
   - `orb.idl`
7. Copy the `$WAS_HOME/java/jre/lib/orb.properties` to `$WAS_HOME/ j2sdk1_4/jre/lib/` directory.
8. Copy all the files from the `$WAS_HOME/java/jre/lib/ext` directory to the `$WAS_HOME/ j2sdk1_4/jre/lib/ext` directory, except for `ibmorb.jar` and `ibmext.jar`.
9. Copy the following files from the `$WAS_HOME/java/jre/lib/sparc` directory to the `$WAS_HOME/ j2sdk1_4/jre/lib/sparc` directory:
   - `liborb.so`
   - `liborb_g.so`
10. Copy the following files from $WAS_HOME/java/lib/ directory to $WAS_HOME/ j2sdk1_4/lib/ directory.
    - `ir.idl`
    - `orb.idl`
11. Copy the file `$WAS_HOME/java/jre/lib/security/java.security` to the `$WAS_HOME/ j2sdk1_4/jre/lib/security` directory.
12. Ensure that you maintain all of the same directory and file access rights.
13. Rename the current `$WAS_HOME/java` directory to, for example, `$WAS_HOME/java1.3.1`.
14. Rename the `$WAS_HOME/j2sdk1_4` directory to `$WAS_HOME/java`.
15. Obtain the IBM JDK 1.4 `ibmorb.jar`, `ibmext.jar`, and `ibmtools.jar` from your IBM customer representative.
16. Place the IBM JDK 1.4 `ibmorb.jar` and `ibmext.jar` JAR files into the new `$WAS_HOME/java/jre/lib/ext` directory.
17. Place the IBM JDK 1.4 `ibmtools.jar` into the new `$WAS_HOME/java/lib` directory.

Note: The JDK is an internal component of WebSphere Application Server. IBM provides maintenance of this component through regular product maintenance, in PTFs and e-fixes. The overall goal is to provide you with a tested and stable operating environment. Although you are free to upgrade the JDK as described in this topic, IBM does not recommend replacing the JDK component outside of normal product maintenance and might require you to reinstall the shipped version of the JDK before submitting a problem.

- **Installing from a directory with a name beginning with disk* fails.**

  Installing WebSphere Application Server Version 5 from a folder that begins with disk* results in an error. Provide another name for the folder.
- **Double-byte character set (DBCS) characters are not supported in the name of the installation directory on Solaris systems.**

  Do not use double-byte characters in the installation directory name on a Solaris system.
- **Change Solaris kernal settings when installing the embedded messaging feature.**

  Several Solaris kernel values are typically too small for the embedded messaging feature. Starting the internal JMS server or client with insufficient kernel resources produces a First Failure Support Technology (FFST) file in the `/var/mqm/errors` directory.

  An example set of kernel values follows. You can change kernel values by editing the `/etc/system` file and rebooting the operating system.

  ```
  set shmsys:shminfo_shmmax = 4294967295
  set shmsys:shminfo_shmseg = 1024
  set shmsys:shminfo_shmmni = 1024
  set semsys:seminfo_semaem = 16384
  set semsys:seminfo_semmni = 1024
  set semsys:seminfo_semmap = 1026
  set semsys:seminfo_semmns = 16384
  set semsys:seminfo_semmsl = 100
  set semsys:seminfo_semopm = 100
  set semsys:seminfo_semmnu = 2048
  set semsys:seminfo_semume = 256
  set msgsys:msginfo_msgmap = 1026
  set msgsys:msginfo_msgmax = 4096
  set rlim_fd_cur=1024
  ```

AIX tips and techniques

- **Ignore DBCS messages when you do not requires DBCS. Otherwise, install the necessary patches.**

  While installing any WebSphere Application Server product, you might see the following messages on AIX 5.1 until you install the missing filesets:

  ```
  Operating system patches of particular concern:

  fileset X11.fnt.ucs.ttf_KR was not found on the system.
  v5.1.0.0 - Font required for Korean character display

  fileset X11.fnt.ucs.ttf_TW was not found on the system.
  v5.1.0.0 - Font required for Taiwanese character display

  fileset X11.fnt.ucs.ttf_CN was not found on the system.
  v5.1.0.0 - Font required for Chinese character display

  fileset jkit.Wnn6.base was not found on the system.
  v2.2.0.2 - Package required for Japanese input method
  ```
- **Apply a fix package to AIX 4.3.3 before installing the Simplified Chinese (GBK) version of WebSphere Application Server.** You can download the fix

from the AIX Fix Distribution Service Web site at URL: `http://techsupport.services.ibm.com/rs6k/fixdb.html`. Search the AIX Version 4 database for APAR number IY22531. You can find a link to the site in the *Installation: Resources for Learning* topic.

- **Use UNIX line-end characters (0x0D0A) to terminate each line of the options response file for silent installation.**

  During a silent installation on AIX machines, the response file passed to the installer program must not contain ASCII line-end characters (0x0D0A). The response file must contain UNIX line-end characters only. When the options response file contains ASCII line-end characters, the **install** command is unsuccessful and does not log or display an error. To verify the cause of failure, use the Java argument `-Dis.debug=1` on the **install** command. The debug information describes a service exception about invalid characters in the options response file.
- **The scroll bar disappears on the installation feature panel.**

  If the scroll bar disappears, use the up and down arrow keys to navigate the features in the Feature panel. Use the tab key to move the focus to the navigation. You can also use the mouse.
- **Ignore the following error messages that appear in the log.txt installation log.**

  Ignore:

```
WASBase, com.ibm.wizard.platform.aix.AixProductServiceImpl,
        wrn,   - WARNING:
Got invalid size of 0 for file:
        /usr/WebSphere/AppServer/config/cells/BaseApplicationServerCell/nodes/DefaultNode/spi.policy:
WASBase, com.ibm.wizard.platform.aix.AixRegistryServiceImpl, wrn, AixRegistryServiceImpl:
Error attempting to modify AIX VPD.
```

- **Install `Java130.rte.lib` Version 1.3.0 before installing the embedded messaging feature.** On AIX Version 4.3.3 and AIX Version 5, you must install `Java130.rte.lib` Version 1.3.0 to ensure that the embedded messaging feature installs correctly. To download a copy of Java 1.3.0, complete these steps:
  1. Go to `www.ibm.com\java: Java Technology Zone`.
  2. Go to the bottom of the page: **Most popular links.**
  3. Click **IBM Developer kit for AIX**.
  4. Click **Register & download**.
  5. Click **Java 1.3.0**.
- **Avoid a potential port conflict between the administrative console and the AIX WebSM system management console.**

  Port 9090 can be in use on both systems when installing the base WebSphere Application Server product or the Network Deployment product. The AIX WebSM system management server listens on port 9090 by default. If you suspect you have a port conflict, verify it by shutting down WebSphere Application Server. Then issue this command:

  ```
  netstat -an |grep 9090
  ```

  If you get a match, another process is already listening on port 9090. If you want the WebSM server and IBM WebSphere Application Server, Version 5 to coexist, change the WebSphere Application Server administrative console port when installing WebSphere Application Server, or after installation. Although not recommended, you can also disable the WebSM server. To disable the WebSM server, issue this command:

  ```
  /usr/websm/bin/wsmserver -disable
  ```

The command permanently disables WebSM server startup. If you want WebSM and WebSphere Application Server to coexist, either change the AIX WebSM port or the WebSphere Application Server administrative console port.

- **Cancel the installation and update your operating system before restarting the installation, if you install on an AIX machine and receive a message that a file set is missing, such as file set `X11.fnt.ucs.ttf`.**
- **Set up the display environment for a real root login on AIX systems.**

  In a normal root login, issue the command **`su`**. For a real root login, issue the command **`su -`**.

  Display settings for a normal root login are automatic. For a real root login, you must set your display environment properly to successfully view the GUI installation wizard. Otherwise, you see a message about `Preparing Java(tm) Virtual Machine...`, and seven rows of dots, but no installation GUI and no further messages. Refer to the documentation for AIX machines to determine proper display settings.
- **Avoid a DSAPI filter-loading error when the Lotus Domino Server starts.**

  On a UNIX-based operating system, if the Lotus Domino Web server starts using a non-root user, you are likely to generate a DSAPI filter-loading error:

  ```
  Error loading DSAPI filter.

  Filter not loaded: /usr/WebSphere/AppServer/bin/libdomino5_http.a
  ```

  Manually change the WebSphere Application Server `bin` directory permissions from `750` to `755` to run Lotus Domino Server as a non-root user and not generate the error. This change does, however, pose a security risk.

  You must also change permissions on the WebSphere Application Server `logs` directory to `777` to let Lotus Domino Server write to the log.

  If the Lotus Domino Server is started as root, the problem does not occur.

Linux tips and techniques

- **Avoiding utility hangs and accessing the deployment manager**

  The default Red Hat installation creates an association between the hostname of the machine and the loopback address, 127.0.0.1. In addition, the /etc/nsswitch.conf file is set up to use /etc/hosts before trying to look up the server using a name server (DNS). This loopback processing can hang utilities that start and stop a server, such as startServer.sh, stopManager.sh, and others, even though the server might successfully start or stop.

  The hang can also cause failures when adding nodes on a Linux deployment manager. If you have trouble synchronizing new nodes with the Linux deployment manager during addNode processing, ensure that the host name is defined properly. The default configuration has localhost defined in the `/etc/hosts` file. The default `/etc/nsswitch.conf` looks only at the host file and not the DNS server.

  To correct this problem, remove the 127.0.0.1 mapping to localhost in the `/etc/hosts` file or edit the name service configuration file (`/etc/nsswitch.conf`) to resolve the proper host name by using the name server.

  For example, remove the 127.0.0.1 mapping from the `/etc/hosts` file, which might look like this example:

  ```
  # IP Address                name of machine
  n.n.n.n                     hostname.domain.com     hostname
  127.0.0.1                   localhost
  ```

Otherwise, change the `etc/nsswitch.conf` file to search DNS before searching the hosts file.

For example, `hosts : dns files`

First Steps tool tips

First Steps is a post-installation ease-of-use tool for directing WebSphere Application Server elements from one place. Options dynamically appear on the First Steps panel, depending on features you install. With all options present, you can use First Steps to start or stop the application server, verify the installation, access the InfoCenter, run the Application Assembly Tool, access the administrative console, access the Samples Gallery, or launch the product registration.

First Steps starts automatically at the end of the installation. If it is not running, start First Steps from the command line:

- On Windows NT or Windows 2000: *`install_root`*`\bin\firststeps.bat`
- On UNIX-based server platforms: *`install_root`*`/bin/firststeps.sh`

Using the installation verification test

After installing the product, you are ready to use the installation verification test (IVT).

The IVT program scans product log files for errors and verifies core functionality of the product installation.

1. If you started the administrative server of a Version 3.5.x and later, or Version 4.0.x and later WebSphere Application Server product to let the installation wizard export its configuration and applications to Version 5, stop the server before running the IVT.

   Otherwise, the Version 5 server might not start due to port conflicts between the two servers.
2. Select **Verify Installation** from the First Steps panel and observe the results in the First Steps status window. A log file for the installation is created and stored in the `logs` directory with the name of `ivt.log`.
3. If the First Steps tool is not running, start it from the `bin` directory.
   - On Windows NT or Windows 2000 platforms: firststeps.bat
   - On UNIX-based platforms: ./firststeps.sh

   You can also run First Steps from the Windows NT or Windows 2000 Start Menu.
4. You can also start the IVT tool from the `bin` directory.
   - On Windows NT or Windows 2000 platforms: ivt
   - On UNIX-based platforms: ./ivt.sh

The IVT program starts the server automatically if the server is not running. Once the server initializes, the IVT runs a series of verification tests and reports pass or fail status in a console window. It also logs results to the *install_root* `\logs\ivt.log` file.

Troubleshooting the installation

If you did not receive the *Successful installation* message, troubleshoot the installation, as described below.

1. Use the First Steps tool to run the installation verification test (IVT). Check the *install_root* `\logs\ivt.log` file for a summary of test results. Correct any errors and retry.

   The installation wizard automatically starts the First Steps tool at the end of installation.
2. Check the installation log files for errors after installing:

   During installation, a single entry in the `log.txt` file points to the temporary log file, either `%TEMP%\log.txt` on Windows NT or Windows 2000, or `/tmp/log.txt` on UNIX-based platforms. The installation program copies the file to location shown at the end of the installation.

   Installation log locations when installing WebSphere Application Server

Component	**Installation log path name**	
	Windows NT or Windows 2000 system temp directory	**UNIX-based operating system: `/tmp/`**
Embedded messaging feature (based on the MQSeries) prerequisites error log	mq_prereq.log	
Component	**Installation log path name**	
	Windows NT or Windows 2000: *install_root*\logs\	**UNIX-based operating system: *install_root*/logs/**
WebSphere Application Server	log.txt	
IBM HTTP Server	ihs_log.txt	
Embedded messaging feature (based on the MQSeries) installation log	mq_install.log	
Embedded messaging feature (based on the MQSeries) configuration log	createMQ.*node*.server1.log	
Default Application	installDefaultApplication.log	
Samples Gallery	installSamples.log	
Administrative console	installAdminConsole.log	
Migration tools	WASPostUpgrade.log	

Installation log locations when installing Network Deployment

Component	**Installation log path name**	
	Windows NT or Windows 2000 system temp directory	**UNIX-based operating system: `/tmp/`**
Embedded messaging feature (based on the MQSeries) prerequisites error log	mq_prereq.log	
Component	**Installation log path name**	
	Windows NT or Windows 2000: *install_root*\logs\	**UNIX-based operating system: *install_root*/logs/**
Network Deployment	log.txt	
Embedded messaging feature (based on the MQSeries) installation log	mq_install.log	
Administrative console	installAdminConsole.log	
File Transfer	installFiletransfer.log	
Migration tools	WASPostUpgrade.log	

Note: The *install_root* symbol used above represents the path where you choose to install the product. It is not an environment variable.

3. Turn on tracing if the installation logs do not contain enough information to determine the cause of the problem.
   - Report the `stdout` and `stderr` logs to the console window, by adding the `-is:javaconsole` parameter to the **Install.exe** or **install** command:
     – On Windows NT or Window 2000 platforms:
     ```
     Install.exe -is:javaconsole
     ```
     Or capture it to a file with:
     ```
     Install.exe -is:javaconsole > drive:\captureFileName.txt
     ```
     – On Linux/390 platforms:
     ```
     ./install.sh -is:javaconsole
     ```
     Or capture it to a file with:
     ```
     ./install.sh -is:javaconsole > captureFileName.txt 2>&1
     ```
     – On other Linux platforms and UNIX-based operating platforms:
     ```
     ./install -is:javaconsole
     ```
     Or capture it to a file with:
     ```
     ./install -is:javaconsole > captureFileName.txt 2>&1
     ```
   - Capture additional information to a log of your choice with the `-is:log` *`filename`* option.
   - Turn on additional installation logging by passing the `-W Setup.product.install.logAllEvents="true"` parameter to the **Install.exe**, **install.sh**, or **install** command:
     – On Windows NT or Window 2000 platforms:
     ```
     Install.exe -W Setup.product.install.logAllEvents="true"
     ```
     – On Linux/390 platforms:

```
install.sh -W Setup.product.install.logAllEvents="true"
```

– On other Linux platforms and UNIX-based operating platforms:

```
install -W Setup.product.install.logAllEvents="true"
```

If you install on an AIX 5.1 system, with maintenance level 2, it is possible that the Web-based system manager, a standard component of AIX systems, already uses port 9090. When starting the server you get information that port 9090 is already in use. To resolve the conflicting port use, change the port assignment for the HTTP_TRANSPORT_ADMIN port in the `server.xml` file. The file path is:

```
usr/websphere/appserver/config/cells
            /cell/nodes/node/servers/server1/server.xml
```

4. Use the First Steps tool or the command line method to start the application server.

   To start the server from the command line:

   - On Windows NT or Windows 2000 platforms: *install_root*\bin\startServer server1
   - On UNIX-based server platforms: *install_root*/bin/startServer.sh server1

5. Verify whether the server starts and loads properly, by looking for a running Java process and the *Open for e-business* message in the `SystemOut.log` and `SystemErr.log` files. If there is no Java process or the message does not appear, examine the same logs for any miscellaneous errors. Correct any errors and retry.

   You can find the `SystemOut.log` and `SystemErr.log` files in the *`install_root`*`\logs\server1` (Windows) or *`install_root`*`/logs/server1` (UNIX-based) directory.

6. Use the First Steps tool or the command line method to stop the application server, if it is running, and to start the deployment manager.

   To stop the server from the command line:

   - On Windows NT or Windows 2000 platforms: *install_root*\bin\stopServer server1
   - On UNIX-based server platforms: *install_root*/bin/stopServer.sh server1

   To start the deployment manager from the command line:

   - On Windows NT or Windows 2000 platforms: *install_root*\bin\startServer dmgr
   - On UNIX-based server platforms: *install_root*/bin/startServer.sh dmgr

7. Verify whether the server starts and loads properly by looking for a running Java process and the *Server dmgr open for e-business* message in the `dmgr_stdout.log` and `dmgr_stderr.log` files. If there is no Java process or the message does not appear, examine the same logs for any miscellaneous errors. Correct any errors and retry.

8. Refer to the plug-in configuration documentation, if you have installed plug-ins and the Web server does not come up properly.

9. Start the Snoop servlet.

   Note: In a Network Deployment environment, the Snoop servlet is available in the domain only if you included the DefaultApplication when adding the application server to the cell. The -includeapps option for the **addNode** command migrates the DefaultApplication to the cell. If the application is not present, skip this step.

a. Point your browser to URL: `http://localhost:9090/snoop` to test the internal HTTP transport provided by the application server. Point your browser to `http://localhost/snoop` to test the Web server plug-in.

b. Start the application server.

c. Start the IBM HTTP Server or the Web server you are using.

   Use a command window to change directories to the IBM HTTP Server installed image, or to the installed image of your Web server. Issue the appropriate command to start the Web server, such as these commands for IBM HTTP Server:

   To start the IBM HTTP Server from the command line:

   - On Windows NT or Windows 2000 platforms: **apache**
   - On UNIX-based server platforms: **./apachectl start**

d. Verify that Snoop is running.

   Either URL should display the **Snoop Servlet - Request/Client Information** page.

10. Start the WebSphere Application Server administrative console.

    a. Start the application server.

    b. Point your browser to URL: `http://localhost:9090/admin`.

    c. Type any ID and click **OK** at the administrative console window.

    The server starts. The administrative console starts. You can access the administrative console through the browser. The administrative console accepts your login.

11. Federate the base application server into the cell.

    To add the base application server into the cell:

    - On Windows NT or Windows 2000: *install_root*\AppServer\bin\addNode.bat localhost 8879
    - On UNIX-based server platforms: *install_root*/AppServer/bin/addNode.sh localhost 8879

12. Verify that the application server was incorporated into the cell.

    The command window should display a sequence of messages when you issue the addNode command:

```
Begin federation of node xxxx with deployment manager at localhost:8879.
Successfully connected to deployment manager Server: localhost:8879
Creating Node Agent configuration for node: xxxx
Reading configuration for Node Agent process: xxxx
Adding node xxxx configuration to cell: AdvancedDeploymentCell
Performing configuration synchronization between node and cell.
Launching Node Agent process for node: xxxx
Node Agent launched. Waiting for initialization status.
Node Agent initialization completed successfully. Process id is: 3012
Node xxxx has been successfully federated.
```

    The last message is an indicator of success. There should also be a second java.exe process running. The stdout.log and stderr.log in the *node_name* directory should contain a *Server node_name open for e-business* message.

13. Resolve any IP address caching problems.

    By default, the Java 2 SDK caches the IP address for the DNS naming lookup. After resolving the host name successfully, the IP address stays in the cache. By default, the cache entry remains forever. This default IP caching mechanism can cause problems, as described below.

    Problem scenario 1

Suppose the application server at host1.ibm.com has an initial IP address of 1.2.3.4. When a client at host2.ibm.com conducts a DNS lookup of host1.ibm.com, it stores the 1.2.3.4 address in the cache. Subsequent DNS name lookups return the cached value, 1.2.3.4. The cached value is not a problem until the host1.ibm.com IP address changes, to 5.6.7.8, for example. The client at host2.ibm.com does not retrieve the current IP address but always retrieves the previous address from the cache. If this scenario occurs, the client cannot reach host1.ibm.com unless you stop and restart the client process.

Problem scenario 2

Suppose the application server at host1.ibm.com has an initial IP address of 1.2.4.5. Although the IP address of the application server does not change, a network outage can record an exception code as the IP address in the cache, where it remains until the client is restarted on a working network. For example, if the client at host2.ibm.com disconnects from the network because of an unplugged cable, the disconnected lookup of the application server at host1.ibm.com fails. The failure causes the IBM Developer Kit to put the special exception code entry into the IP address cache. Subsequent DNS name lookups return the exception code, which is java.net.UnknowHostException.

IP address caching and WebSphere Application Server process discovery

If you change the IP address of a federated WebSphere Application Server node, processes running in other nodes cannot contact the changed node until you stop and restart them.

If a deployment manager process starts on a disconnected node, it cannot communicate with cell member processes until you stop and restart the deployment manager process. For example, plugging in an unplugged network cable does not restore proper addresses in the IP cache until the deployment manager process is restarted.

Using the IP address cache setting

You can always stop and restart a deployment manager process to refresh its IP address cache. However, this might be expensive or inappropriate.

The networkaddress.cache.ttl (public, JDK1.4) and sun.net.inetaddr.ttl (private, JDK1.3) parameters control IP caching. The value is an integer that specifies the number of seconds to cache IP addresses. The default value, -1, specifies to cache forever. A value of 0 specifies to never cache.

Using a zero value is not recommended for normal operation. If you do not anticipate network outages or changes in IP addresses, use the cache forever setting. Never caching introduces the potential for DNS spoofing attacks.

For more information

The Java 2 SDK, Standard Edition 1.4 Web site at http://java.sun.com/j2se/1.4/docs/guide/net/properties.html describes the private sun.net.inetaddr.ttl property, which also works in Java 2 SDK, Standard Edition 1.3. The *Networking* section of the Java 2 SDK, Standard Edition 1.4 Web site at http://java.sun.com/j2se/1.4.1/jcp/beta/ describes a change in the behavior of the java.net.URLConnection class.

Migrating and coexisting

Determine whether you have an existing version of WebSphere Application Server installed on the machine where you plan to install your Version 5 product.

If you have a previous version, you must plan whether to copy the configuration and applications of the previous version to the new version, which is *migration*.

Migration does not uninstall the previous version. The earlier release is still functional. If you run it at the same time as the Version 5 installation, the two versions are said to be *coexisting*. You can choose to provide non-default port assignments for Version 5, to support coexistence by selecting the coexistence option.

WebSphere Application Server contains migration tools that provide all migration functionality. The installation wizard can call the migration tools, or you can call them manually at a later time. The migration tools migrate applications and configuration information to the new version, as described in detail in the *Migrating* topic (welc_migrating in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html) and in the Configuration mapping during migration topic.

You can also find information about migrating the configuration and applications from a previous version of WebSphere Application Server to Version 5 in the IBM Redbook, Migrating to WebSphere V5.0: An End-to-End Migration Guide, SG24-6910-00.

This section contains the following topics:
- Overview of migration and coexistence
- Migrating to Network Deployment
- Configuration mapping during migration
- Migrating administrative configurations manually
  - Migrating from an unsupported operating system
  - WASPreUpgrade command
  - WASPostUpgrade command
- Configuring WebSphere Application Server after migration
- Coexistence support
- Setting up Version 3.5.x and Version 5 coexistence
- Setting up Version 4.0.x and Version 5 coexistence
- Setting up Version 5 coexistence
- Federating multiple Version 5 installation instances
- Port number settings in WebSphere Application Server versions

Overview of migration and coexistence

In overview, the migration tools perform a fairly routine migration from Version 4 to Version 5. For example, Java 2 Platform, Enterprise Edition (J2EE) 1.2 EAR files in Version 4 work in Version 5 of WebSphere Application Server, which also supports J2EE 1.3. Similarly, it is not necessary to redeploy EJB 1.1 JAR files when moving them from Version 4 to Version 5, which also supports EJB 2.0 JAR files.

In overview, the migration from Version 3.5 to Version 5 involves significant changes in application structures, development, and deployment. The migration tools assist in this transition, by migrating system configurations and creating J2EE artifacts, including mapping previous security settings to J2EE security roles. These security mappings let you access migrated assets during the transition. The migration tools create initial J2EE enterprise applications based on Version 3.5.x configurations. However, because of the significant changes in the application structures, carefully test and fine tune migrated applications using development and deployment tools.

There are four combinations of migration and coexistence that you can select from the installation wizard or during silent installation:

- Migrate only
- Coexist only
- Migrate and coexist
- Neither migrate nor coexist

If you neither migrate nor coexist with an earlier version of WebSphere Application Server, you are choosing to ignore the previous installation. You can run only one version at a time because of conflicting default port assignments, although it is possible that both versions might run at the same time without conflict if you use non-default ports in the earlier version. To resolve conflicting port assignments, the coexistence panel lets you manually assign ports for Version 5 to ensure that it can run with an earlier version.

You can specify port assignments for coexistence on the installation wizard coexistence panel, by editing configuration files manually, by wsadmin scripting, or by using the **Servers > Application Servers > server1 > End Points** administrative console page.

WebSphere Application Server products uses a different set of default port numbers for coexistence than it does for the initial installation of a WebSphere Application Server product.

Coexistence port definitions

Port names, default value, and description	Value
• Installation wizard name: **HTTP Transport Port** • Name in base WebSphere Application Server server.xml configuration file: HTTPTransport_1 EndPoint_1 • Name in Network Deployment server.xml configuration file: **Not applicable** • virtualhosts.xml: HostAlias_1 • Silent installation response file option name: coexistencePanelBean.httpTransportPort • Base WebSphere Application Server, default value: 9080 • Network Deployment, default value: Not applicable • InfoCenter description: Port used for request queues between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application reside (crun_transport in the InfoCenter)	9085

Port names, default value, and description	Value
• Installation wizard name: **HTTPS Transport Port** • Name in base WebSphere Application Server server.xml configuration file: HTTPTransport_2 EndPoint_2 • Name in Network Deployment server.xml configuration file: **Not applicable** • virtualhosts.xml: HostAlias_3 • Silent installation response file option name: coexistencePanelBean.httpsTransportPort • Base WebSphere Application Server, default value: 9443 • Network Deployment, default value: Not applicable • InfoCenter description: Port used for request queues between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application reside (crun_transport)	9444

Port names, default value, and description	Value
• Installation wizard name: **HTTP Admin Console Port** • Name in base WebSphere Application Server server.xml configuration file: HTTPTransport_3 EndPoint_3 • Name in Network Deployment server.xml configuration file: **HTTPTransport_1 EndPoint_1** • virtualhosts.xml: HostAlias_4 • Silent installation response file option name: coexistencePanelBean.adminConsolePort • Default value: 9090 • InfoCenter description: Port used for request queues between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application (including the administration console application) reside (crun_transport)	9091

Port names, default value, and description	Value
• Installation wizard name: **HTTPS Admin Console Secure Port** • Name in base WebSphere Application Server server.xml configuration file: HTTPTransport_4 EndPoint_4 • Name in Network Deployment server.xml configuration file: **HTTPTransport_1 EndPoint_2** • virtualhosts.xml: HostAlias_5 • Silent installation response file option name: coexistencePanelBean.secureAdminConsolePort • Default value: 9043 • InfoCenter description: Port used for request queues between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application (including the administration console application) reside (crun_transport)	9044

Port names, default value, and description	Value
• Installation wizard name: **Bootstrap port** • Name in base WebSphere Application Server serverindex.xml configuration file: BOOTSTRAP_ADDRESS EndPoint_1 • Name in Network Deployment serverindex.xml configuration file: BOOTSTRAP_ADDRESS **EndPoint_2** • Silent installation response file option name: coexistencePanelBean.bootstrapPort • Base WebSphere Application Server, default value: 2809 • Network Deployment, default value: 9809 • InfoCenter description: The bootstrap port setting together with the host name forms the initial context for JNDI references. (tnam_develop_naming)	2810

Note: The default port assignment for the Network Deployment product is 9809. The default port assignment for the base WebSphere Application Server product is 2809. You might want to use a coexistence value for a coexisting Network Deployment product that is more in line with the default value. For example, you might prefer to use a value of 9810.

Port names, default value, and description	Value
• Installation wizard name: **SOAP Connector Address** • Name in base WebSphere Application Server serverindex.xml configuration file: SOAP_CONNECTOR-ADDRESS EndPoint_2 • Name in Network Deployment serverindex.xml configuration file: SOAP_CONNECTOR-ADDRESS **EndPoint_4** • Silent installation response file option name: coexistencePanelBean.soapConnectorAddress • Default value: 8880 • InfoCenter description: Port used for the Simple Object Access Protocol (SOAP) connector (cxml_connector)	8881

Port names, default value, and description	Value
• Installation wizard name: **DRS client address** • Name in serverindex.xml configuration file: DRS_CLIENT_ADDRESS EndPoint_3 • Silent installation response file option name: coexistencePanelBean.drsClientAddress • Default value: 7873 • InfoCenter description: Port used to configure the Data Replication Service (DRS), which is a JMS-based message broker system for dynamic caching (tprf_dynamiccache)	7874

Port names, default value, and description	Value
• Installation wizard name: **JMS server queued address** • Name in base WebSphere Application Server serverindex.xml configuration file: JMSSERVER_QUEUED_ADDRESS EndPoint_4 • Name in Network Deployment serverindex.xml configuration file: **Not applicable** • Silent installation response file option name: coexistencePanelBean.jmsServerQueuedAddress • Default value: 5558 • InfoCenter description: Port used to configure the WebSphere JMS provider topic connection factory settings (umj_ptcfw)	5568

Port names, default value, and description	Value
• Installation wizard name: **JMS server security port** • Name in base WebSphere Application Server server.xml configuration file: JMSServer_1 Address_3 • Name in Network Deployment server.xml configuration file: **Not applicable** • Silent installation response file option name: coexistencePanelBean.jmsServerSecurityPort • Default value: 5557 • InfoCenter description: Port used in JMS security processing (cm_secty)	5567

Port names, default value, and description	Value
• Installation wizard name: **JMS Server Direct Address** • Name in base WebSphere Application Server serverindex.xml configuration file: JMSSERVER_DIRECT_ADDRESS EndPoint_5 • Name in Network Deployment serverindex.xml configuration file: **Not applicable** • Silent installation response file option name: coexistencePanelBean.jmsServerDirectAddress • Default value: 5559 • InfoCenter description: Port used to configure the WebSphere JMS provider topic connection factory settings (umj_ptcfw)	5569

Port names, default value, and description	Value
• Installation wizard name: **SAS SSL ServerAuth Address** • Name in serverindex.xml configuration file: SAS_SSL_SERVERAUTH_LISTENER_ADDRESS EndPoint_6 • Silent installation response file option name: coexistencePanelBean.sasSSLServerAuthAddr • Base WebSphere Application Server, default value: 0; With security, assign your own non-conflicting value: *nnnn* • Network Deployment, default value: 9401 but is not enabled until you enable security • InfoCenter description: Port used to listen for requests to use the server (tsec_inboundtransport)	0 **See note.**

Note: Assign a coexistence port other than 0, which dynamically takes a free port. You must have a fixed port that clients can use to find the SAS SSL ServerAuth Address. There is no mechanism for clients to use to find a dynamically assigned port.

Port names, default value, and description	Value
• Installation wizard name: **CSIV2 ServerAuth Listener Address** • Name in base WebSphere Application Server serverindex.xml configuration file: CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS EndPoint_7 • Name in Network Deployment serverindex.xml configuration file: CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS **EndPoint_8** • Silent installation response file option name: coexistencePanelBean.csivServerAuthListenerAddr • Base WebSphere Application Server, default value: 0; With security, assign your own non-conflicting value: *nnnn* • Network Deployment, default value: 9403 but is not enabled until you enable security • InfoCenter description: Port used to listen for requests to use the server	0 **See note.**

Note: Assign a coexistence port other than 0, which dynamically takes a free port. You must have a fixed port that clients can use to find the CSIV2 ServerAuth Listener Address. There is no mechanism for clients to use to find a dynamically assigned port.

Port names, default value, and description	Value
• Installation wizard name: **CSIV2 MultiAuth Listener Address** • Name in base WebSphere Application Server serverindex.xml configuration file: CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS EndPoint_8 • Name in Network Deployment serverindex.xml configuration file: CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS **EndPoint_7** • Silent installation response file option name: coexistencePanelBean.csivMultiAuthListenerAddr • Base WebSphere Application Server, default value: 0; With security, assign your own non-conflicting value: *nnnn* • Network Deployment, default value: 9402 but is not enabled until you enable security • InfoCenter description: Port used to listen for requests to use the server (tsec_inboundtransport)	0 **See note.**

Note: Assign a coexistence port other than 0, which dynamically takes a free port. You must have a fixed port that clients can use to find the CSIV2 MultiAuth Listener Address. There is no mechanism for clients to use to find a dynamically assigned port.

Port names, default value, and description	Value
• Installation wizard name: **ORB Listener Port** • Name in base WebSphere Application Server serverindex.xml configuration file: Not applicable • Name in Network Deployment serverindex.xml configuration file: ORB_LISTENER_ADDRESS EndPoint_5 • Silent installation response file option name: Not applicable - See note. • Base WebSphere Application Server default value: Not applicable • Network Deployment default value: 9100 • InfoCenter description: Port used to listen for requests to use the server (tsec_inboundtransport)	9101

Note: Set the ORB Listener Port after a silent installation of the Network Deployment product as described in the ″Configuring inbound transports″ topic (tsec_inboundtransport in the InfoCenter).

Port names, default value, and description	Value
• Network Deployment installation wizard name: **Cell Discovery Address** • Base WebSphere Application Server installation wizard name: Not applicable • Default value: 7277 • Description: Port used in cell discovery	9989

Note: Set the Cell discovery address after a silent installation of the Network Deployment product as described in the *Cell discovery* (trun_watch_cell) topic.

If you choose to both migrate and coexist, you are selecting two tasks with roughly opposite ends. The goal of migration is to reconstruct your earlier version in a nearly identical Version 5 environment, including applications, configuration settings, URI descriptors, and Web server context roots. The goal of coexistence is to create an environment that is not in conflict with an earlier version, so far as port settings are concerned. If you select both migration and coexistence, the installation wizard migrates the earlier version configuration and applications. Then the wizard changes port settings to support coexistence with the earlier version, which means that both nodes can start and run at the same time. Migration occurs first. Coexistence processing occurs next and alters the following configuration files to the default settings shown above:

- The virtualhosts.xml file:
  – HTTP Transport Port
  – IBM HTTP Server Port
  – HTTPS Transport Port
  – HTTP Admin Console Port
  – HTTPS Admin Console Secure Port
- The serverindex.xml file:
  – Bootstrap port
  – SOAP Connector Address
  – DRS client address
  – JMS server queued address
  – JMS Server Direct Address

- SAS SSL ServerAuth Address
  - CSIV2 ServerAuth Listener Address
  - CSIV2 MultiAuth Listener Address
- The server1\server.xml file
  - HTTP Transport Port
  - HTTPS Transport Port
  - HTTP Admin Console Port
  - HTTPS Admin Console Secure Port
  - JMS server security port

You must then start server1 and use it to change any ports that are in conflict, so that there is no conflict with the earlier version. Then you can run both versions at the same time.

There are two other problems that might occur when you choose both migration and coexistence:

- One problem is conflicting context roots when attempting to share the same Web server.

  Follow the procedure in the *Migrating plug-ins, one machine at a time* topic (tins_websmig2) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html, to learn how to configure a Web server for sharing between WebSphere Application Server versions. The topic links to procedures that are specific to Web server products, such as the *Migrating IBM HTTP Server to support multiple WebSphere Application Server versions* topic (tins_websIHS).

  A procedure for configuring a Web server for sharing between WebSphere Application Server versions is described in the InfoCenter for the base WebSphere Application Server product.
- The other problem occurs if you install more than two WebSphere Application Server Version 5 products on the same machine.

  You must resolve port conflicts as described in the Setting up Version 5 coexistence topic.

Coexistence is described as an optional step below. Migration is also optional. To migrate applications and configurations to Version 5, follow this procedure:

1. Prepare to migrate or update product prerequisites and corequisites to supported versions.

   Review the prerequisites on the IBM WebSphere Application Server supported hardware, software, and APIs Web site at http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html for current requirements.
2. Prepare to use the migration function of the installation wizard, to migrate silently, or to migrate manually.
   - You have the option of letting the installation wizard use the migration tools to migrate an existing WebSphere Application Server 3.5.x or 4.0.x version to Version 5 automatically.

     The installation wizard detects a previously installed version, and displays a migration and coexistence panel. You can select either option, both options, or no option.

a. Start the administrative server of WebSphere Application Server Standard Edition (Version 3.5.x) or WebSphere Application Server Advanced Edition (Versions 3.5.x or 4.0.x).

   Starting the administrative server allows the migration tools to use XMLConfig to export the configuration data repository. It is not necessary that you start the administrative server for WebSphere Application Server Advanced Single Server Edition, Version 4. The migration tools use the XMI configuration files directly.

b. Select the migration option and click **Next** when using the installation wizard. When using the silent installation method, select the appropriate option.

   Enter a fully qualified path for the backup directory, where the migration tools store and read the saved configuration and other files. During migration, the wizard backs up the old version administrative configuration and user data files, but does not uninstall the old version. The wizard automatically configures the new installation with the previous configuration data. It also copies some application data to the newer version.

   The installation wizard uses the WASPreUpgrade migration tool to export the current administrative configuration. After this phase completes, the wizard runs the second phase, which uses the WASPostUpgrade migration tool to convert the backed-up administrative configuration into the new Version 5 configuration. You can stop the administrative server of the earlier version after the migration is complete. You must stop the earlier version server before starting the Version 5 server.

   The installation program prompts you for the following information during migration:

   – Backup directory.
   – Currently installed WebSphere Application Server directory. This directory must differ than the one where you install Version 5.
   – Name of the administrative node of the previous version. The default is the computer name, but both the previous node name and the Version 5 node name must be the same.

     To determine the administrative node name of the previous version, you can do either of these actions:

     - Start the administrative console of the earlier release. View the **Nodes** folder to determine the node name for the system you want to migrate.
     - Perform an XMLConfig export from the administrative console. Review the output xml file to look for <node action="update" name="*nodename*"> to determine the node name.

c. Plan to complete the migration after the installation and migration are complete.

   Follow these steps to complete the migration:

   1) Stop the administrative server from the earlier version.
   2) Load existing applications into your development environment and fix any known problems.
   3) Start the servers in the new installation.
   4) Deploy the changed applications in a test environment.
   5) Test all applications thoroughly.

6) Follow normal test procedures as you move the test environment into production.

The installation wizard backs up the earlier version but does not uninstall it, exports the current configuration, and migrates the configuration to the new installation. If there are errors during the WASPreUpgrade phase, such as the administrative server from the earlier version is not running, the installation wizard does not run the WASPostUpgrade phase. The wizard displays the `WASPreUpgrade.log` and `WASPostUpgrade.log` log files if there are errors.

- You can migrate the configuration of a previous version to the Network Deployment product using the installation wizard or during a silent installation. Specify non-conflicting port assignments in the silent options response file to silently configure for coexistence during the silent installation. You can also migrate an earlier version configuration silently. You must supply values for silent migration options when you tailor the options response file:
  a. Direct the installation program to migrate a previous installation by specifying the -W previousVersionDetectedBean.previousVersionDetected= ″true″ parameter.
  b. Use one of the following values to identify the specific version to migrate:
     – For WebSphere Application Server Advanced Edition, Versions 3.x and 4.0.x, use one of the following parameters:
       - -W previousVersionPanelBean.selectedVersionEdition=″AE″
       - -W previousVersionPanelBean.selectedVersionEdition=″advanced″
     – For WebSphere Application Server Advanced Single Server Edition, Version 4.0.x, use the -W previousVersionPanelBean.selectedVersionEdition= ″AEs″ parameter.
     – For WebSphere Application Server Standard Edition, Version 3.x, use the -W previousVersionPanelBean.selectedVersionEdition= ″standard″ parameter.
  c. Specify the location where the previous version is installed in the -W previousVersionPanelBean.selectedVersionInstallLocation= ″/opt/WebSphere/AppServer″ parameter.
  d. Specify the path to the configuration file:
     – For WebSphere Application Server Advanced Edition, Versions 3.x and 4.0.x, use the -W previousVersionPanelBean.selectedVersionConfigFile= ″/opt/WebSphere/AppServer/config/admin.config″ parameter.
     – For WebSphere Application Server Advanced Single Server Edition, Version 4.0.x, use the -W previousVersionPanelBean.selectedVersionConfigFile= ″/opt/WebSphere/AppServer/config/server-cfg.xml″ parameter.
     – For WebSphere Application Server Standard Edition, Version 3.x, use the -W previousVersionPanelBean.selectedVersionConfigFile= ″/opt/WebSphere/AppServer/config/server-cfg.xml″ parameter.
  e. Specify the version number of the previous version, such as ″4.0″, ″4.0.1″, or ″3.5″, in the -W previousVersionPanelBean.previousVersionSelected= ″4.0″ parameter.
  f. Indicate that you are selecting the version you have identified with the -W previousVersionPanelBean.migrationSelected= ″true″ parameter.

g. Specify the backup directory where WASPreUpgrade is to store information from the previous version with the -W migrationInformationPanelBean.migrationBackupDir= ″/tmp/migrationbackup″ parameter.
h. Specify the directory for storing migration logs with the -W migrationInformationPanelBean.migrationLogfileDir= ″/tmp/migrationlogs″ parameter.

- Migrate manually if you prefer a more incremental approach.

The *Migrating* topic (welc_migrating) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html, describes migrating from Version 4.x (very little to consider) or Version 3.5.x (several things to consider). It describes differences between Version 3.5.x and Version 5 brought about by the full compliance of Version 5 with Java 2 Platform, Enterprise Edition (J2EE) specifications.

3. Migrate Web server plug-ins as described in the *Preparing to install and configure a Web server* topic (tins_webserver) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.
4. Set up multiple versions of WebSphere Application Server to coexist.

   There must be no run-time conflicts for multiple instances and versions of WebSphere Application Server to run at the same time on the same machine. Potential conflicts can occur in these areas:

   - Prerequisites and corequisites
   - Operating system registration
   - Environmental conflicts
   - File system
   - Port assignments

   a. Run Version 3.5.x and Version 5 together.
   b. Run Version 4.0.x and Version 5 together.
   c. Install Version 5 more than once on the same machine.

Migration migrates all your resources and applications, but does not migrate entities in your classes directory. As the WebSphere Application Server installation wizard migrates applications to Version 5, it also migrates any applications that use IBM WebSphere Application Server Enterprise Edition, Version 4.0.x. The base migration removes (from the server configuration) those parts of your configuration that are dependent upon the following IBM WebSphere Application Server Enterprise Edition, Version 4.0.x services:

- **Business Rule Beans**
- **Internationalization**
- **Work Area Services**

Migration backs up all files in the following directories.

For Version 3.5.x

- `bin`
- `classes`
- `deployableEJBs` (Advanced Edition only)
- `deployedEJBs` (Advanced Edition only)

- `hosts`
- `properties`
- `servlets`

For Version 4.x

- `bin`
- `classes`
- `config` (Version 4.0.x Advanced Single Server Edition only)
- `installableApps`
- `installedApps`
- `installedConnectors` (Version 4.x Advanced Edition only)
- `properties`

You now understand coexisting with, or migrating the applications and configuration from, a previous version of WebSphere Application Server.

Migration from Version 4.x to Version 5 does not require extensive tuning; migration from Version 3.5.x to Version 5 does require you to examine the migrating applications. See the Migrating administrative configurations manually topic for a description of fine tuning a migration. Part of the procedure for manual migration includes a description of what to look for after using the migration tools.

Migrating to Network Deployment

Migrating WebSphere Application Server Advanced Edition to WebSphere Application Server Network Deployment requires migrating all nodes in the collection to WebSphere Application Server.

Your current configuration from a previously installed version contains one or more node configurations, one for each node in the repository. You have a choice of where to install Network Deployment. You can install it either:

- On one of the nodes where you are installing and migrating applications to WebSphere Application Server.

  This choice maps the single domain with multiple nodes to a single Version 5 cell with the same number of nodes. Network Deployment is on one of the nodes. This placement provides a configuration that is equivalent to the previously installed version.
- On an additional machine that does not have WebSphere Application Server.

  This choice maps the single domain with multiple nodes to a single Version 5 cell with the same number of nodes, but with Network Deployment on an additional node.

You can vary the order in which you migrate these nodes. This task describes the most direct method.

Note: After federating an application server node into a deployment manager cell, you cannot use the migration tools on the application server node. To use these tools again, remove the node from the cell, use the tools, and add the node to the cell again.

You can also perform a silent migration to Network Deployment during a silent installation.

1. Perform a typical or custom WebSphere Application Server installation on all nodes in the repository. The installation procedure is described in the InfoCenter for the base WebSphere Application Server product.
2. Migrate each application server node to the base WebSphere Application Server product, as described in the WebSphere Application Server InfoCenter.

   You must migrate each node that you intend to add to the Network Deployment configuration.
3. Install the Network Deployment product.
   - When installing Network Deployment on a WebSphere Application Server node that you migrated:
     – If the previous version is still installed and running, select the migration option during the installation.
     – If the previous version is no longer installed, perform a manual migration against the WASPreUpgrade `backup` directory on this or another migrated node. Issue the **WASPostUpgrade** ***backupDirectory*** command from the `bin` directory of the install_root.
   - When installing Network Deployment on a machine that did not have an earlier version of WebSphere Application Server, perform a manual migration against the WASPreUpgrade `backup` directory on another node that you migrated from the previous domain. Issue the **WASPostUpgrade** ***backupDirectory*** command from the `bin` directory of the install_root.

   You can also install the Network Deployment product silently, and migrate an earlier version configuration silently. You must supply values for silent migration options when you tailor the options response file:
   a. Direct the installation program to migrate a previous installation by specifying the -W previousVersionDetectedBean.previousVersionDetected= ″true″ parameter.
   b. Use one of the following values to identify the specific version to migrate:
      - For WebSphere Application Server Advanced Edition, Versions 3.x and 4.0.x, use one of the following parameters:
        – -W previousVersionPanelBean.selectedVersionEdition=″AE″
        – -W previousVersionPanelBean.selectedVersionEdition=″advanced″
      - For WebSphere Application Server Advanced Single Server Edition, Version 4.0.x, use the -W previousVersionPanelBean.selectedVersionEdition= ″AEs″ parameter.
      - For WebSphere Application Server Standard Edition, Version 3.x, use the -W previousVersionPanelBean.selectedVersionEdition= ″standard″ parameter.
   c. Specify the location where the previous version is installed in the -W previousVersionPanelBean.selectedVersionInstallLocation= ″/opt/WebSphere/AppServer″ parameter.
   d. Specify the path to the configuration file:
      - For WebSphere Application Server Advanced Edition, Versions 3.x and 4.0.x, use the -W previousVersionPanelBean.selectedVersionConfigFile= ″/opt/WebSphere/AppServer/config/admin.config″ parameter.
      - For WebSphere Application Server Advanced Single Server Edition, Version 4.0.x, use the -W previousVersionPanelBean.selectedVersionConfigFile= ″/opt/WebSphere/AppServer/config/server-cfg.xml″ parameter.

- For WebSphere Application Server Standard Edition, Version 3.x, use the -W previousVersionPanelBean.selectedVersionConfigFile= ″/opt/WebSphere/AppServer/config/server-cfg.xml″ parameter.

e. Specify the version number of the previous version, such as ″4.0″, ″4.0.1″, or ″3.5″, in the -W previousVersionPanelBean.previousVersionSelected= ″4.0″ parameter.

f. Indicate that you are selecting the version you have identified with the -W previousVersionPanelBean.migrationSelected= ″true″ parameter.

g. Specify the backup directory where WASPreUpgrade is to store information from the previous version with the -W migrationInformationPanelBean.migrationBackupDir= ″/tmp/migrationbackup″ parameter.

h. Specify the directory for storing migration logs with the -W migrationInformationPanelBean.migrationLogfileDir= ″/tmp/migrationlogs″ parameter.

i. Enter a node name, host name, and cell name for the new Version 5 installation. The node name is used for administration must be unique within the group of nodes that comprise the cell. The host name is the DNS name or IP address for this computer. The cell name is a logical name for the group of nodes.

   1) Use the -W nodeNameBean.nodeName=″nodenameManager″ parameter to specify a unique node name. You can use the short host name of the machine as the node name portion of the name and append Manager to it to compose the name.

   2) Use the -W nodeNameBean.cellName=″nodenameNetwork″ parameter to specify the cell name. You can use the host name as the node name portion of the name and append Network to it to compose the name.

   3) Use the -W nodeNameBean.hostName=″hostNameOrIPAddress″ parameter to specify the host name for the machine, which can be the fully qualified network name, the short name, or the IP address.

You can also configure the options response file for coexistence with an earlier version by specifying non-conflicting port assignments in the file.

4. Start the deployment manager on the Network Deployment node.

   On the system that has Network Deployment installed, start the deployment manager in either of these ways:

   - Click **Programs** > **WebSphere Application Server** > **Deployment Manager** from the Start menu on a Windows NT or Windows 2000 platform.
   - Issue the **startManager** command from the `bin` directory of the install_root.

5. Use the deployment manager administrative console to add each WebSphere Application Server node to the cell.

   The following procedure supports adding migrated nodes to a deployment manager cell, whether the nodes are cloned, standalone, or a combination of the two.

   a. Use the Network Deployment administrative console to add each node to the Network Deployment cell.

   b. Install any clustered applications on the appropriate cluster, using either the administrative console or the **wsadmin** command.

      The deployment manager automatically propagates enterprise applications to application server nodes in the cluster.

After federating an application server node into the Network Deployment cell, the configuration repository for the cell contains the master copy of the node configuration. From that point on, the deployment manager maintains the permanent configuration for the node. Changes you make with the deployment manager administrative console directly update the configuration files stored on the Network Deployment node.

6. Use First Steps to perform the installation verification test (IVT).

Occasionally, for example after rebooting an application server machine, you must restart the nodeagent server on the application server node, by running the **startNode** command from the `bin` directory of the application server install_root. To keep your application server nodes running, without having to access the `bin` directory of each one, use the operating system to monitor and restart the nodeagent process on each application server node. (You can also set up the dmgr server as a managed process on the deployment manager node.) Adding a node automatically issues the **startNode** command for the node.

Configuration mapping during migration

This topic describes the basic migration scenario, which always involves a single machine. One typical example of this is a development environment on a standalone machine. Another example is a server group node from a Version 4.0.x environment migrated to a Version 5 node that you later federate into a deployment manager cell.

The Version 5 migration tools map objects and attributes to the Version 5 environment when you restore a configuration from a previous version.

Stdin, stdout, stderr, passivation and working directories

The location for these directories is typically within the installation directory of a previous version. The default location for `stdin`, `stdout`, and `stderr` is the `logs` directory of the Version 5 install_root. The migration tools attempt to migrate existing passivation and working directories. Otherwise, appropriate Version 5 defaults are used.

Note: In a coexistence scenario, using common directories between versions can create problems.

Property files from Version 3.5.x and 4.0.x

Migration does not process property files from Version 3.5.x and 4.0.x. You must manually convert settings in these files to the Version 5 equivalent configuration.

Command line parameters

The migration tools convert appropriate command line parameters to JVM settings in the server process definition. Most settings are mapped directly. The mapping is not direct from a Version 3.5 and 4.0 server-owned process to a Version 5 process that owns multiple servers. For example, it is pointless to migrate memory heap size settings because sizes that work well in a server-owned process do not work well in a process-owned server, and vice-versa.

Bootstrap port

Migration maps a default bootstrap NameServer port setting, 900, from Version 3.5.x and 4.0.x Advanced Edition to the Version 5 NameServer default, 2809. The migration tools map a non-default value directly into the Version 5 environment.

Note: For Advanced Single Server Edition migration, the bootstrap NameServer port is mapped to the NameServer of the application server defined in the server configuration file.

Models, clones, and server groups

Version 3.5.x models and clones and Version 4.0.x server groups are redefined in Version 5 as clusters. Application servers are the only objects supported as models and cluster members in Version 5. This change is a significant difference from Version 3.5.x, in which many objects could be models and clones. All models and clones relating to application servers are mapped to clusters in Version 5.

During the migration of all other objects that you could previously clone, special mapping occurs. All clones are treated as simple objects and are mapped as if they are not cluster members. Models that are not application server models are ignored and not mapped.

Cluster members

Version 4.0.x server groups are converted to Version 5 clusters. Migration configures cluster members differently than standalone servers. When migrating a server group, all servers in the group are assigned to the HTTP transport port number of the server group, regardless of whether or not they each had a unique port number. Therefore, after migration, all the application servers in the new cluster use the same HTTP transport port. Migrated cluster members cannot run until federated into a Network Deployment cell. You can use the administrative console to assign unique port numbers, which lets you run the server without federating it.

Default Server

The name of the default server in Version 5 is **server1**. All objects previously owned by the **Default Server** of the prior version, are owned by the **server1** of Version 5 after migration.

Enterprise applications for cluster members

Migration does not deploy enterprise applications on cluster members. You must manually deploy these applications on the cluster using scripting or the deployment manager administrative console.

JDBC drivers and datasources

Version 5 significantly redefines JDBC and datasource objects. The migration tools map version 3.5.x and version 4.0.x datasources to Version 5 datasources, using predecessor settings as input variables. The datasource that is used is the WebSphere Application Server 4.x data source that uses the old ConnectionManager architecture.

Migration after federation

You can no longer migrate a previous version configuration to a Version 5 application server node, after the node is federated into a deployment manager cell. The master copy of the configuration no longer resides on the Version 5 application server node. There is a master configuration on the deployment manager node.

You can perform the migration by removing the application server node from the cell, migrating, and refederating the node to the cell. As the node is federated, the deployment manager copies the migrated configuration into the master configuration that it maintains for the application server node.

Name bindings

There is a new naming structure in Version 5. Enterprise bean references that were valid in previous versions no longer work in Version 5. However, you can use the administrative console to add a name binding that maps an old name into the new Version 5 naming structure. The name of the Version 3.5.x or 4.0.x enterprise bean reference can be both the name of the binding and the JNDI name in the Version 5 name space.

Node name

A Version 3.5.x and a Version 4.x repository can contain more than one node name and associated children. The WASPostUpgrade tool processes only those objects and children that match the node name of the node being migrated. The tool identifies node names in configuration files it is migrating, selecting any in the configuration file that match the long network name or short network name of the machine being migrated.

PageList servlet

The configuration of the PageList servlet has changed in Version 5. Direct use of the servlet has been deprecated. The PageList servlet is available as part of the servlet extension configuration in the WAR file. All references are updated to the servlet configuration supported in Version 5.

Properties directory and classes directory

Migration does not copy files from these prior version directories into the Version 5 configuration. Property files are not compatible between versions. The `classes` directory might contain incompatible files. You must migrate these files on an individual basis to move them into the Version 5 configuration.

Samples

There is no migration of samples from previous versions. There are equivalent Version 5 samples that you can use.

Security

Java 2 Security is enabled by default in Version 5. Security enablement might cause some applications to run on Version 4.0 and not run on Version 5. There are several techniques that you can use to define different levels of Java 2 Security in Version 5. One is to create a `was.policy` file as part of the application, to enable all security permissions. The migration tools call the **wsadmin** command to add an existing `was.policy` file in the Version 5 `properties` directory to enterprise applications as they are being migrated. The migration tools perform this task while moving Version 4.0 applications into Version 5.

Global security that uses Lightweight Third Party Authentication (LTPA) authentication in Versions 3.5.x and 4.0.x is migrated to the base WebSphere Application Server product and to the Network Deployment product. However, although global security was enabled in Versions 3.5.x and 4.0.x, it is disabled during migration to Version 5.

If you add this node later to an IBM WebSphere Application Server Network Deployment, Version 5 configuration, you can enable and use the LTPA configuration. Use the administrative console to generate keys for the migrated LTPA authentication mechanism. After generating the keys, you can enable global security.

Global security that uses *localos* authentication mechanisms in versions 3.5.x and 4.0.x is migrated to the Network Deployment product. However, although global security was enabled in Versions 3.5.x and 4.0.x, it is disabled during migration to Version 5. The Network Deployment product does not support the SWAM authentication mechanism. Migration sets the authentication mechanism in Version 5 to LTPA. Use the administrative console to generate keys for the migrated LTPA authentication mechanism. After generating the keys, you can enable global security.

Version 4.x introduced properties to support tuning the JNDI search timeout value along with LDAP reuse connection. These two properties are now settings in the Security Center of the Version 5 administrative console. There is no migration of Version 4.x property values to the Version 5 settings.

- The jndi.LDAP.SearchControl.TimeLimit property is equivalent to the Version 5 Search Timeout setting, which is `300` by default in Version 5.
- The jndi.LDAP.URLContextImplementation property is equivalent to the Version 5 Reuse Connection setting, which is `true` by default in Version 5.

Use the Version 5 administrative console to change these settings to match your Version 4 property values, if necessary.

Servlet package name changes

The package that contains the DefaultErrorReporter, SimpleFileServlet, and InvokerServlet servlets has changed for Version 5. In Versions 3.5.x and 4.0.x, the servlets are in the com.ibm.servlet.engine.webapp class. In Version 5, the servlets are in the com.ibm.ws.webcontainer.servlet class.

Transport ports

The migration tools migrate all ports. The tools log port conflict warnings if a port is already defined in the configuration. You must resolve port conflicts before running the servers that are in conflict, at the same time.

The default transport type of the Servlet Engine in Version 3.5.x is Open Servlet Engine (OSE). Because Version 5 no longer supports OSE transport, the migration tools map these transports to HTTP transports, using the same port assignments.

You must manually add VirtualHost alias entries for each port.

Web modules

The Version 5 level of J2EE requires Web container behavior changes in regard to setting content type. If a default servlet writer does not set the content type, not only does the Version 5 Web container no longer default to it, the Web container returns the call as "null". This might cause some browsers to display resulting Web container tags incorrectly. Migration sets the `autoResponseEncoding` IBM extension to `true` for Web modules as it migrates enterprise applications. This prevents the problem.

Version 3.5 > Version 5 migration

The migration tools assist in the transition from Version 3.5.x to Version 5, by migrating system configurations and creating J2EE artifacts, including J2EE security roles mapping. The migration tools create initial J2EE enterprise applications based on Version 3.5.x configurations. However, because of the significant change in application structures, plan to carefully test and fine tune migrated applications, using development and deployment tools, to determine exactly how the applications function in Version 5.

Analyze the `WASPostUpgrade.log` file for detailed information about migrated enterprise beans. The J2EE programming model specifies an architecture for how applications are created and deployed. Because applications in Version 3.5.x do not have the same architecture, the WASPostUpgrade tool recreates applications. It creates all migrated Web resources and enterprise beans in J2EE applications. It maps all enterprise applications from the Version 3.5.x installation into J2EE applications with the same name, deployed in the same server.

The WASPostUpgrade tool maps Web resources and enterprise beans that are not included in an enterprise application, into a default J2EE application that includes the name of the server. The tool maps Web applications to J2EE WAR files. The tool deploys enterprise beans as EJB 1.1 beans in J2EE JAR files. The tool combines resources in a J2EE EAR file and deploys it in the Version 5 configuration. There are some differences between the EJB 1.0 and EJB 1.1 Specifications, but in most cases, EJB 1.0 beans can run successfully as EJB 1.1 beans.

Version 3.5 > Version 5 mapping details

- **datasources.xml**

  You can use a Version 3.5.x `datasources.xml` file to augment datasource configuration settings. Version 3.5.x stores the file in the `properties` directory. The migration tools migrate an existing `datasources.xml` file by merging properties in the file into the datasource and JDBC driver configuration.
- **Enterprise applications**

  The Version 3.5.x enterprise-application entries are optional, they are most often used to organize sets of objects together for Security definitions. The enterprise bean and web-application portions of the enterprise-application point to their respective entries in other portions of the xml file. Each enterprise-application is processed to create a J2EE application with the same name. The enterprise bean and Web-application entries are used as pointers to the definitions of enterprise beans and Web-applications. The details of these entries are then used to build a J2EE application.

  For enterprise bean files, the JAR-file definition is used to find the JAR files to redeploy and add to the J2EE application The Web-application document-root entries are used to find the resources used within the Web-application (HTML, JSP pages, and so forth) These files are copied to the WAR file within the J2EE application. The Web-application classpath entries are used to find servlets and JAR files that are copied to the WAR file within the J2EE application.

  Enterprise applications are created during the migration from Version 3.5.x. These are created as J2EE 1.2 compatible enterprise applications and contain EJB 1.1-, Servlet 2.2- and JSP 1.1-level modules. This provides the most straight forward compatibility and enables interoperability with previous WebSphere Application Server versions.
- **Enterprise beans**

  Version 3.x supports only the EJB 1.0 Components Specification level. Version 5 supports EJB 1.1 and 2.0 components. However, many EJB 1.0 beans can successfully deploy as EJB 1.1 beans. The migration tools redeploy enterprise beans automatically as part of the application migration phase. Check the `WASPostUpgrade.log` file for deployment details of these enterprise beans. Make any necessary changes and redeploy.

No redeployment is required when moving EJB 1.1 JAR files from Version 4.

Specify only one backend datastore vendor per JAR file. If there are enterprise beans that use different backends, package them into separate JAR files.

- **J2EE security**

  The security authorization model in version 3.5.x is based on the notion of Enterprise Application and Method Groups. The cross product of the enterprise application and the method groups is a WebSphere Application Server permission. J2EE specifies an authorization model based on roles.

  To convert from the WebSphere Application Server permission model in version 3.5.x to the role based authorization model in Version 5, the migration tools create a one-to-one mapping from a WebSphere Application Server permission to a new role under that application. Therefore, for each enterprise application and each method group in Version 3.5.x, the migration tools create a role in Version 5, contained in the J2EE application deployment descriptor. The authorized subjects for each role are contained in an authorization table found in the J2EE application binding.

  J2EE specifies an authorization model based on roles. WebSphere Application Server interprets the role to mean a set of permissions to access a resource. In the case of an enterprise bean method invocation, the permission to access the method on a particular bean is specified by a method permission. This method permission is associated with one or more roles in the deployment descriptor in the bean jar file.

  In the case of accessing Web resources, the permission to access a Web URI and invoke a HTTP method on that URI is specified in terms of Web resource collections and security constraints in J2EE. The deployment descriptor of the Web application WAR file contain the security constraints and Web resource collections.

- **JSP levels**

  Version 5 runs JSP 1.0 and 1.1 objects as JSP 1.2 objects, which is the only supported level.

- **Servlet Redirector**

  Version 5 does not support the Servlet Redirector from previous versions. The migration tools ignore these objects.

- **Servlet package name changes when migrating from Version 3.5.x to Version 5**

  If the Version 3.5.x configuration defines the SimpleFileServlet servlet, the servlet is not migrated. The migration tools set the FileServingEnabled attribute in the ibm-web-ext.xmi Web module file to `true`.

  If the Version 3.5 configuration defines the InvokerServlet servlet, the servlet is not migrated. The migration tools set the ServeServletsByClassnameEnabled attribute in the ibm-web-ext.xml Web module file to `true`.

  If the Version 3.5.x configuration defines the DefaultErrorReporter servlet, the servlet is migrated into the web.xml Web module file. Migration uses the new package to set the class name.

- **Transports**

The default transport type of the Servlet Engine in Version 3.5.x is Open Servlet Engine (OSE). Because Version 5 no longer supports OSE transport, the migration tools map these transports to HTTP transports, using the same port assignments. You must manually add VirtualHost alias entries for each port.

Version 4.0 > Version 5 migration

This migration is much less complicated than moving from Version 3.5.x. The Version 4.0.x configuration is already at the J2EE 1.2 level. Although Version 5 is at the J2EE 1.3 level, J2EE 1.2 objects are supported.

- **Enterprise beans**

  No redeployment is required when moving EJB 1.1 JAR files from Version 4.0.

  Specify only one backend datastore vendor per JAR file. If there are enterprise beans that use different backends, package them into separate JAR files.

- **JMS Resources**

  All JMS resources from Version 4.0 are mapped into generic JMS resources in the Version 5 configuration. Reconfigure JMS resources that use IBM MQ Series as IBM MQ Series specific resources. MQ JMS resources have better integration with System Management. There is no need to manually define entries in the name space. You can see the backing MQ queue definitions through MQ JMS entries.

- **JSP precompiling**

  In Version 4.0.x, the classes generated from JSP pages are in a package based on the directory structure of the WAR file. Any JSP at the top of the context root is in the `unnamed` package. JSP pages in subdirectories of the root are in packages named after the subdirectories. In Version 5, the classes generated from JSP pages are all in the package `org.apache.jsp`. Therefore, the class files are not compatible between versions.

  When migrating an enterprise application from Version 4.0.x to Version 5, recompile the JSP pages to regenerate the class files into the correct packages.

  The migration tools provide this support, by using the `-preCompileJSPs` option of the **wsadmin** tool during the installation of the application.

  Use the same option to install any Version 4.0.x enterprise applications that you manually move to Version 5.

- **J2EE Security**

  You can apply security in two Version 4.x locations to enterprise applications. Information in the repository has precedence over information in the enterprise application bindings. The migration tools migrate information in the repository to the enterprise application.

- **Servlet package name changes when migrating from Version 4.0 to Version 5**

  If the Version 4.0 `web.xml` Web module file defines the SimpleFileServlet servlet, the migration tools update the class name to reflect the Version 5 package. The tools also set the FileServing Enabled attribute to `true`.

  If the Version 4.0 web.xml Web module file defines the InvokerServlet servlet, the migration tools update the class name to reflect the Version 5 package. The tools also set the ServeServletsByClassnameEnabled attribute to `true`.

If the Version 4.0 web.xml Web module file defines the DefaultErrorReporter servlet, the migration tools update the class name to reflect the Version 5 package.

Migrating administrative configurations manually

If you use a earlier version of WebSphere Application Server, the system administrator might have fine-tuned various application and server settings for your environment. It is important to have a strategy for migrating these settings with maximum efficiency and minimal loss.

You can migrate administrative configurations with the installation wizard or manually, as this task describes. If you decide to migrate manually, do not select the migration check box on the installation wizard migration panel.

You can perform incremental manual migration by calling the migration tools multiple times, each time specifying a different configuration file. There are various reasons for having multiple configuration files. Whatever the reason, migrating one configuration file at a time lets you test applications incrementally before continuing to the next configuration file.

Before using the migration tools, consult the Version 5 Release Notes document to understand what e-fixes you must apply to earlier versions. Applying fixes to an earlier version might also apply fixes to files that have a role in the migration. Apply any fixes to ensure the most effective migration of configurations and applications possible.

Note: After federating an application server node into a deployment manager cell, you cannot use the WASPreUpgrade migration tool on the application server node. To use the tool again, remove the node from the cell, use the tools, and add the node to the cell again.

Manual migration provides a more incremental migration approach than the complete migration that the installation wizard provides. IBM provides a set of migration tools for migrating administrative configurations to the base WebSphere Application Server product or to the Network Deployment product from either edition of Version 3.5.x, or from Version 4.x. The overall migration process is to back up the current configuration and necessary files, install the Version 5 product, and restore the configuration.

1. Save the current configuration using the WASPreUpgrade tool.

   The WASPreUpgrade tool provides status to the screen and to log files in the backup directory. ASCII log file names start with the text WASPreUpgrade and include a date timestamp.

   The WASPreUpgrade tool saves all files from the following directories in the existing Version 3.5.x or Version 4.x configuration to the backup directory:

   For Version 3.5.x

   - `bin`
   - `classes`
   - `deployableEJBs` (Advanced Edition only)
   - `deployedEJBs` (Advanced Edition only)
   - `hosts`
   - `properties`
   - `servlets`

For Version 4.x

- `bin`
- `classes`
- `config` (Version 4.0.x Advanced Single Server Edition only)
- `installableApps`
- `installedApps`
- `installedConnectors` (Version 4.x Advanced Edition only)
- `properties`

The WASPreUpgrade tool saves selected files from the Version 3.5.x and Version 4.x `bin` directory. It also exports the existing application server configuration from the repository. If you migrate Version 3.5.x Advanced Edition or Version 4.x Advanced Edition, ensure the administrative server of the existing environment is running.

If you migrate from Version 4.x Advanced Edition, the **WASPreUpgrade** command calls the Version 4.x **XMLConfig** command to export the existing application server configuration from the repository. If errors occur during this part of the **WASPreUpgrade** command, you might have to apply fixes to the Version 4.x installation to successfully complete the export step. See the IBM Support page for the latest fixes that might be applicable. When viewing this information from the InfoCenter, you can click **Support** to link to the IBM Support page.

2. Install the Version 5 product.

   Do not select the migration option, if it appears.

   If you select the coexistence option, the installation wizard updates the server1 configuration with coexistence port settings that you specify. WASPostUpgrade processing might change the coexisting port assignments so that there is a conflict. A conflict can cause server1 to fail to start if the previous Administrative Server or any of its associated application servers are running. After each use of WASPostUpgrade, verify Version 5 port settings in two files:

   - Verify the `BOOTSTRAP_ADDRESS` port assignment for `server1` in the `serverindex.xml` file

     If the BOOTSTRAP_ADDRESS port of the earlier version is 900, migration maps this to 2809. If the BOOTSTRAP_ADDRESS port of the earlier version is not 900, migration maps the value to server1 in a Advanced Edition migration, or to the actual server name in an Advanced Single Server Edition migration.
   - Verify the `HTTP Transport` port assignments in the `server.xml` file

     WASPostUpgrade processing adds the HTTP Transport ports from the earlier version to the Version 5 `server.xml` file. This means that server1 contains duplicate HTTP Transport port assignments, from both the coexistence panel and the previous version *Default Server*.
3. Migrate the previous configuration to the new installation with the WASPostUpgrade tool.

   The WASPostUpgrade tool migrates Version 3.5.x or Version 4.x configuration information created by the WASPreUpgrade tool, to the Version 5 installation. Because the Version 5 product adheres to the J2EE programming model and Version 3.5.x does not, significant changes are required to apply the Version 3.5.x configuration to a Version 5 installation.

The WASPostUpgrade tool does not migrate Samples because there are new ones for J2EE in Version 5. Use the new Samples instead of the ones provided with the Version 3.5.x product.

The WASPostUpgrade tool records detailed information specific to each enterprise bean it deploys, in the `WASPostUpgrade.log` file.

Stop the administrative server of the earlier version before running the Version 5 node.

4. Configure WebSphere Application Server after migration.

   Configuring WebSphere Application Server after migration is a way of verifying the results of the migration tools. You can also use the *Configuration mapping during migration* topic to verify the results of the migration. The topic has a detailed description of how the migration tools migrate objects, and what you should verify.

If you have read this topic to get an understanding of manual migration, you now have enough information to decide whether to perform manual migration, or to let the installation wizard automatically migrate your previous version of WebSphere Application Server.

Migrating from an unsupported operating system

You can migrate an earlier version of WebSphere Application Server Version 3.5.x or Version 4.0.x release that is running on an operating system that Version 5 does not support.

1. Start up the WebSphere Application Server Version 3.5.x or Version 4.0.x Administrative Server.
2. Run the **WASPreUpgrade** command line migration tool.

   There are two options. You can run the command from the `migration\bin` (or `migration/bin`) directory in the *`platform_root`* of the Version 5 CD-ROM. Or, you can copy the files in the directory on the CD-ROM to a directory you create on your hard drive.

   Identify the Version 3.5.x or 4.0.x release, and identify a backup directory where the command stores configuration files and migrating applications from the earlier version. See the *WASPreUpgrade* topic for command syntax.

   a. Run the command from the `migration\bin` (or `migration/bin`) directory in the *`platform_root`* of the Version 5 CD-ROM.

      Identify the backup directory and the location of the configuration files.

```
CD_drive:\WASPreUpgrade backupDirectory filepath\WebSphere\AppServer yourNodeName
```

      If this works, go to Step 4. If this does not work for some reason, perform steps 2B through 2F.

   b. Make a **migration** directory on your hard drive.
   c. Copy the `WASPreUpgrade.bat` (or `WASPreUpgrade.sh`) and the `setupCmdLine.bat` (or `setupCmdLine.sh`) files from the `migration\bin\` (or `migration/bin/`) directory in the *`platform_root`* of the Version 5 CD-ROM, to the directory you created on your hard drive.
   d. Edit the `setupCmdLine.bat` (or `setupCmdLine.sh`) file in your new directory.

      Change the following variables:

      - **WAS_HOME** to point to the fully qualified path to the migration directory you created
      - **JAVA_HOME** to point to the fully qualified path to your JDK\Java directory

   e. **(Optional)** Ensure that the executable bit is on for the `setupCmdLine.sh` and `WASPreUpgrade.sh` files in the `migration/bin` directory in the *`UNIX-based_platform_root`* of the Version 5 CD-ROM, if you are backing up a UNIX-based installation.
   f. Run the command from the `migration` directory you created.

      Identify the backup directory and the location of the configuration files.

      ```
      migDir\WASPreUpgrade backupDirectory filepath\
         WebSphere\AppServer yourNodeName
      ```
3. Shut down the WebSphere Application Server Version 3.5.x or Version 4.0.x release by stopping all server nodes in the configuration, including the administrative server node.
4. **(Optional)** Tar or zip the backup directory and FTP it to another system.
5. Install the new operating system, keeping the same host name.

   If possible, keep the system name and passwords the same as the old system. Place any database files related to applications you are migrating in the same path as the previous system. In general, try to keep paths the same. However, do not install Version 5 in the same directory as the previous version. If you do change paths and names, you can edit the XML configuration files to reflect the changes. Make such changes before running the **WASPostUpgrade** command below.
6. **(Optional)** FTP the backup directory from the other system and unzip it.
7. Install WebSphere Application Server, Version 5. Do not select the migration option, if it appears.
8. Run the **WASPostUpgrade** command line migration tool, from the `bin` directory of the Version 5 install_root.

   Specify the backup directory (and any non-standard configuration file name in the directory) that the **WASPreUpgrade** command created. See the *WASPostUpgrade* topic for proper command syntax.

   ```
   install_root\bin\WASPostUpgrade  WAS_HOME\migration
   ```

WASPreUpgrade command

The **WASPreUpgrade** command is a migration tool for migrating the configuration and applications of previous versions to a Version 5 application server node that is not federated in a deployment manager cell.

The command file is located in the `bin` subdirectory of the *`install_root`* directory.

Syntax

```
WASPreUpgrade backupDirectory currentWASDirectory
             [-adminNodeName administrationNodeName ]
             [-nameServiceHost host_name [-nameServicePort port_number ]]
             [-import xmiDataFile ]
             [-traceString trace_spec [-traceFile file_name ]]
```

Parameters

The first two arguments are required and positional. The third argument is required and positional only when upgrading from WebSphere Application Server, Version 3.5.x or Advanced Edition, Version 4.0.x.

Supported arguments include:

backupDirectory
   Required name of the directory in which the WASPreUpgrade tool stores the saved configuration and files, and from which the WASPostUpgrade

tool later reads the configuration and files. The WASPreUpgrade tool creates this directory if it does not already exist.

This parameter is equivalent to the -W migrationInformationPanelBean.migrationBackupDir= ″/tmp/migrationbackup″ parameter in the silent installation options response file.

currentWASDirectory

Required name of the install_root for the current Version 3.5.x or Version 4.x installation. This version can be either WebSphere Application Server Standard Edition, Version 3.5.x, WebSphere Application Server Advanced Edition, Version 3.5.x, or any form of WebSphere Application Server, Version 4.0.x.

This parameter is equivalent to the -W previousVersionPanelBean.selectedVersionInstallLocation= ″/opt/WebSphere/AppServer″ parameter in the silent installation options response file.

-adminNodeName

The name of the node containing the administrative server for the currently installed product. The value of this argument must match the node name given in the topology tree on the `Topology` tab of the administrative console for the currently installed product. The WASPreUpgrade tool calls the XMLConfig tool using this parameter. This parameter is only required when upgrading from WebSphere Application Server Standard Edition, Version 3.5.x, WebSphere Application Server Advanced Edition, Version 3.5.x, or WebSphere Application Server Advanced Edition, Version 4.0.x.

There is no equivalent parameter in the silent installation options response file.

-nameServiceHost -nameServicePort

When specified, the WASPreUpgrade tool passes these optional parameters to the XMLConfig tool. Use these parameters to override the default host name and port number used by the XMLConfig tool.

There is no equivalent parameter in the silent installation options response file.

-import

The name of the WebSphere Application Server Advanced Single Server Edition, Version 4.0 XML Metadata Interchange (XMI) configuration file to process. This parameter is optional because the program uses the `config\server-cfg.xml` file by default.

When migrating a configuration that uses anything other than the default `server-cfg.xml` file name, you must use the -import option along with a path to point to the non-default XMI configuration file. You also must use the -import and path option when running the WASPostUpgrade tool later, to point to the non-default XMI configuration file in the directory created by the WASPreUpgrade tool.

This parameter is equivalent to the -W previousVersionPanelBean.selectedVersionConfigFile= ″/opt/WebSphere/AppServer/config/server-cfg.xml″ parameter in the silent installation options response file.

-traceString -traceFile

Optional parameters to gather trace information for IBM Service personnel. Specify a trace_spec of "`*=all=enabled`" (with quotation marks) to gather all trace information.

There is no equivalent parameter in the silent installation options response file.

Logging

The WASPreUpgrade tool displays status to the screen while it is running. It also saves a more extensive set of logging information in the *`backup`* directory. You can view the `WASPreUpgrade.log` file with a text editor.

Examples

The following examples demonstrate correct syntax.

Migrating from a 3.5.x Standard Edition or Advanced Edition or from a 4.0.x Advanced Edition

The following example specifies a backup directory named backupDirectory, and identifies the install_root of the existing installation as `d:\WebSphere\AppServer`.

```
WASPreUpgrade backupDirectory d:\WebSphere\AppServer yourNodeName
```

Migrating from a Version 4.0.x Advanced Single Server Edition node with multiple backup directories

This example shows how to migrate incrementally, to migrate separate configuration files from a single node with a single installation of WebSphere Application Server Advanced Single Server Edition. To migrate more than one configuration file, you must run the WASPreUpgrade tool multiple times to multiple backup directories because not all of the applications are in the same `installedApps` directory. For this reason, using a single backup directory for all runs of the WASPreUpgrade tool is not recommended. Use a separate backup directory for each run. The intent of this example is to show how to migrate a single node with multiple configuration files.

1. Run the following WASPreUpgrade commands to migrate applications A, B, C, D, and E, which reside in three separate application directories. Server assumptions include:
   - The application server uses the default configuration file, `server-cfg.xml`, as well as `myServer1-cfg.xml` and `OldServer-cfg.xml`.

   ```
   > WASPreUpgrade "C:\WAS4ABBACKUP"  G:\WebSphere\AppServer
   > WASPreUpgrade "C:\WAS4CDBACKUP"  G:\WebSphere\AppServer
         -import  G:\WebSphere\AppServer\config\myServer1-cfg.xml
   > WASPreUpgrade "C:\WAS4EBACKUP"  G:\WebSphere\AppServer
         -import G:\WebSphere\AppServer\config\OldServer-cfg.xml
   ```
2. Run the following WASPostUpgrade commands to restore the applications and configurations to the Version 5 application server:

   ```
   > WASPostUpgrade  C:\WAS4ABBACKUP
   > WASPostUpgrade "C:\WAS4CDBACKUP" -import "C:\WAS4CDBACKUP\myServer1-cfg.xml"
   > WASPostUpgrade "C:\WAS4EBACKUP"  -import "C:\WAS4EBACKUP\OldServer-cfg.xml"
   ```

WASPostUpgrade command

The **WASPostUpgrade** command is a migration tool for migrating the configuration and applications of previous versions to a Version 5 application server node that is not federated in a deployment manager cell.

Note: If you have federated the node into a cell, invoke the removeNode command from the `bin` folder in the install_root, before using the WASPostUpgrade command line tool. After using the tool, use the deployment manager to add the node to the cell configuration again. This action synchronizes configuration changes made during the migration.

The command file is located in the `bin` subdirectory of the *`install_root`* directory.

The WASPostUpgrade tool installs all migrated applications into the *`WAS_install_root`*`/installedApps` directory for the Version 5 installation. The tool includes applications that it found in the `installedApps` directory and other directories from the earlier version.

Syntax

```
WASPostUpgrade backupDirectory [-import xmi_data_file]
                               [-cellName cell_name]
                               [-nodeName node_name]
                               [-serverName server_name]
                               [-webModuleAdditionalClasspath classpath]
                               [-documentRootLimit number]
                               [-substitute "key1=value1[;key2=value2;[...]]"]
                               [-transportSeed HTTP transport starting block]
                               [[-traceString trace_spec [-traceFile file_name]]
```

Parameters

The first argument is required. Supported arguments include:

backupDirectory
Required name of the directory in which the WASPreUpgrade tool stores the saved configuration and files, and from which the WASPostUpgrade tool reads the configuration and files. The WASPreUpgrade tool creates this directory if it does not already exist.

This parameter is equivalent to the -W migrationInformationPanelBean.migrationBackupDir = ″/tmp/migrationbackup″ parameter in the silent installation options response file.

-import
The name of the WebSphere Application Server Advanced Single Server Edition, Version 4.0 XML Metadata Interchange (XMI) configuration file to process. This parameter is optional because the program uses the `config\server-cfg.xml` file by default.

When migrating a configuration that uses anything other than the default `server-cfg.xml` file name, you must use the -import option along with the path to the non-default XMI configuration file in the directory created by the WASPreUpgrade tool.

This parameter is equivalent to the -W previousVersionPanelBean.selectedVersionConfigFile= ″/opt/WebSphere/AppServer/config/server-cfg.xml″ parameter in the silent installation options response file.

-cellName
Optional parameter to specify the cell name for the program to update. If not specified, the program inspects the configuration for cell names. When one cell name exists, the program uses it. Otherwise, the program returns an error.

This parameter is equivalent to the -W nodeNameBean.cellName=″nodenameNetwork″ parameter in the silent installation options response file.

-nodeName
Optional parameter to specify the node name for the program to update. If not specified, the program inspects the configuration for node names. When one node name exists, the program uses it. Otherwise, the program returns an error.

This parameter is equivalent to the -W nodeNameBean.nodeName=″nodenameManager″ parameter in the silent installation options response file.

-webModuleAdditionalClasspath
Optional parameter to specify the path or the path and file names of specific directories or files that you do not want copied into the Web archive (WAR) file. Instead, the program adds the paths and files to the Web Module extension (`ibm-web-ext.xmi`) `additionalClassPath` attribute. This is only applicable when migrating a Version 3.5.x installation.

There is no equivalent parameter in the silent installation options response file.

-documentRootLimit
Optional parameter to specify the number of files that the program copies from the document-root field of Web-application. It is only applicable to Version 3.5.x upgrades. If not specified, the default is 300.

There is no equivalent parameter in the silent installation options response file.

-substitute
Optional argument passed to the XMLConfig tool. Specify values for security variables to substitute (for example, `-substitute ″NODE_NAME=admin_node;APP_SERVER=default_server″`).

In the input XML data file, each key appears as *key* for substitution. This argument substitutes any occurrence of `$NODE_NAME$` with *admin_node* and `$APP_SERVER$` with *default_server* in the input XML file.

If the substitution string contains semicolons, use `$semiColon$` to separate it from the ″;″ delimiter. On UNIX platforms, add an escape character to each dollar sign ($) within the substitution string (for example, `\$semiColon\$`).

This parameter is applicable for configurations saved from Advanced Edition, Versions 3.5.x and 4.0.x.

There is no equivalent parameter in the silent installation options response file.

-transportSeed
Optional parameter used only on iSeries platforms, to specify the starting seed value to use when creating HTTP transport ports.

There is no equivalent parameter in the silent installation options response file.

-traceString -traceFile
Optional parameters to gather trace information for IBM Service personnel. Specify a trace_spec of "`*=all=enabled`" (with quotation marks) to gather all trace information.

There is no equivalent parameter in the silent installation options response file.

Logging

The WASPostUpgrade tool displays status to the screen while running. It also saves a more extensive set of logging information in the `logs` directory. You can view the `WASPostUpgrade.log` file with a text editor.

Examples

The following examples demonstrate correct syntax.

Migrating from a 3.5.x Standard Edition or Advanced Edition or from a 4.0.x Advanced Edition

The following example identifies the backup directory, named backupDirectory, that contains exported configuration files processed by the WASPreUpgrade tool.

```
WASPostUpgrade backupDirectory
```

This example shows how to migrate incrementally, to migrate separate configuration files from a single node with a single installation of WebSphere Application Server Advanced Single Server Edition. To migrate more than one configuration file, you must run the WASPreUpgrade tool multiple times to multiple backup directories because not all of the applications are in the same `installedApps` directory. For this reason, using a single backup directory for all runs of the WASPreUpgrade tool is not recommended. Use a separate backup directory for each run. The intent of this example is to show how to migrate a single node with multiple configuration files.

1. Run the following WASPreUpgrade commands to migrate applications A, B, C, D, and E, which reside in three separate application directories. Server assumptions include:
   - The application server uses the default configuration file, `server-cfg.xml`, as well as `myServer1-cfg.xml` and `OldServer-cfg.xml`.

   ```
   > WASPreUpgrade "C:\WAS4ABBACKUP"  G:\WebSphere\AppServer
   > WASPreUpgrade "C:\WAS4CDBACKUP"  G:\WebSphere\AppServer
         -import  G:\WebSphere\AppServer\config\myServer1-cfg.xml
   > WASPreUpgrade "C:\WAS4EBACKUP"  G:\WebSphere\AppServer
         -import G:\WebSphere\AppServer\config\OldServer-cfg.xml
   ```
2. Run the following WASPostUpgrade commands to restore the applications and configurations to the Version 5 application server:

   ```
   > WASPostUpgrade  C:\WAS4ABBACKUP
   > WASPostUpgrade "C:\WAS4CDBACKUP" -import "C:\WAS4CDBACKUP\myServer1-cfg.xml"
   > WASPostUpgrade "C:\WAS4EBACKUP"  -import "C:\WAS4EBACKUP\OldServer-cfg.xml"
   ```

Configuring WebSphere Application Server after migration

The installation wizard automatically configures IBM WebSphere Application Server, Version 5 and all other bundled products. There is no need for additional configuration if you do not migrate from an earlier version.

If you use the installation wizard to migrate a previous installation of WebSphere Application Server, Version 3.5.x, there are some items to review before considering your environment fully configured.

1. Check the `WASPostUpgrade.log` file for deployment details of the Version 3.5.x enterprise beans. Make any necessary changes and redeploy.

   Note: No redeployment is required when moving EJB 1.1 JAR files from Version 4.

   The J2EE programming model specifies an architecture for how applications are created and deployed. The WASPostUpgrade tool recreates applications because Version 3.5.x applications do not have the same architecture as Version 5 applications. The new Version 5 J2EE applications contain all migrated Web resources and enterprise beans. All Version 3.5.x enterprise applications become Version 5 J2EE applications with the same name, deployed in the same server.

   The WASPostUpgrade tool maps Web resources and enterprise beans that are not included in an enterprise application, into a default J2EE application that includes the name of the server. The tool maps Web applications to J2EE WAR files. The tool deploys enterprise beans as EJB 1.1 beans in J2EE JAR files. The tool combines resources in a J2EE EAR file and deploys it in the Version 5 configuration. There are some differences between the EJB 1.0 and EJB 1.1 Specifications, but in most cases, EJB 1.0 beans can run successfully as EJB 1.1 beans.

   Version 3.5.x supports only the EJB 1.0 components specification level. Version 5 supports EJB 1.1 components. However, many EJB 1.0 beans can successfully deploy as EJB 1.1 beans. The migration tools redeploy enterprise beans automatically as part of the application migration phase.

   Note: After federating an application server node into a deployment manager cell, you cannot use the migration tools on the application server node. To use these tools again, remove the node from the cell, use the tools, and add the node to the cell again.

2. Examine any Lightweight Third Party Authentication (LTPA) security settings you might have used, and apply the settings in the WebSphere Application Server security settings.

   The WASPostUpgrade tool migrates applicable security settings in the Version 3.5.x environment to J2EE security attributes.

   Global security that uses Lightweight Third Party Authentication (LTPA) authentication in Versions 3.5.x and 4.0.x is migrated to the base WebSphere Application Server product and to the Network Deployment product. However, although global security was enabled in Versions 3.5.x and 4.0.x, it is disabled during migration to Version 5.

   If you add this node later to an IBM WebSphere Application Server Network Deployment, Version 5 configuration, you can enable and use the LTPA configuration. Use the administrative console to generate keys for the migrated LTPA authentication mechanism. After generating the keys, you can enable global security.

   Global security that uses *localos* authentication mechanisms in versions 3.5.x and 4.0.x is migrated to the Network Deployment product. However, although global security was enabled in Versions 3.5.x and 4.0.x, it is disabled during migration to Version 5. The Network Deployment product does not support the

SWAM authentication mechanism. Migration sets the authentication mechanism in Version 5 to LTPA. Use the administrative console to generate keys for the migrated LTPA authentication mechanism. After generating the keys, you can enable global security.

3. Check the `WASPostUpgrade.log` file in the `logs` directory, for details about JSP 0.91 objects that the migration tools do not migrate.

   Version 5 does not support JSP 0.91 objects. The migration tools do not migrate JSP objects configured to run as JSP 0.91 objects. The migration tools do, however, recognize the objects in the output and log them. Version 5 runs JSP 1.0 and 1.1 objects as JSP 1.2 objects, which is its only supported level.
4. Review Version 3.5.x models and clones to identify non-migrated objects that you must recreate in Version 5.

   Version 3.5.x models and clones and Version 4.x server groups have been dramatically redefined in Version 5 as clusters and cluster members. Application servers are the only objects supported as models and cluster members in Version 5. This change is a significant difference from Version 3.5.x, in which many objects are models and clones. All models and clones relating to application servers are mapped to cluster members in Version 5.

   During the migration of all other objects that you could previously clone, special mapping occurs. All clones are treated as simple objects and are mapped as if they are not cluster members. Models that are not application server models are ignored and not mapped.

   Version 4.x Server Groups are converted to Version 5 clusters.
5. Identify and manually migrate non-migrated nodes in Version 3.5.x and Version 4.x repositories that have multiple nodes.

   A Version 3.5.x and a Version 4.x repository can contain more than one node name and its associated children. The WASPostUpgrade tool processes only those objects and children that match the migrating node. This determination is made by checking the names of nodes in configuration files with fully qualified and non-qualified network names of the migrating machine.
6. Examine any applications that use IBM extensions.

   In many cases, IBM provides additional features and customization options that extend the specification level even further. If your existing Version 3.x applications use IBM extensions from earlier product versions, you might need to perform mandatory or optional migration to use the same kinds of extensions in the Version 5.

   Migration from Version 4.x requires little conversion.
7. Update J2EE resources in client JAR files to the new resource format with the ClientUpgrade tool.

   J2EE applications might exist on the client, if the client has client JAR files with J2EE resources.
8. Migrate Version 3.5.x XML applications to supported XML APIs

   If your XML applications use XML for Java API, Version 2.0.x or earlier, you must migrate them to API Version 3.1 or the equivalent open-source version. Although there are inherent performance improvements in later versions of the XML for Java API, you can gain additional performance by explicitly using nonvalidating parsers in application environments where you can trust the data.

   The most significant change is that the TX-compatible APIs are no longer available. The Document API retains the XML manipulation APIs that were in TXDocument, but you must rewrite the following functionality:

- Creating and loading an XML parser: Use a Java API for XML Processing (JAXP) factory class.
- Writing out the Document Object Model (DOM) tree: Use a serializer. One drawback to the DOM Level 2 implementation in this level of the XML for Java API is that the grammar (DTD or schema) is no longer a node in the DOM tree, so you cannot write it out. As a result, only external grammars are recommended. You can query the system ID of the root element and use it to retrieve the name from the statement. After the tree is written to an XML file, you can read the file as text and insert a statement.

Note: In addition to the XML API changes, it is important to understand that J2EE Version 1.3 mandates the use of JAXP 1.1, DOM Version 2, and SAX Version 2. JAXP Version 1.2, DOM Version 3, and SAX Version 3 are not allowed in products that are compliant with the J2EE Version 1.3 specification. This prohibition exists because the newer versions were *experimental* at the time of the J2EE Version 1.3 specification. Because WebSphere Application Server is compliant with the J2EE Version 1.3 specification, WebSphere Application Server has support for JAXP Version 1.1, DOM Version 2 and SAX Version 2 only.

You must only recompile a Version 4.0.x XML application to migrate it to the Version 5 level.

You are now finished with pre-test configuration. You might have to fine tune your WebSphere Application Server environment as you test it. Test all redeployed applications before moving them into production.

Coexistence support

Coexistence, as it applies to WebSphere Application Server products, is the ability of multiple installations of WebSphere Application Server to run on the same machine at the same time. Multiple installations include multiple versions and multiple instances of one version. Coexistence also implies various combinations of Web server interaction.

The Version 5 WebSphere Application Server products can coexist with supported, previous versions as described below. The installation wizard looks for these existing installations to determine if it should prompt you for coexistence information:

- IBM WebSphere Application Server Standard Edition and Advanced Edition, Version 3.5.5 and up
- IBM IBM WebSphere Application Server, Version 5 Advanced Server Single Edition and Advanced Edition, Version 4.0.2 and later
- IBM IBM WebSphere Application Server, Version 5 Enterprise Edition, Version 4.0.2 and later

Multiversion coexistence scenarios appear in the following table. Open the InfoCenter for the Network Deployment product to see coexistence scenarios for that product.

Multiversion coexistence scenarios

Installed product	IBM WebSphere Application Server, Version 5	IBM WebSphere Application Server Network Deployment, Version 5
IBM WebSphere Application Server Standard Edition, Version 3.5.5 and later	Supported	Supported
IBM WebSphere Application Server Advanced Edition, Version 3.5.5 and later	Supported	Supported
IBM WebSphere Application Server Advanced Single Server Edition, Version 4.0.2 and later	Supported	Supported
IBM WebSphere Application Server Advanced Edition, Version 4.0.2 and later	Supported	Supported
IBM WebSphere Application Server Enterprise Edition, Version 4.0.2 and later	Supported	Supported

In addition to multiversion coexistence, WebSphere Application Server also lets you install multiple times on one machine (multiple installation instances), or install once and have multiple configurations (multiple configuration instances).

Multiple Version 5 instances on one machine include:

- Multiple application server instances from multiple installations of the base WebSphere Application Server product
- Multiple application server configuration instances from a single installation of the base product
- Multiple deployment manager instances from multiple installations of the Network Deployment product
- Multiple deployment manager configuration instances from a single installation of the Network Deployment product

Multiple WebSphere Application Server instance scenarios

Installed product	IBM WebSphere Application Server, Version 5	IBM WebSphere Application Server Network Deployment, Version 5
IBM WebSphere Application Server, Version 5	Supported, but with limitations	Supported
IBM WebSphere Application Server Network Deployment, Version 5	Supported	Supported, but with limitations

Setting up Version 3.5.x and Version 5 coexistence

You must migrate prerequisite and corequisite programs to the levels required by WebSphere Application Server, Version 5. You must also identify ports in use in Version 3.5.x before you begin the Version 5 installation, to avoid possible conflicts during coexistence. The first two steps in this task describe these activities.

You can install WebSphere Application Server Version 3.5.5 (and later) and Version 5 on the same node. When the Version 5 installation wizard detects the Version 3.5.x installation, it displays the migration and coexistence panel, where you can select either option, or neither option. If you select coexistence, the installation wizard displays the port number panel, to ensure you install Version 5 without port conflicts.

Silent installation also supports configuring for coexistence silently. You can specify non-conflicting port assignments in the options response file.

By default, there are port conflicts between Version 3.5.x and Version 5 that you must resolve. Also, if you migrate more than two Version 3.5.5 nodes to Version 5, there are port conflicts that you must resolve, as described in the *Setting up Version 5 coexistence* topic.

1. Migrate prerequisite and corequisite programs to the levels required by WebSphere Application Server, Version 5.

   Review the prerequisites on the IBM WebSphere Application Server supported hardware, software, and APIs Web site at http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html.
2. Resolve port conflicts.

   Refer to the Port number settings in WebSphere Application Server versions topic to see a list of default port numbers, and where they are defined.

   Inspect the configuration of the Version 3.5.x product, either WebSphere Application Server Advanced Edition, or WebSphere Application Server Standard Edition.

   If necessary, use the following command to examine existing node and port settings when the administrative server is running.

   ```
   xmlConfig -export config.xml -nodeName theNodeName
   ```

   Review the output xml file to look for <node action="update" name="*nodename*"> to determine the node name. You can also find port number assignments in the output.xml file. The installation wizard displays a default set of coexistence port numbers as suggested Version 5 port numbers. Change the values to ports that are not in use. The installation wizard uses whatever values you approve.

   Change conflicting HTTP transport ports manually, if necessary.
3. Associate a Web server with each WebSphere Application Server:
   - Use a separate Web server for each WebSphere Application Server.
     a. Create a Web server instance using the Web server documentation.
     b. Select the appropriate Web server plug-in feature during WebSphere Application Server installation, and identify the Web server configuration file location. (For example, identify the location of the `httpd.conf` file for IBM HTTP Server.)
   - Use the same Web server for both WebSphere Application Server versions.

     Upgrade the Web server to the common level supported by both versions of the application server.

     Prepare the same Web server to support both application server versions:
     – Migrate IBM HTTP Server (tins_websIHS in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html).
     – Migrate iPlanet (tins_websiPl).

   - Migrate Lotus Domino (tins_websDom).
   - Migrate Microsoft IIS (tins_websIIS).
4. **(Optional)** Fix any problems with environmental variables on Windows NT or Windows 2000 platforms.

   For example, installing WebSphere Application Server, Version 5 updates the system variable PATH, potentially affecting tools with the same name across installations. To run tools with conflicting names, alter the PATH environment variable in a command window and place the directory for the former installation before the directory for the latter installation. For example, PATH=E:\WebSphere\AppServer\35\bin;%PATH%. Then, invoke the tools from the `bin` directory.

Setting up Version 4.0.x and Version 5 coexistence

You must migrate prerequisite and corequisite programs to the levels required by WebSphere Application Server, Version 5. You must also identify ports in use in Version 4.0.x before you begin the Version 5 installation, to avoid possible conflicts during coexistence. The first two steps in this task describe these activities.

You can install WebSphere Application Server Version 4.0.x and Version 5 on the same node. When the Version 5 installation wizard detects the Version 4.0.x installation, it displays the migration and coexistence panel, where you can select either option, both options, or neither option. If you select coexistence, the installation wizard displays the coexistence panel to ensure you install Version 5 without port conflicts.

Silent installation also supports configuring for coexistence silently. You can specify non-conflicting port assignments in the options response file.

By default, there are port conflicts between Version 4.0.x and Version 5 that you must resolve. Also, if you migrate more than two Version 4.0.x nodes to Version 5, there are port conflicts that you must resolve, as described in the Setting up Version 5 coexistence topic.

1. Migrate prerequisite and corequisite programs to the levels required by WebSphere Application Server, Version 5.

   Review the prerequisites on the IBM WebSphere Application Server supported hardware, software, and APIs Web site at http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html.
2. Resolve port conflicts.

   Refer to the Port number settings in WebSphere Application Server versions topic to see a list of default port numbers, and where they are defined.

   Inspect the configuration of the previous version:
   - **For WebSphere Application Server Advanced Single Server Edition, Version 4.0.x:** Inspect the `server-cfg.xml` file to get port values for the configuration.
   - **For WebSphere Application Server Advanced Edition, Version 4.0.x:** Inspect the admin.config file to get port values for the configuration. When the administrative server is running, use this command:

     ```
     xmlConfig -export config.xml -nodeName theNodeName
     ```

     Review the `config.xml` file to look for <node action="update" name="*nodename*"> to find the appropriate node and port number assignments in the file. The installation wizard displays a default set of coexistence port numbers as suggested Version 5 port numbers. Change the values to ports that are not in use. The installation wizard uses whatever values you approve.

Change conflicting HTTP transport ports manually, if necessary.

3. Associate a Web server with each WebSphere Application Server.
   - Use a separate Web server for each WebSphere Application Server.
     a. Create a Web server instance using the Web server documentation.
     b. Select the appropriate Web Server plug-in feature during WebSphere Application Server installation, and identify the Web server configuration file location. (For example, identify the location of the `httpd.conf` file for IBM HTTP Server.)
   - Use the same Web server for both WebSphere Application Server versions.

     To use the same Web server for both application server versions, you must first upgrade the Web server to the common level supported by both versions of the application server.

     Follow this procedure to use the same Web server for both WebSphere Application Server versions.

   a. Select the appropriate Web Server plug-in feature during WebSphere Application Server installation, and identify the Web server configuration file location. (For example, identify the location of the `httpd.conf` file for the IBM HTTP Server that is associated with Version 4.0.x.)
   b. Edit the Web server configuration file to remove entries for Version 4.0.x as described in the *Migrating plug-ins, one machine at a time* topic (tins_websmig2) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

      The WebSphere Application Server, Version 5 plug-in acts as the routing agent to route requests to both versions.
   c. Generate the plug-in configuration files for both versions of the application server.
   d. Edit the Version 4.0.x `plugin-cfg.xml` file and the Version 5 `plugin-cfg.xml` file, to merge their entries into a combined file of all Web context roots.

      Web context roots must be unique across application server versions. If there is a common context root across versions, requests are served by the plug-in instance that was last loaded.

      If you have both Version 4.0.x Samples and Version 5 Samples in the file, you can access only the Version 5 Samples after merging the configuration files because there is one context root for Samples. This is described in the *Preparing to install and configure a Web server* topic (tins_webserver) topic in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

      Whenever you modify the `plugin-cfg.xml` file for either version, you must manually merge the files again, to make a master file.
   e. Replace the original `plugin-cfg.xml` file of the Version 5 installation with the master file.
4. **(Optional)** Fix any problems with environmental variables on Windows NT or Windows 2000 platforms.

   For example, installing WebSphere Application Server, Version 5 updates the system variable PATH, potentially affecting tools with the same name across installations. To run tools with conflicting names, alter the PATH environment variable in a command window and place the directory for the former installation before the directory for the latter installation. For example, PATH=E:\WebSphere\AppServer\40\bin;%PATH%. Then, invoke the tools from the `bin` directory.

Setting up Version 5 coexistence

After installing the base WebSphere Application Server or the Network Deployment product, you can install either one again on the same machine.

Select the new installation option from the installation wizard panel, to install a new instance instead of adding features to the last installation, and to install into a separate installation directory. Select the coexistence option to provide non-conflicting ports.

Each installation of the base product is a standalone application server (server1) with its own set of unique configuration files.

Each installation of the Network Deployment product is a standalone deployment manager (dmgr) with its own set of unique configuration files for its cell.

Be aware of multiple instance limitations, between:

- The base WebSphere Application Server product and the Network Deployment product.

  The deployment manager and the application server have no port conflicts in the default configuration settings for both products. There are no port conflicts if you install a total of two instances, one base WebSphere Application Server product and one Network Deployment product.

  If you install more than two instances, the third and subsequent installations require that you change all port numbers on the coexistence panel, to avoid potential conflicts.
- The base WebSphere Application Server product and another installation of the base WebSphere Application Server product, or the Network Deployment product and another installation of the Network Deployment product.

  You can install another instance of either product, by selecting to do so at the prompt, and by changing the installation directory.

  The most recent installation is the only one in the operating system registry.

  If you install more than two total instances of either product (more than one of each, or two of either and none of the other), the third and subsequent installations require that you change all port numbers on the coexistence panel to avoid potential conflicts.

  To uninstall a product instance, always use the operating system remove program function, such as the Add/Remove program on Windows 2000. To uninstall an unregistered instance, use the **Uninstall.exe** or **uninstall** command (or the **uninstall.sh** command on Linux/390 platforms) in the `_uninstall` directory of the *`install_root`* that matches the instance you intend to remove.

Reasons to use multiple installation instances include:

- You can achieve complete isolation between each WebSphere Application Server instance. You can uninstall one instance independently of the others.
- You can federate each installation instance of the base WebSphere Application Server product to any deployment manager cell.

- You can install the base WebSphere Application Server more than once on the same machine.
- You can install the Network Deployment product more than once on the same machine.

Reasons to not use multiple installation instances include:
- The machine might have a hard disk space constraint.
- You cannot use the operating system registry to locate any but the last installed instance of a WebSphere Application Server product.

  When you install any product a second time, the last installation is the one that appears in the registry.
- Uninstalling the last instance removes any record of the product in the registry.

  Suppose you have installed three instances of the base WebSphere Application Server product. Further suppose that you use the operating system remove program function to uninstall the registered third copy of the base product. There is no longer a registry record to indicate the existence of the other two installation instances. Other applications cannot use a query of the operating system registry to detect the presence of either base WebSphere Application Server product instance.

The *Creating multiple Version 5 configuration instances* topic describes installing the base WebSphere Application Server product or the Network Deployment product once and creating multiple configuration instances.

Use the following procedure to install multiple installation instances.
1. Use the installation wizard for multiple installations.

   If you intend to share a single Web server among installations, install a Web server plug-in feature during only one installation, as described in the next step.
2. Share a Web server among multiple installation instances.
   a. Select a Web server plug-in feature only during one installation.
   b. Generate the plug-in configuration files for every installation instance.
   c. Edit the `plugin-cfg.xml` configuration files to merge them into one master configuration.
   d. Replace the original `plugin-cfg.xml` file with the master file, on the application server where you selected the Web server plug-in feature.

   You can access samples from only one of the installation instances.
3. **(Optional)** Federate multiple installation instances into a deployment manager cell.
4. **(Optional)** Create additional servers in a multiple instance or coexistence environment.
5. Change port assignments in configuration files if you have a node that you cannot start because of port conflicts.

Federating multiple Version 5 installation instances

There are two ways to federate a base WebSphere Application Server node into a deployment manager cell:
- Using the deployment manager administrative console, as described in the trun_svr_conf_nodes article in the InfoCenter

- Using the addNode command line script from the `bin` directory of the node you are federating, as described in the rxml_addnode article in the InfoCenter

Always use the includeapps parameter of the **addNode** command unless you have previously federated the applications on the base WebSphere Application Server node into the cell. See the description of the addNode command for more information about its parameters. The administrative console also provides an options for including applications on the base node into the cell.

Federating from the deployment manager administrative console requires a running base WebSphere Application Server server.

Consider a scenario where you install both the base WebSphere Application Server product and the Network Deployment product on the same development machine. Installing a production application server on the same machine as the deployment manager is not recommended unless the machine has the capacity to handle both jobs. To let both products run at the same time, you must install the base WebSphere Application Server product using the coexistence panel, to select ports that do not conflict with the deployment manager ports. After installation, stop the application server if it is running. Start the deployment manager server. Issue the **addNode** command line script from the `bin` directory of the base node. The deployment manager federates the base node and instantiates the node agent for the base node.

A coexistence environment might have multiple base WebSphere Application Server product installations on one machine. You can federate each installation into the same cell, or into different cells. Whenever the deployment manager federates a node into its cell, it configures a node agent for the node with a set of default ports. If the base node has the embedded messaging feature, the deployment manager also configures a WebSphere Application Server JMS provider, jmsserver, which is a server with another set of default ports.

Plan to configure the node agent and the JMS provider with port assignments that differ from the defaults, to ensure there is no conflict with coexisting installation instances that have similar configurations.

1. Use the **stopNode** command line script to stop any running node agents that are using default ports on the same physical machine.

   Issue the command from the `bin` directory of the application server install_root.
2. Federate the application server node into the deployment manager cell, using the **addNode** command line script from the `bin` directory of the node to federate. This command automatically starts the node agent, with default ports.

   For example, suppose you federate an installation instance that has a node name of ***MyNode***.
3. Log on to the deployment manager administrative console. Go to **System Administration > Node Agents > nodeagent > End Points**. Change port numbers for all end points and save the changes.
4. Log on to the deployment manager administrative console, if you have installed the embedded messaging feature on the node you federated:
   a. Go to **System Administration > Node Agents > jmsserver > Security Port Endpoints**. Change port numbers for all end points and save the changes.
   b. Go to **System Administration > Node Agents > jmsserver > End Points**. Change port numbers for all the end points and save the changes.
5. Synchronize the changes using **System Administration > Nodes**. Select the node **MyNode** and click **Synchronize**.

6. Stop the node agent on the application server node using the **stopNode** command line script. Restart the node agent using the **startNode** command.
7. Change port assignments in the configuration files if you have an application server or deployment manager node that you cannot start because of port conflicts.

Port number settings in WebSphere Application Server versions

This topic provides reference information about identifying port numbers in versions of WebSphere Application Server, as a means of determining port conflicts that might occur when you intend for an earlier version to coexist or interoperate with Version 5.

Version 3.5.x port numbers

Resolve port conflicts by inspecting the configuration of the Version 3.5.x product, either WebSphere Application Server Advanced Edition or WebSphere Application Server Standard Edition. When the administrative server is running, use this command to examine current node and port settings:

```
xmlConfig -export config.xml -nodeName theNodeName
```

Look for the OSE Remote port assignments.

WebSphere Application Server Version 3.5.x default port definitions in the admin.config file

Port name	Value	
	Standard Edition	Advanced Edition
lsdport (Location Service Daemon)	9000	9000
bootstrapPort	900	900
LSDSSL (With security on)	9001	9001

Version 4.0.x port numbers

For WebSphere Application Server Advanced Single Server Edition, Version 4.0.x: Inspect the `server-cfg.xml` file to find the Web container HTTP transports port values for the configuration.

For WebSphere Application Server Advanced Edition, Version 4.0.x: When the administrative server is running, use this command to extract the configuration from the database:

```
xmlConfig -export config.xml -nodeName theNodeName
```

Look for the Web container HTTP transports port assignments.

WebSphere Application Server Version 4.0.x port definitions

Port name	Value	File		
		Advanced Edition	Enterprise Edition	Advanced Single Server Edition
bootstrapPort	900	admin.config	admin.config	server-cfg.xml
lsdPort	9000			
LSDSSLPort	9001			
HTTP transport port	9080	database	database	
HTTPS transport port	9443			
Admin Console HTTP transport port	9090			
ObjectLevelTrace	2102			
diagThreadPort	7000			

Version 5 port numbers

You can use the administrative console to configure the Version 5 application server, to resolve conflicting ports.

WebSphere Application Server Version 5 default port definitions

Port name (InfoCenter reference)	WebSphere Application Server	Network Deployment	File
	Value		
HTTP_Transport (crun_transport)	9080	Not applicable	• server.xml • virtualhosts.xml
HTTPS Transport Port / HTTPS_Transport (crun_transport)	9443	Not applicable	
HTTP Admin Console Port / HTTP_Transport_Admin (crun_transport)	9090	9090	
HTTPS Admin Console Secure Port / HTTPS_Transport_Admin (crun_transport)	9043	9043	
Internal JMS Server / JMSSERVER_Security_Port (cm_secty)	5557	Not applicable	server.xml
JMSSERVER_Queued_Address (umj_ptcfw)	5558	Not applicable	serverindex.xml
JMSSERVER_Direct_Address (umj_ptcfw)	5559	Not applicable	
BOOTSTRAP_Address (tnam_develop_naming)	2809	9809	
SOAP_Connector-Address (cxml_connector)	8880	8879	
DRS_Client_Address (tprf_dynamiccache)	7873	7989	
SAS_SSL_ServerAuth_Listener_Address (tsec_inboundtransport)	0	9401	
CSIV2_SSL_ServerAuth_Listener_Address (tsec_inboundtransport)	0	9403	
CSIV2_SSL_MutualAuth_Listener_Address (tsec_inboundtransport)	0	9402	
IBM HTTP Server Port (trun_plugin_vhost)	80	Not applicable	• virtualhosts.xml • plugin-cfg.xml

Port name (InfoCenter reference)	WebSphere Application Server	Network Deployment	File
	Value		
IBM HTTPS Server Admin Port	8008	Not applicable	plugin-cfg.xml
CELL_Discovery_Address (trun_watch_cell)	Not applicable	7277	serverindex.xml
ORB_Listener_Address (tsec_inboundtransport)	Not applicable	9100	
CELL_Multicast_Discovery_Address (trun_watch_cell)	Not applicable	7272	

When you federate an application server node into a deployment manager cell, the deployment manager instantiates a nodeagent on the application server node. The node agent uses these port assignments by default:

WebSphere Application Server Version 5 nodeagent default port definitions

Port name (InfoCenter reference)	Value	File
BOOTSTRAP_Address (tnam_develop_naming)	2809	serverindex.xml
ORB_Listener_Address (tsec_inboundtransport)	9900	
SAS_SSL_ServerAuth_Listener_Address (tsec_inboundtransport)	9901	
CSIV2_SSL_MutualAuth_Listener_Address (tsec_inboundtransport)	9101	
CSIV2_SSL_ServerAuth_Listener_Address (tsec_inboundtransport)	9201	
NODE_Discovery_Address (trun_svr_conf_nodes)	7272	
NODE_Multicast_Discovery_Address (trun_svr_conf_nodes)	5000	
DRS_Client_Address (tprf_dynamiccache)	7888	
SOAP_Connector-Address (cxml_connector)	8878	

When you federate an application server node with the embedded messaging server feature into a deployment manager cell, the deployment manager instantiates a JMS server process, jmsserver, on the application server node. The JMS server uses these port assignments by default:

WebSphere Application Server Version 5 JMS server default port definitions

Port name (InfoCenter reference)	Value	File
JMSSERVER_Direct_Address (umj_ptcfw)	5559	serverindex.xml
JMSSERVER_Queued_Address (umj_ptcfw)	5558	
SOAP_Connector-Address (cxml_connector)	8876	
JMSSERVER_Security_Port (cm_secty)	5557	server.xml

Automatically restarting server processes

There are several server processes that the operating system can automatically restart when the server processes stop abnormally. To set up this function on a UNIX-based operating system, you must have root authority to edit the inittab. On a Windows NT or Windows 2000 operating system, you must belong to the Administrator group and have the advanced user rights *Act as part of the operating system* and *Log on a service* to define Windows services. The installation wizard grants you the advanced user rights, if your user ID is part of the Administrator group. It displays a message that states that your user ID will have the rights the next time you log on.

There are various environments where you might use this function of automatically restarting servers. You can restart the **server1** server process in a standalone WebSphere Application Server environment, for example. Here is a list of processes you might consider restarting:

- The **server1** process on a standalone application server
- The **dmgr** process on a deployment manager node
- The **nodeagent** server process on any application server node in a deployment manager cell
- The **jmsserver** process on the application server node, if the application server node has the embedded messaging feature
- The **IBM HTTP Server** process
- The **IBM HTTP Administration** process
- The **WebSphere Embedded Messaging Publish And SubscribeWAS_***node_name***_jmsserver** process, if the application server node has the embedded messaging feature
- The **WebSphere Embedded Messaging Publish And SubscribeWAS_***node_name***_server1** process, if the application server node has the embedded messaging feature

You can create Windows services during installation, using the installation wizard. The wizard lets you create services for these servers:

- The **server1** process in a standalone base product environment, defined as a manually started (versus automatic) service
- The **IBM HTTP Server** process and the **IBM HTTP Administration** process, defined as automatically started services if you have chosen to install the IBM HTTP Server feature during the base product installation
- The **dmgr** process on a deployment manager node, defined as a manually started (versus automatic) service

The installation wizard does not provide a way to create a service for a node agent because the deployment manager instantiates each node agent after installation when you add an application server node to the deployment manager cell. For this reason, you must manually create a function that automatically starts a failed node agent server process.

You must manually create a shell script that automatically starts any of the processes previously mentioned, on a UNIX-based operating system. Each Windows service or UNIX shell script controls a single process, such as a standalone WebSphere Application Server instance. Multiple standalone application server processes require multiple Windows services or UNIX scripts, which you can define.

In a Network Deployment environment, the **addNode** or **startNode** command starts a single unmonitored node agent process only, and does not start all of the processes that you might define on the node. While running, the node agent monitors and restarts application server processes on that node, on either a Windows or a UNIX-based platform. Each application server process has MonitoringPolicy configuration settings that the node agent uses when monitoring and restarting the process.

It is recommended that you set up a monitored nodeagent server process manually, either through a Windows service, or through the rc.was example shell script on UNIX-based platforms. The operating system monitors and automatically restarts the node agent server, nodeagent, if the process terminates abnormally, which means that it stops without going through a normal shutdown. It is also recommended that you set up the deployment manager server, dmgr, as a monitored process. As mentioned, you can do this during installation on a Windows NT or Windows 2000 platform. On a UNIX-based platform, use the `rc.was` example shell script to set up the deployment manager dmgr server as a monitored process.

If you do not install the WebSphere Application Server base product or the WebSphere Application Server Network Deployment product as a Windows service during installation, you can use the **WASService.exe** command line tool, in the *install_root*`/bin` directory, to do so at a later time. You can use the tool to add any WebSphere Application Server process as a Windows service. The operating system can then monitor each server process, and restart the server if it stops.

1. **(Optional) Use the installation wizard** to set up a Windows service to automatically monitor and restart processes related to the WebSphere Application Server product.
   - Perform the following procedure from the installation wizard for the Network Deployment product:
     a. Click **Run WebSphere Application Server Network Deployment as a service**.
     b. Enter your user ID and password and click **Next**.

     The installation wizard creates the following service during the installation:

     ```
     WebSphere Application Server V5 - dmgr
     ```

     The WebSphere Application Server V5 - dmgr service controls the **dmgr** server process, which is the deployment manager server that administers the cell.
2. **(Optional) After installing**, use the WASService.exe tool to manually define the **nodeagent** server process as a Windows NT or Windows 2000 service.

   You can use the same tool to manually define a Windows service for another installation or configuration instance of either the base WebSphere Application Server product or the Network Deployment product.
3. **(Optional) After installing**, set up a UNIX-based shell script to automatically monitor and restart the nodeagent server or any other related server process.
   a. Locate the `rc.was` example shell script, which is in the *install_root*`/bin` directory.
   b. Create a new shell script for each process that the operating system is to monitor and restart.
   c. Edit each shell script according to comments in its header, which provide instructions for identifying a WebSphere Application Server process.

d. Edit the inittab table of the operating system, to add an entry for each shell script you have created.

   Comments in the header of the rc.was script show a sample inittab entry line for adding the script. This inittab entry causes the UNIX-based system to call each shell script whenever the system initializes. As it runs, each shell script monitors and starts the server process you specified.

Similar to a Windows service, each shell script monitors and restarts an individual WebSphere Application Server server process in a standalone environment, or a node agent or deployment manager process in an WebSphere Application Server Network Deployment environment.

You can also use the **Start the Server** and **Stop the Server** commands to control the IBM WebSphere Application Server on a Windows NT or Windows 2000 system. Access these commands from the Start menu, clicking **Start > Programs > IBM WebSphere > Application Server V5.0**.

You can also use the **Start the Manager** and **Stop the Manager** commands to control the Network Deployment dmgr server on a Windows NT or Windows 2000 system. Access these commands from the Start menu, clicking **Start > Programs > IBM WebSphere > Deployment Manager V5.0**.

Note:

Processes started by a **startServer** (or **startNode** or **startManager**) command are not running as monitored processes, regardless of whether you have configured them to be.

For example, you can configure a base application server as a WebSphere Application Server Windows service. However, if you start the application server instance using the **startServer** command, the Windows system does not monitor or restart the application server because it was not started as a Windows service. The same is true on UNIX-based platforms. You must start the server process with a shell script based on the example rc.was script, to have the server running as a monitored process.

Creating multiple Version 5 configuration instances

Multiple configuration instances, as it applies to WebSphere Application Server products, is the ability to install the base WebSphere Application Server product or the Network Deployment product once, and to use the **wsinstance** command to create multiple ***instances of the initial installation***, all running on the same machine at the same time.

In contrast to multiple configuration instances is *coexistence*, which refers to multiple *installations* of WebSphere Application Server, running on the same machine at the same time. Both coexistence and multiple configuration instances of the base WebSphere Application Server product imply various combinations of Web server interaction.

You can find a description of coexistence in the Coexistence support topic.

This section contains the following topics:
- Procedure for creating configuration instances
- The wsinstance command

Procedure for creating configuration instances

You can install the base WebSphere Application Server or the Network Deployment product one time and use the wsinstance tool in the `install_root\bin\wsinstance` folder to create multiple configuration instances. Each WebSphere Application Server configuration instance is a standalone instance with a unique name, and its own set of configuration files and user data folders. Configuration folders include `config`, `etc` and `properties`. User data folders include `installedApps`, `installableApps`, `temp`, `logs`, `tranlog` and `wstemp`. It also has the Administration application to manage the configuration instance. It shares all run-time scripts, libraries, the Software Development Kit, and other files with the initial application server.

Each configuration instance of the Network Deployment product is a standalone deployment manager (dmgr) with its own set of unique configuration files and user data folders for the cell. Configuration folders include `config`, `etc` and `properties`. User data folders include `installedApps`, `installableApps`, `temp`, `logs`, `tranlog` and `wstemp`. The dmgr configuration instance also includes the Administration and File Transfer applications to manage the configuration instance.

You can configure and operate each configuration instance independently of other instances.

These limitations apply to multiple configuration instances:

- You cannot federate a configuration instance of the base product into a deployment manager cell. A deployment manager cannot manage a configuration instance.
- You cannot use the **Uninstall.exe** or **uninstall** command (or the **uninstall.sh** command on Linux/390 platforms) to remove a configuration instance. You must remove configuration instances with the -delete option of the **wsinstance** command, before uninstalling the installation instance.
- You cannot migrate the configuration and applications from an earlier release to a Version 5 configuration instance.

Reasons to use configuration instances include:

- You install the product only once.
- You can have standalone environments when installing the full product multiple times is not an acceptable solution because of space or procedural constraints.
- You can have a centralized development environment that is easy to maintain, where multiple developers and testers share one machine.
- You can create identical environments, where everyone is developing, testing, or performing in an identical workspace.

One reason to not use configuration instances is that you cannot federate a configuration instance of the base product into a deployment manager cell. A deployment manager cannot manage a configuration instance.

Use the following procedure to create and configure multiple Version 5 instances:

1. Install the Network Deployment product.
2. Use the **wsinstance** command in the `WAS_HOME\bin\wsinstance` folder, to create a configuration instance of the server you installed in step 1.

   Refer to the description of the **wsinstance** command to learn more about the command, and to see examples of use.

You must specify a unique directory path and a unique node name for each configuration instance.

You must also specify unique port numbers for each configuration instance. For example, on a Windows 2000 platform, specify ports beginning at `20002`, for node shasti, in configuration instance_root `G:\shasti\WebSphere`, on the planetjava machine, by issuing this command:

```
wsinstance.bat -name shasti -path G:\shasti\WebSphere
             -host planetjava  -startingPort 20002 -create
```

This command creates a separate set of configuration and other data files.

3. Run the **setupCmdLine.bat** (or **setupCmdLine.sh**) script in the `bin` directory of the *`instance_root`* folder to set the WebSphere Application Server environment to the configuration instance.

   After completing this step, you can start the application server configuration instance with the **startManager** or **startServer** command.
4. **(Optional)** Federate multiple installation instances into a deployment manager cell.
5. **(Optional)** Create additional servers in a multiple instance or coexistence environment.
6. Change port assignments in configuration files if you have a node that you cannot start because of port conflicts.

The wsinstance command

The **wsinstance.bat** (or **wsinstance.sh** for UNIX-based platforms) command line tool creates multiple configuration instances of one initial installation of either the base WebSphere Application Server or the deployment manager.

Each WebSphere Application Server configuration instance is a standalone instance with a unique name, and its own set of configuration files and user data folders. Configuration folders include `config`, `etc` and `properties`. User data folders include `installedApps`, `installableApps`, `temp`, `logs`, `tranlog` and `wstemp`. It also has the Administration application to manage the configuration instance. It shares all run-time scripts, libraries, the Software Development Kit, and other files with the initial application server.

Each deployment manager configuration instance is a different cell. You can federate application server installation instances into a deployment manager configuration instance, but you cannot federate application server configuration instances into the cell.

Each configuration instance of the Network Deployment product is a standalone deployment manager (dmgr) with its own set of unique configuration files and user data folders for the cell. Configuration folders include `config`, `etc` and `properties`. User data folders include `installedApps`, `installableApps`, `temp`, `logs`, `tranlog` and `wstemp`. The dmgr configuration instance also includes the Administration and File Transfer applications to manage the configuration instance.

The **wsinstance.bat** command file is located in the `wsinstance` subdirectory of the `bin` directory in the *`install_root`* of the product.

Syntax

Windows NT and Windows 2000:

```
wsinstance.bat  -name instanceName
                -path instanceLocation
                -host hostName
               [-startingPort startingPort]
                -create|-delete
                -debug
```

UNIX-based operating platforms:

```
wsinstance.sh  -name instanceName
               -path instanceLocation
               -host hostName
              [-startingPort startingPort]
               -create|-delete
               -debug
```

Parameters

Supported arguments include:

-name Specifies the instance name of the new configuration instance. Ensure this value is unique. The **wsinstance** command uses this name to construct the node name, which is *instanceName_hostname* for the new configuration instance of a deployment manager. The instance name is also the name of the cell.

-path Specifies the file path of the instance. All required folders for the instance are in this directory, which is unique to the configuration instance.

-host Specifies the hostname, which is the name of the host on which you are creating the configuration instance. This should match the host name you specified during installation of the initial product.

-create|-delete
Specifies whether to create or delete the configuration instance.

-startingPort
Optional parameter. Specifies the starting port number for generating all ports for the configuration instance. If not specified, the **wsinstance** command uses default ports, or custom-defined ports from a file that you can create.

Example of startingPort parameter use

The **wsinstance** command generates an `*instanceName*_portdef.props` file in the `wsinstance` subdirectory of the `bin` directory in the `*install_root*` folder. The **wsinstance** command assigns port numbers in the file to the configuration instance as it creates the instance.

If you do not use the -startingPort parameter the first time you create a configuration instance, the **wsinstance** command adds one (1) to the default port numbers for the base WebSphere Application Server product. If you create two configuration instances without using the -startingPort parameter, both instances have the same, conflicting port numbers.

You can create the `*instanceName*_portdef.props` file manually with predefined ports. You do not have to specify the -startingPort parameter again. The **wsinstance** command reads an existing `*instanceName*_portdef.props` file, to use the port numbers specified in the file. This command lets you manually create the file and specify the port numbers, before creating the configuration instance.

Use the template file, `portdef.props`, to create a new *`instanceName`*`_portdef.props` file, before creating the new configuration instance.

The following example `shasti_portdef.props` file is created with this command:

```
wsinstance.bat -name shasti -path G:\shasti\WebSphere
               -host planetnt -startingPort 20002 -create
```

```
HTTPS_TRANSPORT_ADMIN=20002
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=20004
HTTP_TRANSPORT_ADMIN=20003
HTTP_TRANSPORT=20005
HTTPS_TRANSPORT=20006
INTERNAL_JMS_SERVER=20007
BOOTSTRAP_ADDRESS=20008
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=20009
DRS_CLIENT_ADDRESS=20011
SOAP_CONNECTOR_ADDRESS=20010
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=20012
JMSSERVER_QUEUED_ADDRESS=20013
JMSSERVER_DIRECT_ADDRESS=20014
```

Examples for Windows NT and Windows 2000 systems

An example of creating a configuration instance for user `shasti` follows:

```
wsinstance.bat -name shasti -path G:\shasti\WebSphere -host planetnt -create
```

On a base WebSphere Application Server product installation, the command creates an application server configuration instance, named `shasti`, with a node name of `shasti_planetnt`.

On a Network Deployment product installation, the command creates a deployment manager configuration instance named `shasti`, in a cell named `shasti`, with a node name of `shasti_planetnt`.

Examples for UNIX-based platforms

Example of creating a configuration instance for user `shasti`:

```
wsinstance.sh -name shasti -path /usr/shasti/WebSphere -host planetaix -create
```

On a base WebSphere Application Server product installation, the command creates an application server configuration instance, named `shasti`, with a node name of `shasti_planetaix`.

On a Network Deployment product installation, the command creates a deployment manager configuration instance named `shasti`, in a cell named `shasti`, with a node name of `shasti_planetaix`.

Example of a creating a configuration instance for user `shasti` in a multiuser environment:

1. Create the configuration instance:

```
>wsinstance.sh -name shasti
               -path ~shasti/WebSphere
               -host myhost
               -startingPort 12000
               -create
```

2. Change the owner of the folder:

```
>chown shasti ~shasti/WebSphere
```

3. Add a call to script `~shasti/WebSphere/bin/setupCmdLine.sh` in the profile of user `shasti` to set the environment when he logs in. User `shasti` can go directly into the `WebSphere_install_root/bin` location and start the server.

4. Give these folder permissions to user `shasti`:

```
install_root/bin, java, --- rx(read, execute)
install_root/properties, deploytool,
                  config, lib, classes, null, samples, Web    ----rx(read,execute)
```

Example of deleting a configuration instance

The following command deletes the configuration instance named shasti:

```
wsinstance.sh -name shasti -host planetaix -delete
```

Creating servers in coexistence or multiple instance environments

WebSphere Application Server lets you create multiple servers based on an existing template, or using an existing server as a template. You can generate unique ports for the new server during its creation. Always select the unique port option when creating servers in a coexistence environment of multiple versions, installations or configuration instances, due to the likelihood of conflicting port assignments. Verify port assignments for the newly created server and change them if necessary.

1. Create a server (MyServer, for example) using either the administrative console (trun_svr_create in the InfoCenter) or wsadmin scripts.
2. Log on to the administrative console.
3. Go to Servers > Application Servers > MyServer > End Points (urun_rsvr in the InfoCenter).
4. Go into each end point and change the port numbers.
5. Change the HTTP Transport ports.
   a. Go to Servers > Application Servers > MyServer > Web Container > HTTP Transports (urun_rhttptransport_prop in the InfoCenter)
   b. Change the HTTP transport port numbers.
   c. Make a record of the new port numbers.
6. Change the JMS server ports, if you create servers from an application server template that is not federated into a deployment manager cell, but does have the embedded messaging feature.
   a. Go to **Servers > Application Servers > MyServer > Server Components > JMS Servers > Security Port Endpoint**.
   b. Change the port numbers.

Changing HTTP transport ports

This topic describes how to change HTTP transports manually by editing configuration files. Use this procedure when a conflicting HTTP transport setting is preventing an application server or deployment manager instance from starting.

Edit the configuration files to resolve conflicting port assignments, as described below.

1. Look for symptoms of port number conflicts.

   Troubleshooting topics describe symptoms and ways to identify and fix possible port number conflicts. Among the topics described are these:

- ″Administration and Administrative Console troubleshooting tips″ (rtrb_admincomp in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at

  http://www-3.ibm.com/software/webservers/appserv/infocenter.html)
- ″Client program does not work″ (rtrb_clientprobs in the InfoCenter)
- ″Debugging WebSphere Application Server applications″ (ttrb_debugwsa)
- ″Errors after enabling SSL, or SSL-related error messages″ (rtrb_sslprobs)
- ″Problems starting or using the wsadmin command″ (rtrb_wsadminprobs)
- ″Troubleshooting migration problems″ (rtrb_migratnprobs)
- ″Web Container troubleshooting tips″ (rtrb_webcntrcomp)
- ″Web module or application server dies or hangs″ (rtrb_appdies)
- ″Web resource (JSP file, servlet, HTML file, image) does not display″ (rtrb_pagedisplayprob)

2. **(Optional)** Edit an application server configuration.
   a. Open the `server.xml` configuration file for the `server1` application server.

      The file path for the server1 configuration, with a node name of **mynode**, is:

      `install_root/config/cells/mynode/nodes/mynode/servers/server1`

   b. Look for `transports xmi:type="applicationserver.webcontainer:HTTPTransport"`.

      The administrative console application uses transport ports 9090 and 9043. The sample applications use transport ports 9080 and 9443. Change the port numbers and save the file. Make a record of the new port numbers.

   c. Open the `virtualhosts.xml` file in the *install_root*`/config/cells/mynode` folder.

      This file contains alias entries for transport ports defined in the `server.xml` file.

   d. Look for `aliases xmi:id` to change port number assignments for any ports you changed.
3. **(Optional)** Edit a deployment manager node configuration.

   The deployment manager uses HTTP transport ports for the administrative console application. The default port is 9090.

   a. Open the `server.xml` configuration file for the `dmgr` deployment manager server.

      The file path for the dmgr configuration, with a cell name of **myManager**, and a node name of **mynode**, is:

      `install_root/config/cells/myManager/nodes/mynode/servers/dmgr`

   b. Look for `transports xmi:type="applicationserver.webcontainer:HTTPTransport"`.

      The administrative console application uses transport ports 9090 and 9043. Change the port numbers and save the file. Make a record of the new port numbers.

   c. Open the `virtualhosts.xml` file in the *install_root*`/config/cells/mynode` folder.

      This file contains alias entries for transport ports defined in the `server.xml` file.

   d. Look for `aliases xmi:id` to change port number assignments for any ports you changed.

Product version and history information

This topic describes XML data files that store product information for Version 5 WebSphere Application Server products. By default, the document type declarations (DTDs) for these files are in the `properties/version/dtd` folder of the install_root or the server root directory. See the *Storage locations* section for more information.

This topic includes:

- An overview of product information files
  - A description of file locations in the product tree
  - A list of reports for displaying product information
- A description of logging and backing up update operations
  - A description of the file naming convention
  - A description of storage locations
  - A description of how the update service makes operational use of the product information
- A data dictionary that describes data in the files

Product version and history information files

WebSphere Application Server products store version information in the `properties/version` directory as XML files. The following six types of files are in the directory:

platform.websphere
: One file whose existence indicates that a WebSphere Application Server product is installed. An example of the file follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE websphere PUBLIC "websphereId" "websphere.dtd">
<websphere name="IBM WebSphere Application Server" version="5.0"/>
```

Other files in this directory represent installed items and installation events:

***<product-id>*.product**
: One file whose existence indicates the particular WebSphere Application Server product that is installed. Data in the file indicates the version, build date, and build level. For example, the file might be the `ND.product` file, which indicates that the installed product is WebSphere Application Server Network Deployment. An example of the file follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE product PUBLIC "productId" "product.dtd">
<product name="IBM WebSphere Application Server for Network Deployment">
  <id>ND</id>
  <version>5.0.0</version>
  <build-info date="10/5/02" level="s0239.28"/>
</product>
```

***<component-name>*.component**
: Any number of component files that each indicate the presence of an installed component, which is part of the product. Data in the file indicates the component build date, build version, component name, and product version. For example, the file might be the `activity.component` file, which indicates that the ***activity*** component is installed. The activity component is part of the Network Deployment product. An example of the file follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component PUBLIC "componentId" "component.dtd">
<component build-date="10/5/02"
       build-version="s0239.28" name="activity" spec-version="5.0"/>
```

***<extension.id>*.extension**
: Any number of extension files that each indicate the presence of an extension that you install as a user extension, as part of a service engagement, or as installed by a third party product. The *<extension.id>*.extension files are not created, logged, or removed by WebSphere Application Server products.

***<efix-id>*.efix**
: Any number of efix files that each indicate the presence of an installed e-fix.

***<ptf-id>*.ptf**
: Any number of ptf files that each indicate the presence of an installed PTF.

Product version file locations

WebSphere Application Server products store version history information in the `properties/version/history` directory as XML files. The following five types of files are in the directory:

event.history
: One file that lists update events that have occurred. An update event is an operation that installs or uninstalls an e-fix or a PTF. The file is sorted by the date and time of the events that are listed.

Other files in this directory represent information about the e-fixes and PTFs that are currently installed, in four types of files:

***<efix-id>*.efixDriver**
: E-fix-driver defining information

***<efix-id>*.efixApplied**
: E-fix installation details

***<ptf-id>*.ptfDriver**
: PTF-driver defining information

***<ptf-id>*.ptfApplied**
: PTF installation details

These XML files are related to installation items by the primary ID information, which is shown here by *<angle brackets>* and italicized text.

Reports

You can view product information by examining files in the `properties/version` directory, including the `properties/version/history` directory.

WebSphere Application Server provides the ability to generate two types of reports about the data in the files, ***Version*** reports and ***History*** reports. The following report-generation scripts are available in the install_root `bin` directory.

Product version reports

The following report generation scripts extract data from XML data files in the `properties/version` folder.

versionInfo.bat
Lets you use parameters to create a version report on Windows NT or Windows 2000 platforms.

Parameters:

-format
TEXT | HTML

Selects the format of the report. The default is TEXT.

-file *<fileName>*

Specifies the output file name. The report goes to standard output (stdout) by default.

-displayComponents
Adds a list of installed components to the report.

-displayComponentDetail
Adds details about installed components to the report.

-displayEFixes
Adds a list of applied e-fixes to the report.

-displayEFixDetail
Adds details about applied e-fixes to the report.

-displayPTFs
Adds a list of applied PTFs to the report.

-displayPTFDetail
Adds details about applied PTFs to the report.

versionInfo.sh
Lets you use parameters to create a version report on UNIX-based platforms. Parameters are the same as for the Windows version.

genVersionReport.bat
Generates the `versionReport.html` report file in the `bin` directory on Windows NT or Windows 2000 platforms. The report includes the list of components, e-fixes, and PTFs.

genVersionReport.sh
Generates the `versionReport.html` report file in the `bin` directory on UNIX-based platforms. The report includes the list of components, e-fixes, and PTFs.

Product history reports

The following report generation scripts extract data from XML data files in the `properties/version/history` folder.

historyInfo.bat
Lets you use parameters to create a history report of installed and uninstalled e-fixes and PTFs, on Windows NT or Windows 2000 platforms. You can also specify a component name to create a report that shows the history for that component.

Parameters:

-format
TEXT | HTML

Selects the format of the report. The default is TEXT.

-file *<fileName>*

Specifies the output file name. The report goes to standard output (stdout) by default.

-updateID <ID>
Specifies the ID of an e-fix or PTF update. When specified, the product history report displays events for only the specified update. When not specified, the report displays events for all updates.

-component <componentName>
Specifies the name of a component. When specified, the product history report displays events for only the named component. When not specified, the report displays events for all components.

historyInfo.sh
Lets you use parameters to create a history report on UNIX-based platforms. Parameters are the same as for the Windows version.

genHistoryReport.bat
Generates the `historyReport.html` report file in the `bin` directory on Windows NT or Windows 2000 platforms. The report includes all updates and components.

genHistoryReport.sh
Generates the `historyReport.html` report file in the `bin` directory on UNIX-based platforms. The report includes all updates and components.

Logging and backing up update operations

WebSphere Application Server products use two other directories when performing update operations, for logging and backups. By default, the two directories are relative to the product version directory, as follows:

***<version.dir>*/log**
Product updates `log` directory

WebSphere Application Server products store log files to document component, e-fix and PTF operations and updates.

***<version.dir>*/backup**
Product updates `backup` directory

WebSphere Application Server products back up components before applying e-fixes and PTFs. If you uninstall an e-fix or PTF, WebSphere Application Server products restore the backed-up component JAR file.

File naming convention

Time stamp
YYYYMMDD_HHMMSS

For example: `20020924_211832` is 24-Sep-2002, 9:18:32 pm, GMT. All time stamps are in GMT.

ID E-fix ID or PTF ID

For example: apar6789c is an e-fix ID; PTF_1 is a PTF ID.

Operation
install | uninstall

E-fix log file names

<*timeStamp*>_<*efixId*>_<*operation*>.log

For example: properties/version/log/20020924_211832_apar6789c_install.log and properties/version/log/20020924_211912_apar6789c_uninstall.log

E-fix component log file names

<*timeStamp*>_<*efixId*>_<*componentName*>_<*operation*>.log

For example: properties/version/log/20020924_211832_apar6789c_ras_install.log and properties/version/log/20020924_211912_apar6789c_ras_uninstall.log

PTF log file names

<*timeStamp*>_<*ptfId*>_<*operation*>.log

For example: properties/version/log/20020924_211832_PTF_1_install.log and properties/version/log/20020924_211912_PTF_2_uninstall.log

PTF component log file names

<*timeStamp*>_<*ptfId*>_<*componentName*>_<*operation*>.log

For example: properties/version/log/20020924_211832_PTF_1_ras_install.log and properties/version/log/20020924_211912_PTF_2_ras_uninstall.log

Backup JAR file names

<*timeStamp*>_<*ptfId*>_<*componentName*>_undo.jar or <*timeStamp*>_<*efixId*>_<*componentName*>_undo.jar

For example: 20020924_211832_apar6789c_ras_undo.jar

Note: Do not delete a backup JAR file. You cannot remove a component update if the corresponding backup JAR file is not present.

Update processing might also use a temporary directory, if necessary. A Java property specifies this directory as described in the next section.

Storage locations

Product information files are located relative to the WebSphere Application Server product install_root, or the server root directory.

Default file paths and Java properties that set them are:

Version directory

<*was.install.root*>/properties/version or <*server.root*>/properties/version, specified by the was.version.dir Java property, or by the server.root Java system property if was.version.dir is not set

History directory

<*version.dir*>/history, specified by was.history.dir

Updates log directory

<*version.dir*>/log, specified by was.version.log.dir

Updates backup directory

<*version.dir*>/backup, specified by was.version.backup.dir

DTD directory

<*version.dir*>/dtd, specified by was.version.dtd.dir

Temporary directory

Specified by the was.version.tmp.dir Java property or by the java.io.tmpdir Java system property, if the was.version.tmp.dir property is not set

Operational description

WebSphere Application Server products update the product version history information while performing events that install or uninstall e-fixes or PTFs. Events that might occur include:

- A WebSphere Application Server product adds an e-fix file (with an extension of `.efix`) to the `version` directory to indicate that an e-fix is currently installed.
- A WebSphere Application Server product removes an e-fix file from the `version` directory when it uninstalls the corresponding e-fix.
- A WebSphere Application Server product adds an e-fix driver file (with an extension of `.efixDriver`) to the `history` directory when an efix is installed. An e-fix driver file contains defining information for an e-fix.
- A WebSphere Application Server product removes an e-fix driver file when it removes the corresponding e-fix.
- A WebSphere Application Server product adds an e-fix application file (with an extension of `.efixApplied`) to the `history` directory when it installs an e-fix. An e-fix application file contains information that identifies component updates that have been applied for an e-fix. The application file also provides links to component log and backup files.
- A WebSphere Application Server product removes an e-fix application file when it removes the corresponding e-fix.
- A WebSphere Application Server product adds a PTF file (with an extension of `.ptf`) to the `version` directory to indicate than a PTF is currently installed.
- A WebSphere Application Server product removes a PTF file from the version directory when it uninstalls the corresponding PTF.
- A WebSphere Application Server product adds a PTF driver file (with an extension of `.ptfDriver`) to the `history` directory when it installs a PTF. A PTF driver file contains defining information for a PTF.
- A WebSphere Application Server product adds a PTF application file (with an extension of `.ptfApplied`) to the `history` directory when it installs a PTF. A PTF application file contains information that identifies component updates that have been applied for a PTF. The application file also provides links to component log and backup files.
- A WebSphere Application Server product makes entries in the history file, `event.history`, when it installs or uninstalls an update.
- A WebSphere Application Server product stores a parent event for each e-fix that it installs or uninstalls.
- A WebSphere Application Server product stores a parent event for each PTF that it installs or uninstalls.
- A WebSphere Application Server product stores child component events for each component update that it installs or uninstalls, beneath the corresponding e-fix or PTF parent event.
- A WebSphere Application Server product stores one log file in the `log` directory as it installs or uninstalls one e-fix or PTF.
- A WebSphere Application Server product stores one log file in the `log` directory as it installs or uninstalls an e-fix or PTF, in response to each component update that occurs.
- A WebSphere Application Server product stores a component backup file in the backup directory for each component update that it installs.

- A WebSphere Application Server product removes a component backup file from the backup directory for each component update that it uninstalls.

Data dictionary

```
Type Family:      websphere product family

File Types:       websphere

File Type:        websphere

Elements:         name                 string     required
                  version              string     required

Persistence:      <versionDir>/platform.websphere

Type Detail:

The websphere file is placed to denote the presence of websphere family
products.

Element Detail:

websphere.name          The WebSphere product family name.
websphere.version       The WebSphere product family version.


Type Family:      product

File Types:       product
                  component
                  extension

File Type:        product

Persistence:      <versionDir>/<id>.product

Elements:         id                   string     required
                  name                 string     required
                  version              string     required
                  build-info           complex    required

Type Detail:

A product file is placed to denote the presence of a specific WebSphere family product.
The product's id is embedded in the product file name.

Element Detail:

product.id              The id of the product.
product.name            The name of the product.
product.version         The version of the product.
product.build-info      An element containing build information for
                        the product.

Element Type:     build-info

Elements:         date                 date       required
                  level                string     required

Type Detail:

A build-info instance details the build of a specific installed websphere
family product.

Element Detail:
```

```
build-info.date          The date on which the product was build.
build-info.level         The level code of the product's build.

File Type:        component

Persistence:      <versionDir>/<name>.component

Elements:         name                  string     required
                  spec-version          string     required
                  build-version         string     required
                  build-date            date       required

File Detail:

A component file denotes the presence of a specific component.  The
component name is embedded in the component file name.

Element Detail:

component.name           The name of the component.
component.spec-version   The specification version of the component.
component.build-version  The build level of the component.
component.build-date     The build date of the component.

File Type:        extension

Persistence:      <versionDir>/<id>.extension

Elements:         id                    string     required
                  name                  string     required

File Detail:

An extension file denotes the presence of a specific extension.  The
extension's id is embedded in the extension file name.

The elements of an extension file are minimally specified.  The listed
elements are required.  Additional elements may be present as determined
by the actual installed extension.

Element Detail:

extension.id             The id of the extension.
extension.name           The name of the extension.


Type Family:      update

File Types:       efix
                  ptf
                  efix-applied
                  ptf-applied

File Type:        efix

Persistence:      <versionDir>/<id>.efix

Elements:         id                    string     required
                  apar-number           string     optional
                  pmr-number            string     optional
                  short-description     string     required
                  long-description      string     required
                  is-temporary          boolean    required
                  build-version         string     required
                  build-date            date       required
                  component-update      complex    min=1, max=unbounded
                  platform-prereq       complex    min=0, max=unbounded
                  product-prereq        complex    min=0, max=unbounded
                  efix-prereq           complex    min=0, max=unbounded
                  custom-property       complex    min=0, max=unbounded

Type Detail:

An efix file denotes the presence of some portion of a specific efix.  The
id of the efix is embedded in the efix file name.

An efix file contains all efix data, such as description, a listing of
component updates, and prerequisite information.

Almost always, when installing an efix, all of the potential component
updates within the efix are required to be installed.

A separate application file must be examined to determine the components
which have been updated for a particular efix.

A list of custom properties may be provided.  These are provided for
future use.

Element Detail:

efix.id                   The id of the efix.
efix.short-description    A short description of the efix.
efix.long-description     A long description of the efix.
efix.is-trial             A flag indicating whether or not an efix is considered
                          a trial fix.  Generally, a trial fix will be
                          followed up with a more permanent fix.
efix.expiration-date      A date on which the efix is to be considered obsolete.
efix.build-version        The build version of the efix.  This is distinct from
                          the build version of component updates contained within
                          the efix.
efix-build-date           The build date of the efix.  This is distinct from the
                          build version of the component updates contained within
                          the efix.
efix.apar-info            A list of APAR's which are associated with the efix.
efix.component-update     A list of updates for components.  For an efix, these are
                          usually all required, and are all patch updates.  At least
                          one component update must be present.
efix.efix-prereq          A list of prerequisite efixes for the efix.  Note that
                          prerequisite efixes may be negative (see below).  This
                          list may be (and is often) empty.
efix.plaform-prereq       A list of platforms on which the efix may be installed.
                          This list may be empty, in which case the efix may be
                          installed on all platforms.
efix.product-prereq       A list of products on which the efix may be installed.
                          This list may be empty, in which case the efix may be
                          installed on all products.
efix.custom-proprty       A list of properties, provided for future use.

File Type:        ptf

Persistence:      <versionDir>/<id>.ptf

Elements:         id                    string     required
                  short-description     string     required
                  long-description      string     required
                  build-version         string     required
                  build-date            date       required
                  component-update      complex    min=1, max=unbounded
                  product-update        complex    min=0, max=unbounded
                  platform-prereq       complex    min=0, max=unbounded
                  product-prereq        complex    min=0, max=unbounded
                  included-efix         complex    min=0, max=unbounded
                  custom-property       complex    min=0, max=unbounded
```

```
Type Detail:

A ptf file denotes the presence of some portion of a specific PTF.  The id of
the PTF is embedded in the ptf file name.

A ptf file contains all PTF data, such as description, a listing of component
updates, and prerequisite information.

Usually, when installing a ptf, certain potential component updates may be
omitted (but, only when the corresponding component is not installed).

A separate application file must be examined to determine the components which
have been updated for a particular ptf.

A ptf may include updates for a number of efixes.

A list of custom properties may be provided.  These are provided for future use.

Element Detail:

ptf.id                   The id of the ptf.
ptf.short-description    A short description of the ptf.
ptf.long-description     A long description of the ptf.
ptf.build-version        The build version of the ptf.  This is distinct from the
                         build version of component updates contained within the ptf.
ptf-build-date           The build date of the ptf.  This is distinct from the
                         build version of the component updates contained within the
                         ptf.
ptf.component-update     A list of updates for components.  For an ptf, these are
                         usually all required, and are all patch updates.  At least
                         one component update must be present.
ptf.plaform-prereq       A list of platforms on which the ptf may be installed.  This
                         list may be empty, in which case the ptf may be installed on
                         all platforms.
ptf.product-prereq       A list of products on which the ptf may be installed.  This
                         list may be empty, in which case the ptf may be installed on
                         all products.
ptf.included-efix        A list of efixes which are included (fixed) by the ptf.
ptf.custom-proprty       A list of properties, provided for future use.

Element Type:     component-update

Elements:         component-name        string     required
                  update-type           enum       required [enumUpdateType]
                  is-required           boolean    required
                  is-recomended         boolean    required
                  is-optional           boolean    required
                  is-external           boolean    required
                  root-property-file    anyURL     optional
                  root-property-name    string     optional
                  root-property-value   anyURL     optional
                  is-custom             boolean    required
                  primary-content       string     required
                  component-prereq      complex    min=0, max=unbounded
                  final-version         complex    optional
                  custom-property       complex    min=0, max=unbounded

Type Detail:

A component update represents a potential component update which is packaged
in an update (an efix or a ptf).

An component update may be required, in which case the parent update may not
be installed unless the component update can be installed.  (A component update
can be installed if the corresponding component is installed.)

A component update may be a custom update, in which case the content which
```

```
was provided must be an executable file.  Otherwise, the content which is
provided must be an update jar file.

A component update has a type.  A final version may be required according to
the update type.

Element Detail:

component-update.component-name   The name the component which is to be updated.
component-update.update-type      The type of the component update, one of 'add',
                                  'replace', 'remove', or 'patch'.  Final version
                                  information must be provided when the update type
                                  is 'add' or 'replace'.
component-update.is-required      A flag which, when true, specifies that the parent
                                  update may not be applied unless this component
                                  update is applied.
component-update.is-recomended    A flag which, when true, specifies that this
                                  component update, although optional, should be
                                  installed.
component-update.is-optional      A flag which, when true, specifies that this update
                                  may be omitted even if its corresponding component
                                  is installed.
component-update.is-external          A flag which, when true, specifies that this
                                      component update may live outside of the usual
                                      install root.
component-update.root-property-file   For a component with an external root, this
                                      properties file provides the root value.
component-update.root-property-name   For a component with an external root, this named
                                      property provides the root value.
component-update.root-property-value  For a component with an external root, this value
                                      provides the default root value.
component-update.is-custom        A flag which, when true, specifies that the update
                                  is a custom update.  When true, the content must
                                  be an executable program.  When false, the content
                                  must be an update jar.
component-update.primary-content  The name of the content which is provided for the
                                  update.  This will be an entry which is packaged
                                  in the 'components' directory of the update.
component-update.component-prereq A list of component versions, one of which must
                                  be present for this update to be installed.
                                  When this list is empty, any component version
                                  is allowed.
component-update.final-version    Final version information for the component.
                                  A final version is required when the update
                                  operation is 'add' or 'replace'.
component-update.custom-property  A list of properties, provided for future use.

Element Type:     apar-info

Elements:         number                string     required
                  date                  date       required
                  short-description     string     required
                  long-description      string     optional

Type Detail:

An apar-info object provides information about an APAR which is associated
with an efix, usually indicating that the efix provides a fix for the APAR.

Element Detail:

apar-info.number             The number of the associated APAR.
apar-info.date               The date of the APAR.
apar-info.short-description  A short description of the APAR.
apar-info.long-description   An optional long description of the APAR.

Element Type:     efix-prereq
```

```
Elements:         efix-id               string     required
                  is-negative           boolean    required
                  install-index         int        optional

Type Detail:

A efix prerequisite instance denotes an efix which must be present (or, if negative,
must be absent) for the parent efix to be installed.

efix prerequisite instances may specify a cycle, in which case the prerequisite
specification is treated as a corequisite specification.

The following chart summarizes the interpretation of prerequisite information for
two efixes:

       efix1      efix2
         -          -         The efixes may be installed without regard to each other.
       efix2        -         efix1 must be installed after efix2 is installed.
         -        efix1       efix2 must be installed after efix1 is installed.
       efix2      efix1       efix1 and efix2 must be installed together.
      !efix2        -         efix1 may not be installed after efix2 is installed.
         -       !efix1       efix2 may not be installed after efix1 is installed.
      !efix2     !efix1       efix1 and efix2 may not ever be installed together.
      !efix2      efix1       This is an erroroneous specification.
       efix2     !efix1       This is an erroroneous specification.

The installation index element provides ordering information for corequisite efixes
which must be installed in a particular order.

Element Detail:

efix-prereq.efix-id        The id of the prerequisite efix.
efix-prereq.is-negative    A flag which indicates if the prerequisite
                           efix is required or prohibited.  If false,
                           the efix must be installed before the parent
                           efix may be installed  If true, the efix must
                           not be installed when the parent efix is
                           installed.
efix-prereq.install-index  An optional index number which is used to
                           order corequisite efixes.

Element Type:     product-update

Elements:         product-id            string     required
                  product-name          string     required
                  build-version         string     required
                  build-date            date       required
                  build-level           string     required

Type Detail:

A product update specifies a replacement to a product file.

The product update information matches the information in product files.

Multiple product updates may be present, in which case each matching
product is updated.

Element Detail:

product-update.product-id     The id of the product which is updated.
product-update.product-name   The name of the product.
product-update.build-version  The build version of the product.
product-update.build-date     The build date of the product.
product-update.build-level    The build level of the product.
```

```
Element Type:     component-prereq

Elements:         component-name       string     required
                  spec-version         string     required
                  build-version        string     required
                  build-date           date       required

Element Type:     platform-prereq

Elements:         architecture         string     required
                  os-platform          string     optional
                  os-version           string     optional

Type Detail:

A platform prerequisite instance denotes a platform which must be present
for an update to be installed.  The element values are according to the
values supplied for the matching java properties.

Note that when multiple platform prerequisites are specified, these
prerequisites have an OR relationship: At least one of the platform
prerequisites must be satisfied.

Element Detail:

platform-prereq.architecture  The name of an architecture which must be
                              present.
platform-prereq.os-platform   The name of an operating system which
                              must be present.  This element is optional.
                              When absent, the architecture is checked,
                              but the os-platform and os-version are not.

platform-prereq.os-version    The version of a the operating system which
                              must be present.  This element is optional.
                              When absent, the architecture and os-platform
                              are checked, but os-version is not.  (When
                              os-platform is absent, os-version should not
                              be set.)

Element Type:     product-prereq

Elements:         product-id           string     required
                  build-version        string     optional
                  build-date           date       optional
                  build-level          string     optional

Type Detail:

A product prerequisite specifies that a particular product must be present for
an update to be installed.

Note that when multiple product prerequisites are specified, these
prerequisites have an OR relationship: At least one of the product
prerequisites must be satisfied.

Note that all of the elements are required.  When multiple products having
the same id are supported by an update, multiple product prerequisites must
be specified.

Element Detail:

product-prereq.product-id     The id of the product which must be present.
product-prereq.build-version  The version of the product which must be present.
product-prereq.build-date     The build date of the product which must be present.
product-prereq.build-level    The level date of the product which must be present.

Element Type:     component-prereq
```

```
Elements:         component-name        string     required
                  spec-version          string     required
                  build-version         string     required
                  build-date            date       required

Type Detail:

A version prerequisite specifies that a particular component version must be
present for an update to be installed.

Note that when multiple version prerequisites are specified, these
prerequisites have an OR relationship: At least one of the version
prerequisites must be satisfied.

Element Detail:

version-prereq.component-name  The name of the component which must be present.
version-prereq.spec-version    The specification version of the component which
                               must be present.
version-prereq.build-version   The version of the component which must be present.
version-prereq.build-date      The build date of the component which must be
                               present.

Element Type:     included-efix

Elements:         efix-id               string     required

Type Detail:

An included efix identifies an efix by id and indicates that efix
is fixed by the ptf.

Element Detail:

included-efix.efix-id    The id of the efix which is fixed by the ptf

Element Type:     custom-property

Elements:         property-name         string     required
                  property-type         string     optional
                  property-value        string     optional

Type Detail:

A custom property encodes a key-value pair, with an optional type
element.  Custom properties are provided for future use.

Element Detail:

custom-property.property-name   The name of the custom property.
custom-property.property-type   An optional type of the custom property.
                                The semantics of this type are defined
                                by user of the property value.
custom-property.property-value  The value of the custom property.

File Type:        efix-applied

Persistence:      <versionDir>/<id>.efixApplied

Elements:         efix-id               string     required
                  component-applied     complex    min=0, max=unbounded

Type Detail:

An efix-applied collection specified what components have been updated for
the efix as specified by the efix id.
```

```
Element Detail:

efix-applied.efix-id            The id of the efix for which applieds are
                                recorded.
efix-applied.component-applied  The list of recorded applications.

File Type:        ptf-applied

Persistence:      <versionDir>/<id>.ptfApplied

Elements:         ptf-id                string     required
                  component-applied     complex    min=0, max=unbounded

Type Detail:

A ptf-applied collection specified what components have been updated for
the ptf as specified by the ptf id.

Element Detail:

ptf-applied.efix-id            The id of the ptf for which applieds are
                               recorded.
ptf-applied.component-applied  The list of recorded applications.

Element Type:     component-applied

Elements:         component-name        string     required
                  update-type           enum       required [enumUpdateType]
                  is-required           boolean    required
                  is-optional           boolean    required
                  is-external           boolean    required
                  root-property-file    anyURL     optional
                  root-property-name    string     optional
                  root-property-value   string     optional
                  is-custom             boolean    required
                  log-name              anyURL     required
                  backup-name           anyURL     required
                  time-stamp            date       required
                  initial-version       complex    optional
                  final-version         complex    optional

Type Detail:

An applied instance is present to indicate the application of an
update for a particular efix or ptf to a particular component.
(The particular efix or ptf is as specified by the applied's
parent.)  An applied provides sufficient information
to undo itself.

The elements of an applied are copies of values from update
events.

Element Detail:

component-applied.component-name   The name of the component which was updated.
component-applied.update-type      The type of the component update.
component-applied.is-required      A flag which, when true, specifies that the parent
                                   update requires this component update.
component-applied.is-optional      A flag which, when true, specifies that the parent
                                   update does not require this component update,
                                   even if the component is installed.
component-applied.is-external          A flag which, when true, specifies that this
                                       component update was applied to a location
                                       different than the usual install_root.
component-applied.root-property-file   For an update against a component having an
                                       external root, this properties file provides
                                       the root value.
component-applied.root-property-name   For an update against a component having an
                                       external root, this named property provides
                                       the root value.
component-applied.root-property-value  For an update against a component having an
                                       external root, this is a record of the
                                       actual external root.
component-applied.is-custom        A flag which, when true, specifies that the
                                   application was a custom update.  When true, an
                                   executable program was applied.  When false, the
                                   contents of an update jar were applied.
component-applied.log-name         The name of the log file which was generated by
                                   this application.
component-applied.backup-name      The name of the backup file which was generated
                                   by this application.
component-applied.time-stamp       The time of this application (the ending time
                                   of the corresponding update event).
component-applied.initial-version  The version of the component before the
                                   application.  This version will be null if
                                   the application was an add.
component-applied.final-version    The version of the component after the application.
                                   This will be null if the update was a removal.

Element Type:     initial-version

Elements:         component-name        string     required
                  spec-version          string     required
                  build-version         string     required
                  build-date            string     required

Type Detail:

A initial-version instance is used to describe a component level as
the initial version of a component.

Element Detail:

initial-version.component-name The name of the component.
initial-version.spec-version   The new specification version for the
                               component following the update.
initial-version.build-version  The new build version for the component.
initial-version.build-date     The new build date for the component.

Element Type:     final-version

Elements:         component-name        string     required
                  spec-version          string     required
                  build-version         string     required
                  build-date            string     required

Type Detail:

A final-version instance is used to supply a component level for a
component which has been added or replaced.

Element Detail:

final-version.component-name The name of the new component.
final-version.spec-version   The new specification version for the
                             component following the update.
final-version.build-version  The new build version for the component.
final-version.build-date     The new build date for the component.

Enum Type:        enumUpdateType

Values:           0 add
                  1 replace
```

```
                 2 remove
                 3 patch

Type Detail:

An update type instance specifies the type of an update.  An 'add' update adds
a component into an installation.  A 'replace' update replaces a particular
version of a component with a different version of that component.  A 'remove'
update removes a component.  A 'patch' update performs a limited update to a
component, in particular, without changing the version of the component.

When adding a component, that component may not already be present.
When replacing or removing a component, that component must be present.
When patching a component, that component must be present.

When replacing or removing a component, or when patching a component, usually,
at least one version prerequisite will be specified for the component update.

Value Detail:

enumUpdateType.add       Specifies that an update adds a component.
enumUpdateType.replace   Specifies that an update replaces a component.
enumUpdateType.remove    Specifies that an update removes a component.
enumUpdateType.patch     Specifies that an update modifies a component, but
                         does not change its version.


Type Family:      history

File Type:        event-history

Persistence:      <historyDir>/event.history

Elements:         update-event          complex    min=0, max=unbounded

Type Detail:

One event history is provided for a websphere product family installation.
This event history contains history of update events, corresponding with
the actual update events for that product family.

Element Detail:

event-history.update-event  The list of update events for the websphere
                            product family.  The top level events are efix
                            and ptf events, each containing one or more
                            component events.

Element Type:     update-event

Elements:         event-type            enum       required [enumEventType]
                  parent-id             string     required
                  id                    string     required

                  update-type           enum       required [enumUpdateType]

                  is-required           boolean    required
                  is-optional           boolean    required

                  is-external           boolean    required
                  root-property-file    anyURL     optional
                  root-property-name    string     optional
                  root-property-value   string     optional

                  is-custom             boolean    required
                  primary-content       anyURI     required
                  event-action          enum       required [enumEventAction]
                  log-name              anyURI     required
                  backup-name           anyURI     required
                  start-time-stamp      dateTime   required
                  end-time-stamp        dateTime   optional

                  status                enum       optional [enumEventResult]
                  status-message        string     optional

                  initial-version       complex    optional
                  final-version         complex    optional
                  update-event          complex    optional

Type Detail:

An update event denotes a single update action, applying to either an
efix, a ptf, or to a component, according to the set event type.

efix and ptf type events each have a collection of component events.

Currently, component events have no child events.

Element Detail:

update-event.event-type       The type of this event, either an efix or
                              ptf type event, or a component type event.
update-event.parent-id        This element is present only for component
                              events.  The id of the parent efix or ptf of
                              this event.
update-event.id               The id of the efix, ptf, or component which
                              was updated, interpreted according to the type
                              of the event.

update-event.update-type      The type of update for component events.

update-event.is-required      A flag which, when true, specifies that this
                              component update is required.
update-event.is-optional      A flag which, when true, specifies that this
                              component update is optional, even if the
                              component is installed.

update-event.is-external         A flag which, when true, specifies that this
                                 update used an external root.
update-event.root-property-file  For an update of an external component, this
                                 properties file contains the external root value.
update-event.root-property-name  For an update of an external component, the
                                 property having this name specifies the external
                                 root value.
update-event.root-property-value For an update of an external component,
                                 the root value.

update-event.is-custom        A flag which, when true, specifies that the
                              application was a custom update.  When true,
                              an executable program was applied.  When false,
                              the contents of an update jar were applied.
update-event.primary-content  The URL of the primary content for the update.

update-event.event-action     The type of action for this event.
update-event.log-name         The name of the log file which was generated
                              for this event.
update-event.backup-name      The name of the backup file which was generated
                              for this event.
update-event.start-time-stamp The XML timestamp of the starting time of the
                              event.  This timestamp follows the XML timestamp
                              format, meaning that time zone information is
                              included.
update-event.end-time-stamp   The XML timestamp of the ending time of the
```

```
                               event.  This timestamp follows the XML timestamp
                               format, meaning that time zone information is
                               included.  When absent, the update operation
                               corresponding to the parent event failed with
                               a non-recoverable exception.
update-event.status            The result of the update.
update-event.status-message    Message text provided in addition to the basic
                               status code.  Exception text is provided through
                               the status-message when an update fails.
update-event.initial-version   This element is not used unless the update is
                               a component type update.  The initial version of
                               the component which was updated.    This element
                               is absent when the update is an add type update.
update-event.final-version     This element is not used unless the update is a
                               component type update.  The final version of the
                               component which was updated.  This element is absent
                               when the update is a remove type update.
update-event.update-event      A collection of child events.  This collection is
                               used for efix and ptf type events.  This collection
                               is empty for component type events.

Element Type:     initial-version

Elements:         spec-version          string     required
                  build-version         string     required
                  build-date            string     required

Type Detail:

A initial-version instance is used to describe a component level as
the initial version of a component.

Element Detail:

initial-version.spec-version   The new specification version for the
                               component following the update.
initial-version.build-version  The new build version for the component.
initial-version.build-date     The new build date for the component.

Element Type:     final-version

Elements:         spec-version          string     required
                  build-version         string     required
                  build-date            string     required

Type Detail:

A final-version instance is used to supply a component level for a
component which has been added or replaced.

Element Detail:

final-version.spec-version   The new specification version for the
                             component following the update.
final-version.build-version  The new build version for the component.
final-version.build-date     The new build date for the component.

Enum Type         enumEventType

Values:           0 efix
                  1 ptf
                  2 component

Type Detail:

An event type instance specifies the type of an update event, which is either
an efix event, a ptf event or a component event.  The interpretation of
```

```
particular event elements depends on the set event type.

Value Detail:

enumEventType.efix       Specifies that an event is for an efix update.
enumEventType.ptf        Specifies that an event is for an ptf update.
enumEventType.component  Specifies that an event is for a component update.

Enum Type:        enumEventAction

Values:           0 install
                  1 uninstall
                  2 selective-install
                  3 selective-uninstall

Type Detail:

An event action instance specified the operation performed by an update, which
can be an install or uninstall operation, and which may be a selective operation.
Component operations are always either install or uninstall type operations, only
efix and ptf operations may be selective operations.

A selective operation is an installation which is applied to a preset list of
components.  In particular, potential component updates may be skipped, and
component updates which were already applied may be reapplied.

A selective uninstall operation is used to back out an update which was cancelled
by the user.

Value Detail:

enumEventAction.install              Specifies that an event is an install
                                     operation.
enumEventAction.uninstall            Specifies that an event is an uninstall
                                     operation.
enumEventAction.selective-install    Specifies that an event is an install
                                     operation with a preset list of components
                                     which are updated.
enumEventAction.selective-uninstall  Specifies that an event is an install
                                     operation with a preset list of components
                                     which are updated.

Enum Type:        enumUpdateType

Values:           0 add
                  1 replace
                  2 remove
                  3 patch

Type Detail:

An update type instance specifies the type of a component update.  An 'add'
update adds a component into an installation.  A 'replace' update replaces
a particular version of a component with a different version of that component.
A 'remove' update removes a component.  A 'patch' update performs a limited
update to a component, in particular, without changing the version of the
component.

When adding a component, that component may not already be present.
When replacing or removing a component, that component must be present.
When patching a component, that component must be present.

When replacing or removing a component, or when patching a component, usually,
at least one version prerequisite will be specified for the component update.

Value Detail:
```

```
enumUpdateType.add       Specifies that an update adds a component.
enumUpdateType.replace   Specifies that an update replaces a component.
enumUpdateType.remove    Specifies that an update removes a component.
enumUpdateType.patch     Specifies that an update modifies a component, but
                         does not change its version.

Enum Type:        enumEventResult

Values:           0 succeeded
                  1 failed
                  2 cancelled

Type Detail:

An event result instance denotes a particular result for an update event,
indicating that the corresponding update was successful, failed, or was
cancelled.

Value Detail:

enumEventResult.succeeded  Specifies that the operation was successful.
enumEventResult.failed     Specifies that the operation failed.
enumEventResult.cancelled  Specifies that the operation was cancelled.
```

Uninstalling WebSphere Application Server

WebSphere Application Server Network Deployment provides an uninstaller program.

The uninstaller program removes registry entries, uninstalls the product, and removes all related features and products, such as plug-ins. However, there are some files that the uninstall program does not remove. The uninstall program does not delete any configuration files that are changed as the result of selecting installation options, or running Samples, for example. To delete all files so that you can reinstall with a clean system, perform a manual uninstall.

You can also uninstall the product manually.

Depending on the IBM WebSphere Application Server, Version 5 products you installed, uninstall the ones that you intend to uninstall in this order:
1. IBM WebSphere Application Server Network Deployment
2. IBM WebSphere Application Server (base product)

This section contains the following topics:
- Procedure for uninstalling IBM WebSphere Application Server Network Deployment and its features
- Manually uninstalling on Windows NT or Windows 2000 platforms
- Manually uninstalling on Linux platforms
- Manually uninstalling on Solaris platforms
- Manually uninstalling on AIX platforms

Procedure for uninstalling IBM WebSphere Application Server Network Deployment and its features

1. Stop any embedded messaging feature services, such as WebSphere Embedded Messaging Publish And Subscribe, JMS servers, or MQ Series Queue Managers, and any related Java processes.
2. Stop the IBM HTTP Server and any related Java processes.

3. Stop any WebSphere Application Server Java processes with the **stopManager** or **stopServer** commands.
4. Run the wizard for uninstalling the program.
   - On Windows NT and Windows 2000 platforms, click **Settings > Control Panel > Add/Remove Programs > WebSphere Application Server Network Deployment**. Removing the product this way calls the Uninstall.exe program: *ND_install_root*_uninst\Uninstall.exe Or, you can call the program directly from the location shown.
   - On Linux/390 platforms, call the uninstall.sh program: *ND_install_root*/_uninst/uninstall.sh
   - On other Linux platforms, or on UNIX-based platforms, call the uninstall program: *ND_install_root*/_uninst/uninstall
5. **(Optional)** Uninstall silently, by using the -silent option on the command.

   Silently uninstalling does not display the wizard. Command syntax is the same except for the -silent option. For example:
   - On Windows NT and Windows 2000 platforms: *install_root*_uninst\Uninstall.exe -silent
   - On Linux/390 platforms: *install_root*/_uninst/uninstall.sh -silent
   - On other Linux platforms, or on UNIX-based platforms: *install_root*/_uninst/uninstall -silent
6. Uninstall manually if you cannot successfully use the uninstaller.

   Manually uninstall the product if the uninstaller program is not present, or if an aborted installation did not create a complete and functional uninstaller program.
   - ″Manually uninstalling on Windows NT or Windows 2000 platforms″
   - ″Manually uninstalling on Linux platforms″
   - ″Manually uninstalling on Solaris platforms″
   - ″Manually uninstalling on AIX platforms″

Manually uninstalling on Windows NT or Windows 2000 platforms

Always use the Uninstall.exe program, if it exists. In some cases, such as an aborted installation, the **Uninstall.exe** command might not exist, or might not be complete and functional. For example, there are several valid scenarios where parts of the embedded messaging feature are not uninstalled, including:
- A message broker is still defined.
- The WebSphere Embedded Messaging Publish and Subscribe Edition (WEMPS) is not removed.
- A message queue manager is still defined.
- Java Message Service and MQ Series are not removed.

Use the following procedure to remove all remnants of WebSphere Application Server so that you can install again.

There are three items you must delete to remove a WebSphere Application Server product or feature: files, registry entries, and MSI record.

Manually uninstalling is generally the same for the Network Deployment product as it is for the base WebSphere Application Server product.

Depending on the WebSphere Application Server products you installed, uninstall the ones that you intend to uninstall in this order:

1. WebSphere Application Server Network Deployment
2. WebSphere Application Server (base product)

In the following steps:

- The default *WAS_install_root* for WebSphere Application Server is the `<drive>:\Program Files\WebSphere\AppServer` directory.
- The default *ND_install_root* for Network Deployment is the `<drive>:\Program Files\WebSphere\DeploymentManager` directory.

1. Ensure you have an Emergency Recovery Disk. Instructions for creating this disk are in the Windows help documentation.
2. Use the `regback.exe` program from the Windows NT or Windows 2000 Resource Kit to back up the Registry.
3. Run the `*ND_install_root*\_uninst\Uninstall.exe` program for Network Deployment , if it exists.
4. Run the `*WAS_install_root*\_uninst\Uninstall.exe` program for WebSphere Application Server, if it exists.
5. Restart the machine as the Uninstall.exe program suggests.
6. Delete product registry entries.

   Remove any product or feature that appears in the Add/Remove Programs panel, which is available from the Control Panel.
7. Invoke **`regedit.exe`** from a command prompt, to edit the Windows System Registry.

   Handle the Registry with care!: You can easily make a mistake while using the regedit.exe editor to view and edit Registry contents. The editor does not warn you of editing errors, which can be extremely dangerous. A corrupt Registry can disrupt your system to the point where your only option is to reinstall the Windows NT or Windows 2000 operating system.
   a. Search (using **Ctrl-F**) for all instances of ***WebSphere***, ***IBM HTTP Server***, or ***IBM MQSeries***, to determine whether you should delete each entry. You might not be able to remove all of the entries related to WebSphere Application Server, which is not a problem.
   b. Expand and select keys related to **WebSphere Application Server** and its features, **IBM HTTP Server**, and **IBM MQ Series**.

      Keys to delete include:
      - HKEY_LOCAL_MACHINE\SOFTWARE\IBM\ HTTP Server\1.3.26
      - HKEY_LOCAL_MACHINE\SOFTWARE\IBM\ MQSeries\CurrentVersion
      - HKEY_LOCAL_MACHINE\SOFTWARE\IBM\ WebSphere Application Server\5.0.0.0
      - HKEY_LOCAL_MACHINE\SOFTWARE\IBM\ WebSphere Network Deployment\5.0.0.0
   c. Click **Edit** > **Delete** from the menu bar for each related key.
   d. Click **Yes** when asked to confirm deletion of the key.
   e. Click **Registry** > **Exit** from the menu bar when you are finished.
8. Restart the Windows NT or Windows 2000 system.

   Reboot to install any of the products again.
9. Use the Microsoft MSI Cleanup Utility to remove MRI records.

The utility is available from the Microsoft Web site as `msicuu.exe`. Click on the `msicuu.exe` file after downloading it, to install. Once installed, it appears on the program menu.

When MSI starts, it lists all products that it knows about. To uninstall using this technique:

a. Select a product and click **Remove** to remove its MSI record.
b. Restart your machine.

10. Delete the base product install_root directory, `<`*`drive`*`>:\WebSphere\AppServer` and all subdirectories.
11. Delete the Network Deployment install_root directory, `<`*`drive`*`>:\WebSphere\DeploymentManager`, and all subdirectories.
12. Delete the `<`*`drive`*`>:\IBMHttpServer` install_root, which is usually in the `<`*`drive`*`>:\` root directory by default.
13. Delete the **<*drive*>:\IBM\MQSeries** install_root.
14. Edit the `vpd.properties` file.
    a. Locate the `vpd.properties` file in the operating system installation directory.

       For example, `c:\windows` or `c:\winnt`.
    b. Remove all lines containing the "WSB", "WSN", or "WSE" strings.
    c. Save the file and close it.

    Do not delete or rename the `vpd.properties` file because the InstallShield for MultiPlatforms (ISMP) program uses it for other products that it installs.

When you finish, the product is completely uninstalled. You are now ready to reinstall.

Manually uninstalling on Linux platforms

Always use the uninstall program (or the uninstall.sh program for Linux/390 platforms), if it exists. In some cases, such as an aborted installation, the uninstall (or uninstall.sh) program might not exist, or might not be complete and functional. For example, there are several valid scenarios where parts of the embedded messaging feature are not uninstalled, including:

- A message broker is still defined.
- The WebSphere Embedded Messaging Publish and Subscribe Edition (WEMPS) is not removed.
- A message queue manager is still defined.
- Java Message Service and MQ Series are not removed.

Use the following procedure to remove all remnants of WebSphere Application Server so that you can install again.

Depending on the WebSphere Application Server products you installed, uninstall the ones that you intend to uninstall in this order:

1. WebSphere Application Server Network Deployment
2. WebSphere Application Server (base product)

In the following steps:

- The default *WAS_install_root* for WebSphere Application Server is the `/opt/WebSphere/AppServer` directory.

- The default *ND_install_root* for Network Deployment is the `/opt/WebSphere/DeploymentManager` directory.

Steps for this task

1. Type **`killall -9 java`** to ensure that no Java processes are running.
2. Halt any running MQ Series queue managers.
   a. Type **`dspmq`** to show the state of any queue managers.
   b. Type **`endmqm -i`** for each running queue manager.
3. Type **`kill -9`** ***`<amq_pid_1> <amq_pid_2>`*** **`...`** ***`<amq_pid_n>`*** to ensure that no MQ Series processes are running.
4. Run the *`ND_install_root`*`/_uninst/uninstall` program for Network Deployment, if it exists.
5. Run the *`WAS_install_root`*`/_uninst/uninstall` program for WebSphere Application Server, if it exists.
6. Search for related packages.

   Type these commands to search for related packages:

   - **`rpm -qa | grep WS`** to show packages for the base WebSphere Application Server product, the Network Deployment product, and the IBM HTTP Server product
   - **`rpm -qa | grep MQ`** to show packages for the embedded messaging feature, which is based on IBM MQSeries technology
   - **`rpm -qa | grep wemps`** to show more packages for the embedded messaging feature

   The examples below show typical package names that might appear.

   If no packages appear when using these commands, skip the next step.
7. Type **`rpm -e`** ***`<packagename>`*** to remove any WebSphere Application Server-related packages.

   For example, execute these commands as root:

```
rpm -e MQSeriesClient-5.3.0-1
rpm -e MQSeriesMsg_Zh_CN-5.3.0-1
rpm -e MQSeriesMsg_Zh_TW-5.3.0-1
rpm -e MQSeriesMsg_de-5.3.0-1
rpm -e MQSeriesMsg_es-5.3.0-1
rpm -e MQSeriesMsg_fr-5.3.0-1
rpm -e MQSeriesMsg_it-5.3.0-1
rpm -e MQSeriesMsg_ja-5.3.0-1
rpm -e MQSeriesMsg_ko-5.3.0-1
rpm -e MQSeriesMsg_pt-5.3.0-1
rpm -e MQSeriesRuntime-5.3.0-1
rpm -e MQSeriesSDK-5.3.0-1
rpm -e MQSeriesJava-5.3.0-1
rpm -e MQSeriesServer-5.3.0-1
rpm -e MQSeriesJava-5.3.0-1
rpm -e wemps-runtime-2.1.0-0
rpm -e wemps-msg-De_DE-2.1.0-0
rpm -e wemps-msg-Es_ES-2.1.0-0
rpm -e wemps-msg-Fr_FR-2.1.0-0
rpm -e wemps-msg-It_IT-2.1.0-0
rpm -e wemps-msg-Ja_JP-2.1.0-0
rpm -e wemps-msg-Ko_KR-2.1.0-0
rpm -e wemps-msg-Pt_BR-2.1.0-0
rpm -e wemps-msg-Zh_CN-2.1.0-0
rpm -e wemps-msg-Zh_TW-2.1.0-0
```

8. Type **`rm -rf /opt/WebSphere/AppServer/ /opt/WebSphere/DeploymentManager/`** to remove the directories. If there are no other entries, you can remove the `WebSphere` directory.
9. Type **`rm -fr /var/mqm /var/wemps /opt/mqm /opt/wemps`** if you are certain that there is no embedded messaging data to preserve.

When the process completes, the product is completely uninstalled. You are now ready to reinstall.

Usage scenario

Example of displaying package names beginning with MQ, for the embedded messaging feature

```
==>rpm -qa | grep MQ
MQSeriesMsg_Zh_CN-5.3.0-1
MQSeriesMsg_Zh_TW-5.3.0-1
MQSeriesMsg_ko-5.3.0-1
MQSeriesClient-5.3.0-1
MQSeriesMsg_de-5.3.0-1
MQSeriesMsg_es-5.3.0-1
MQSeriesMsg_fr-5.3.0-1
WSBMQ1AA-5.0-0
WSBMQ2AA-5.0-0
WSBMQ3AA-5.0-0
MQSeriesMsg_it-5.3.0-1
MQSeriesMsg_ja-5.3.0-1
MQSeriesMsg_pt-5.3.0-1
MQSeriesSDK-5.3.0-1
MQSeriesJava-5.3.0-1
MQSeriesServer-5.3.0-1
MQSeriesRuntime-5.3.0-1
```

Example of displaying package names beginning with wemps, for the embedded messaging feature

```
==>rpm -qa | grep wemps
wemps-msg-De_DE-2.1.0-0
wemps-msg-Es_ES-2.1.0-0
wemps-msg-Fr_FR-2.1.0-0
wemps-msg-It_IT-2.1.0-0
wemps-msg-Ja_JP-2.1.0-0
wemps-msg-Ko_KR-2.1.0-0
wemps-msg-Pt_BR-2.1.0-0
wemps-msg-Zh_CN-2.1.0-0
wemps-msg-Zh_TW-2.1.0-0
wemps-runtime-2.1.0-0
```

Example of displaying package names beginning with WSB, for the base WebSphere Application Server product

```
==>rpm -qa | grep WSB
WSBSR1AA-5.0-0
WSBSR5AA-5.0-0
WSBSR6AA-5.0-0
WSBSM1AA-5.0-0
WSBAS1AA-5.0-0
WSBGK2AA-5.0-0
WSBCO1AA-5.0-0
WSBAC1AA-5.0-0
WSBAT1AA-5.0-0
WSBDT1AA-5.0-0
WSBAU1AA-5.0-0
WSBMQ1AA-5.0-0
WSBMQ2AA-5.0-0
```

```
WSBMQ3AA-5.0-0
WSBMS2AA-5.0-0
WSBMS4AA-5.0-0
WSBMS7AA-5.0-0
WSBES1AA-5.0-0
WSBES3AA-5.0-0
WSBIHAB-1.3-26
WSBPL1AA-5.0-0
WSBLA1AA-5.0-0
WSBJA1AA-5.0-0
WSBCO4AA-5.0-0
WSBCO5AA-5.0-0
WSBJD7AA-1.3-1
WSBJD5AA-1.3-1
WSBJD9AA-1.3-1
```

Manually uninstalling on Solaris platforms

Always use the uninstall program, if it exists. In some cases, such as an aborted installation, the uninstall program might not exist, or might not be complete and functional. For example, there are several valid scenarios where parts of the embedded messaging feature are not uninstalled, including:

- A message broker is still defined.
- The WebSphere Embedded Messaging Publish and Subscribe Edition (WEMPS) is not removed.
- A message queue manager is still defined.
- Java Message Service and MQ Series are not removed.

Use the following procedure to remove all remnants of WebSphere Application Server so that you can install again.

Depending on the WebSphere Application Server products you installed, uninstall the ones that you intend to uninstall in this order:

1. WebSphere Application Server Network Deployment
2. WebSphere Application Server (base product)

In the following steps:

- The default *WAS_install_root* for WebSphere Application Server is the `/opt/WebSphere/AppServer` directory.
- The default *ND_install_root* for Network Deployment is the `/opt/WebSphere/DeploymentManager` directory.

1. Type **`kill -9 <java_pid_1> <java_pid_2>...<java_pid_n>`** to ensure that no Java processes are running.
2. Halt any running MQ Series queue managers.
   a. Type **`dspmq`** to show the state of any queue managers.
   b. Type **`endmqm -i`** for each running queue manager.
3. Type **`kill -9`** ***`<amq_pid_1> <amq_pid_2>`*** **`...`** ***`<amq_pid_n>`*** to ensure that no MQ Series processes are running.
4. Run the *`ND_install_root`*`/_uninst/uninstall` program for Network Deployment, if it exists.
5. Run the *`WAS_install_root`*`/_uninst/uninstall` program for WebSphere Application Server, if it exists.
6. Search for related packages.

   Type these commands to search for related packages:

- **pkginfo | grep WS** to show packages for the base WebSphere Application Server product, the Network Deployment product, and the IBM HTTP Server product
- **pkginfo | grep wemps** to show more packages for the embedded messaging feature
- **pkginfo | grep mqm** to show packages for the embedded messaging feature, which is based on IBM MQ Series technology

The examples below show typical package names that might appear on a system with the base WebSphere Application Server product installed.

If no packages appear when using these commands, skip the next step.

7. Type **pkgrm <packagename1> <packagename2> <packagename3> ...** to remove any WebSphere Application Server-related packages.

   For example, execute this command as root to remove the embedded messaging feature from your system:

   ```
   pkgrm wemps mqjava mqm-upd04 mqm
   ```

   Reply y[es] to all prompts.

   Note: Remove the **mqm** package last, or you cannot remove any other embedded messaging feature packages. You will have to reinstall the embedded messaging feature to remove the packages.

   Alternatively, you can type these commands to search for and remove any WebSphere Application Server-related packages:

   a. **ls |grep WSB|xargs -i pkgrm -n {}** for the base WebSphere Application Server product.
   b. **ls |grep WSN|xargs -i pkgrm -n {}** for the Network Deployment product.
   c. **ls |grep WSC|xargs -i pkgrm -n {}** for the WebSphere Application Server Java client.
   d. **ls |grep wemps|xargs -i pkgrm -n {}** for the embedded messaging feature.
   e. **ls |grep mqjava|xargs -i pkgrm -n {}** for the embedded messaging feature.
   f. **ls |grep mqm|xargs -i pkgrm -n {}** for the embedded messaging feature.
8. Change directory to the `/opt/WebSphere` directory.
9. Type **rm -rf AppServer/ DeploymentManager/** to delete these WebSphere Application Server-related directories. If there are no other entries, you can remove the `WebSphere` directory.
10. Change directories to the `/opt` directory.
11. Type **rm -rf IBMHttpServer/ wemps/ mqm/** to delete the IBM HTTP Server directory, and the two embedded messaging feature directories.

When the process completes, the product is completely uninstalled. You are now ready to reinstall.

Usage scenario

Example of displaying package names beginning with mqm, for the embedded messaging feature

```
pkginfo |grep mqm

application mqm    WebSphere MQ for Sun Solaris
```

Example of displaying package names beginning with wemps, for the embedded messaging feature

```
pkginfo |grep wemps

application wemps   WebSphere Embedded Messaging Publish and Subscribe Edition
```

Example of displaying package names beginning with WSB, for the base package

```
pkginfo | grep WSB

application WSBAA          WebSphere Application Server
application WSBAAAA        Application And Assembly Tools
application WSBAC1AA       adminConsoleFilesComponent
application WSBACAA        Admin Console
application WSBADAA        Admin
application WSBAS1AA       adminScriptingFilesComponent
application WSBASAA        Admin Scripting
application WSBAT1AA       applicationAssemblyToolComponent
application WSBATAA        Application Assembly Tool
application WSBAU1AA       antUtilityComponent
application WSBAUAA        Ant Utility
application WSBCO1AA       commonFiles
application WSBCO4AA       pbwServerConfigWithMQGood
application WSBCO5AA       IsmpLauncherComponent
application WSBCOAA        commonFeature
application WSBDM1AA       DCMStdComponent
application WSBDMAA        Dynamic Cache Monitor
application WSBDT1AA       deployToolComponent
application WSBDTAA        Deploy Tool
application WSBES1AA       messagingSampleFileComponentBean
application WSBES3AA       mdbServerConfigWithMQGoodUnix
application WSBESAA        mqSeriesSamples
application WSBGK2AA       gskitUnixComponent
application WSBGK3AA       gskit4SolarisComponent
application WSBGKAA        gskitFeature
application WSBIHAA        ihsFeature
application WSBIHAB        ihsComponent
application WSBJD9A        javaCommonConfigComponent
application WSBJA1AA       javadocComponent
application WSBJAAA        Javadoc
application WSBJD4AA       javaSolarisComponent
application WSBJD7AA       javaUninstallComponent
application WSBJDAA        Java
application WSBLA1A        logAnalyzerComponent
application WSBLAAA        LogAnalyzer
application WSBMQ1AA       mqSeriesSetupFileComponent
application WSBMQ2AA       mqSeriesBinComponent
application WSBMQ3AA       mqSeriesLibFilesComponent
application WSBMQAA        MQSeries
application WSBMS2AA       mqSeriesUnixPrereqBean
application WSBMS4AA       mqSeriesUnixInstall
application WSBMS6AA       mqSeriesSunConfig
application WSBMSAA        mqSeriesServer
application WSBPL1AA       component8
application WSBPL21AA      Domino
application WSBPLAA        Plugins
application WSBPS1AA       perfServletComponent
application WSBPSAA        Performance Servlet
application WSBPTAA        PerformanceAndAnalysisTools
application WSBSM1AA       samplesComponent
application WSBSMAA        Samples
application WSBSR1AA       serverStdComponent
application WSBSR5AA       serverConfigWithSamplesComponent
application WSBSR6A        serverCommonConfigComponent
application WSBSRAA        Server
application WSBTV1AA       tivoliViewerComponent
application WSBTVAA        TivoliPerformanceViewer
```

Manually uninstalling on AIX platforms

Always use the uninstall program, if it exists. In some cases, such as an aborted installation, the uninstaller program might not exist, or might not be complete and functional. For example, there are several valid scenarios where parts of the embedded messaging feature are not uninstalled, including:

- A message broker is still defined.
- The WebSphere Embedded Messaging Publish and Subscribe Edition (WEMPS) is not removed.
- A message queue manager is still defined.
- Java Messaging Service and MQ Series are not removed.

Use the following procedure to remove all remnants of WebSphere Application Server so that you can install again.

Depending on the WebSphere Application Server products you installed, uninstall the ones that you intend to uninstall in this order:

1. WebSphere Application Server Network Deployment
2. WebSphere Application Server (base product)

In the following steps:

- The default *WAS_install_root* for WebSphere Application Server is the `/usr/WebSphere/AppServer` directory.
- The default *ND_install_root* for Network Deployment is the `/usr/WebSphere/DeploymentManager` directory.

1. Type **`kill -9`** ***`<java_pid_1> <java_pid_2>...<java_pid_n>`*** to ensure no Java processes are running.
2. Halt any running MQ queue managers.
   a. Type **`dspmq`** to show the state of any queue managers.
   b. Type **`endmqm -i`** for each running queue manager.
3. Type **`kill -9`** ***`<amq_pid_1> <amq_pid_2> ... <amq_pid_n>`*** to ensure that no MQ processes are running.
4. Run the *`ND_install_root`*`/_uninst/uninstall` program for Network Deployment, if it exists.
5. Run the *`WAS_install_root`*`/_uninst/uninstall` program for WebSphere Application Server, if it exists.
6. Search for other related packages.

   Either use smit to remove packages, or search for and remove packages manually:

   - Type **`smit`** to clean up remnants.
     a. Click **Software Installation and Maintenance**.
     b. Click **Software Maintenance and Utilities**.
     c. Click **Remove Installed Software**.
     d. Click **LIST** by Software name.

   e. Search for packages that contain the words **IBM** or **WS** to identify all WebSphere Application Server-related packages, including those that belong to **IBM HTTP Server** and the embedded messaging feature, which is based on **IBM MQ Series** technology.
   f. Change the **PREVIEW ONLY** option to **NO**.
   g. Click **OK**.
- Manually search for, and remove related packages. Type these commands to search for, and remove related packages:
   a. Search for related packages:
      1) **`lslpp -l | grep WS`** to show packages for the base WebSphere Application Server product, the Network Deployment product, and the IBM HTTP Server product
      2) **`lslpp -l | grep mqm`** to show packages for the embedded messaging feature, which is based on IBM MQ Series technology
      3) **`lslpp -l | grep wemps`** to show more packages for the embedded messaging feature.

         You can also execute these commands as root to remove the embedded messaging feature from your system:
         ```
         installp -u wemps mqjava mqm
         ```
         Reply y[es] to all prompts.

      The examples below show typical package names that might appear on a system with the base WebSphere Application Server product.

      If no packages appear when using these commands, skip the next step.
   b. Type **`smitty remove <packagename1> <packagename2> <packagename3> ...`** to remove any WebSphere Application Server-related packages.

7. Change directory to the `/opt/WebSphere` directory.
8. Type **`rm -rf AppServer/ DeploymentManager/`** to delete these WebSphere Application Server-related directories. If there are no other entries, you can remove the `WebSphere` directory.
9. Change directory to the `/usr/opt` directory.
10. Type **`rm -rf wemps/`** to delete the embedded messaging feature directory.
11. Change directory to the `/usr` directory.
12. Type **`rm -rf IBMHttpServer/ mqm/`** to delete the IBM HTTP Server directory, and the embedded messaging feature directory.

Results

When the process completes, the product is completely uninstalled. You are now ready to reinstall.

Usage scenario

Example of displaying package names beginning with mqm, for the embedded messaging feature

```
==>lslpp -l | grep mqm

  mqm.base.runtime           5.3.0.1  COMMITTED  WebSphere MQ Runtime for
  mqm.base.sdk               5.3.0.1  COMMITTED  WebSphere MQ Base Kit for
  mqm.client.rte             5.3.0.1  COMMITTED  WebSphere MQ Client for AIX
  mqm.java.rte               5.3.0.1  COMMITTED  WebSphere MQ Java Client and
```

```
  mqm.msg.De_DE             5.3.0.1  COMMITTED  WebSphere MQ Messages - German
  mqm.msg.Es_ES             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.msg.Fr_FR             5.3.0.1  COMMITTED  WebSphere MQ Messages - French
  mqm.msg.It_IT             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.msg.Ja_JP             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.msg.Zh_CN             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.msg.Zh_TW             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.msg.de_DE             5.3.0.1  COMMITTED  WebSphere MQ Messages - German
  mqm.msg.en_US             5.3.0.1  COMMITTED  WebSphere MQ Messages - U.S.
  mqm.msg.es_ES             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.msg.fr_FR             5.3.0.1  COMMITTED  WebSphere MQ Messages - French
  mqm.msg.it_IT             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.msg.ja_JP             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.msg.ko_KR             5.3.0.1  COMMITTED  WebSphere MQ Messages - Korean
  mqm.msg.pt_BR             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.msg.zh_CN             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.msg.zh_TW             5.3.0.1  COMMITTED  WebSphere MQ Messages -
  mqm.server.rte            5.3.0.1  COMMITTED  WebSphere MQ Server
```

Example of displaying package names beginning with wemps, for the embedded messaging feature

```
==>lslpp -l | grep wemps
```

```
wemps.base.runtime        2.1.0.0  COMMITTED  WebSphere Embedded Messaging
```

Example of displaying package names beginning with WS, for WebSphere Application Server-related products

```
==>lslpp -l | grep WS
```

```
  WSBAA                     5.0.0.0  COMMITTED  ISMP installed entry
  WSBAAAA                   5.0.0.0  COMMITTED  Installs tools for assembly
  WSBAC1AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBACAA                   5.0.0.0  COMMITTED  Includes adminconsole.ear, the
  WSBADAA                   5.0.0.0  COMMITTED  Installs the Administrative
  WSBAS1AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBASAA                   5.0.0.0  COMMITTED  Includes wsadmin, the
  WSBAT1AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBATAA                   5.0.0.0  COMMITTED  Includes a GUI-based tool for
  WSBAU1AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBAUAA                   5.0.0.0  COMMITTED  Includes Apache Ant, a
  WSBCO1AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBCO4AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBCO5AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBCOAA                   5.0.0.0  COMMITTED  ISMP installed entry
  WSBDT1AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBDTAA                   5.0.0.0  COMMITTED  Includes a command-line
  WSBES1AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBES3AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBESAA                   5.0.0.0  COMMITTED  Includes samples for
  WSBGK2AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBGKAA                   5.0.0.0  COMMITTED  ISMP installed entry
  WSBIHAA                  1.3.26.0  COMMITTED  Installs an Apache-powered Web
  WSBIHAB                  1.3.26.0  COMMITTED  ISMP installed entry
  WSBJA1AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBJAAA                   5.0.0.0  COMMITTED  Installs WebSphere public
  WSBJD3AA                  1.3.1.0  COMMITTED  ISMP installed entry
  WSBJD7AA                  1.3.1.0  COMMITTED  ISMP installed entry
  WSBJD9AA                  1.3.1.0  COMMITTED  ISMP installed entry
  WSBJDAA                   1.3.1.0  COMMITTED  ISMP installed entry
  WSBLA1AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBLAAA                   5.0.0.0  COMMITTED  Includes a graphical utility
  WSBMQ1AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBMQ2AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBMQ3AA                  5.0.0.0  COMMITTED  ISMP installed entry
  WSBMQAA                   5.0.0.0  COMMITTED  Installs Java Messaging
```

```
WSBMS2AA                5.0.0.0  COMMITTED  ISMP installed entry
WSBMS4AA                5.0.0.0  COMMITTED  ISMP installed entry
WSBMS5AA                5.0.0.0  COMMITTED  ISMP installed entry
WSBMSAA                 5.0.0.0  COMMITTED  Includes JMS
WSBPL1AA                5.0.0.0  COMMITTED  ISMP installed entry
WSBPLAA                 5.0.0.0  COMMITTED  Installs plugins to configure
WSBPTAA                 5.0.0.0  COMMITTED  Installs tools for performance
WSBSM1AA                5.0.0.0  COMMITTED  ISMP installed entry
WSBSMAA                 5.0.0.0  COMMITTED  Includes samples, including
WSBSR1AA                5.0.0.0  COMMITTED  ISMP installed entry
WSBSR5AA                5.0.0.0  COMMITTED  ISMP installed entry
WSBSR6AA                5.0.0.0  COMMITTED  ISMP installed entry
WSBSRAA                 5.0.0.0  COMMITTED  Installs the application
```

Installation: Resources for learning

Use the following links to find relevant supplemental information about installation. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to the IBM WebSphere Application Server product, but is useful all or in part for understanding the product. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

This section contains the following topics:
- Planning, business scenarios, and IT architecture
- Programming model and decisions
- Programming instructions and examples
- Programming specifications
- Administration
- Support

Planning, business scenarios, and IT architecture

- **IBM WebSphere Application Server supported hardware, software, and APIs** at http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

  The official site for determining product prerequisites for hardware, software and APIs for all WebSphere Application Server products.
- **IBM WebSphere Developer Domain** at http://www7b.software.ibm.com/wsdd/

  The home of technical information for developers working with WebSphere products. You can download WebSphere software, take a fast path to developer domain zones, such as VisualAge Java or WebSphere Application Server, learn about WebSphere products through a newcomers page, tutorials, technology previews, training, and Redbooks, get answers to questions about WebSphere products, and join the WebSphere community, where you can keep up with the latest developments and technical papers.
- **IBM WebSphere Application Server library and InfoCenters Web site** at http://www-3.ibm.com/software/webservers/appserv/infocenter.html

  The IBM WebSphere Application Server Library Web site contains links to all WebSphere Application Server InfoCenters, for all versions. It also lets you access each InfoCenter in your native language.

- **IBM WebSphere Application Server home page** at http://www.ibm.com/software/webservers/appserv/

  The IBM WebSphere Application Server home page contains useful information, including support links and downloads for e-fixes, tools, and trials.
- **IBM WebSphere software platform home page** at http://www.ibm.com/websphere

  The IBM WebSphere software platform home page introduces WebSphere products and describes how companies can easily transform to an e-business, with software that can grow as fast as the business it supports.
- **Migrating to WebSphere V5.0: An End-to-End Migration Guide, SG24-6910-00** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246910.html

  This IBM Redbook is the definitive migration guide for migrating earlier versions of WebSphere Application Server to Version 5. Read this book to formulate an optimal migration strategy.
- **Just announced: IBM WebSphere Application Server Version 5!** at http://www-3.ibm.com/software/info1/ websphere/ index.jsp?tab=products/ appserv

  The IBM WebSphere Application Server Version 5 product announcement.
- **What's new in IBM WebSphere Application Server Version 5?** at http://www-3.ibm.com/software/info1/ websphere/ index.jsp?tab=products/ appserv_whatsnew

  The official list of new features at the What's new in IBM WebSphere Application Server Version 5? Web page.
- **IBM WebSphere Application Server Version 5 fact sheet** at http://www-3.ibm.com/software/webservers/ appserv/v5/ appsvrv5factsheet.pdf

  This fact sheet describes IBM WebSphere Application Server Version 5.
- **The power of Edge of Network technology in IBM WebSphere Application Server Version 5** at http://www-3.ibm.com/software/ info1/ websphere/ index.jsp?tab=products/ appserv_edge

  A description of WebSphere Application Server Edge Components Version 5.
- **WebSphere Application Server - Express, V5** at http://www-3.ibm.com/software/info1/ websphere/index.jsp?tab=products/ appserv_express

  A description of WebSphere Application Server Express, Version 5.
- **WebSphere Application Server, Version 5** at http://www-3.ibm.com/software/info1/ websphere/index.jsp?tab=products/ appserv_v5

  A description of the base product, WebSphere Application Server, Version 5.
- **WebSphere Application Server Enterprise, Version 5** at http://www-3.ibm.com/software/info1/websphere/index.jsp?tab=products/ appserv_enterprise

  A description of WebSphere Application Server Enterprise, Version 5.
- **IBM WebSphere Application Server Version 5 news release** at http://www-3.ibm.com/events/ibmdeveloperworkslive/news_bisoftware.html

  A new release about IBM WebSphere Application Server, Version 5 on the IBM developerWorks Live! site.

- **IBM WebSphere Application Server for z/OS, Version 5** at http://www-3.ibm.com/software/info1/websphere/index.jsp?tab=products/appserv_zos

  A description of WebSphere Application Server for z/OS, Version 5.
- **InfoCenter for WebSphere Application Server Edge components** at http://www-3.ibm.com/software/webservers/appserv/ecinfocenter.html

  The InfoCenter for WebSphere Application Server Edge components contains complete documentation for the Caching Proxy and the Load Balancer in these PDF online books, *WebSphere Application Server Concepts, Planning, and Installation for Edge Components*, the *WebSphere Application Server Caching Proxy Administration Guide*, and the *WebSphere Application Server Programming Guide for Edge Components*.
- **IBM's Web Services architecture debuts** at http://www.ibm.com/developerworks/webservices/library/w-int.html?dwzone=webservices

  Introducing IBM Web Services, a distributed software architecture of service components. This brief overview and in-depth interview on IBM developerWorks covers the fundamental concepts of Web Services architecture and what they mean for developers. The interview with Rod Smith, Vice President of Emerging Technologies at IBM, explores which types of developers Web Services targets, how Web services reduce development time, what developers can do with Web Services now, and takes a glance at the economics of dynamically discoverable services.
- **developerWorks: Patterns for e-business: Redbooks listing** at http://www.ibm.com/developerworks/patterns/library/index.html#redbooks

  This Web page lists links to pattern resources under these categories:
  - Current patterns Redbooks
  - Superseded patterns Redbooks (valid for back-level product versions)
  - Independent analyst reports
  - Patterns CD order offer
  - Back-level version patterns Web site (zip downloads and old Flash tutorial)
  - Customer references
  - White papers
  - Multimedia presentations and screen cams
  - Webcasts
  - Patterns development kit
  - WebSphere technical exchange presentations
- **developerWorks: IBM Patterns for e-business** at http://www.ibm.com/developerworks/patterns/index.html

  The IBM developerWorks site is the source for IBM patterns for e-business, a set of tested, reusable intellectual assets that you can use to design and implement your e-business network and architecture!
- **Self-Service: Select application pattern** at http://www.ibm.com/developerworks/patterns/u2b/select-application-topology.html

  This Web page describes the self-service business pattern, also known as the User-to-Business or U2B pattern. The self-service e-business pattern captures the essence of direct interactions between interested parties and an e-business.

Interested parties include customers, business partners, stakeholders, employees, and all other individuals with whom the business intends to interact.

- **Self-Service: Standalone single channel application pattern: Run-time patterns** at http://www.ibm.com/developerworks/patterns/u2b/at1-runtime.html

  This Web page describes alternative run-time solution patterns for the self-service business pattern. These run-time patterns are important concepts that any e-business designer should become familiar with.
- **Patterns for e-business: A Strategy for Reuse** at http://www.mcpressonline.com/ibmpress/5206.htm

  Get an inside look at how successful businesses build their e-business architectures. In this book, four IBM e-business experts capture years of experience into easy-to-follow guidelines. Deliberately focusing on Business patterns, integration patterns, and application patterns, the authors share with you proven architectural patterns that can help get you up and running quickly, while at the same time reducing your risks. Because today's economy demands that e-business initiatives emphasize profitability and return on investment, the authors also offer guidance on methods to minimize cost, yet ensure quality.
- **Patterns and Web Services** at http://www.ibm.com/developerworks/patterns/guidelines/web-services.pdf

  Patterns architects have reviewed the impact emerging Web Services technologies have on each of the asset layers of the Patterns for e-business designs. Their findings are summarized in this new White paper, in PDF Format.
- **developerWorks: Facilitating the application development process using the IBM Patterns for e-business** at http://www.ibm.com/developerworks/patterns/guidelines/lord.pdf

  This is the most recent White paper from John Lord, a well known IBM consulting IT architect. The paper analyzes the successful use of the Patterns for e-business in an application development scenario.
- **Self-Service Patterns using WebSphere Application Server, Version 4.0, SG24-6175-00** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246175.html

  This Redbook discusses the Standalone Single Channel application pattern of the Self-Service business patterns. This application pattern describes a situation where you are building an application that has no need to connect to backend or legacy data. The Directly Integrated Single Channel application pattern extends this discussion to describe the situation where you need to access existing data on legacy or third-party systems. Although we do not implement the Directly Integrated Single Channel application pattern in this project, the discussions here are relevant.
- **Patterns: Connecting Self-Service Applications to the Enterprise, SG24-6572-00** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246572.html

  This Redbook discusses the Self-Service::Directly Integrated Single Channel application pattern, which covers Web applications needing one or more point-to-point connections with backend applications.
- **Self-Service Applications using IBM WebSphere Application Server V4.0 and IBM MQSeries Integrator, SG24-6160-01** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246160.html

This Redbook focuses on the task of designing and implementing a self-service application using the Router application pattern and the Decomposition application pattern, as defined by the IBM Patterns for e-business.

The Router application pattern provides intelligent routing from multiple clients to multiple backend applications using a hub-and-spoke architecture. The interaction between the user and the backend application is a one-to-one relation, meaning the user interacts with applications one at a time. The primary business logic resides in the backend tier. This book shows how to use IBM MQSeries Integrator and IBM WebSphere Application Server to implement a router type application.

The Decomposition application pattern expands on the router pattern, providing all the features and functions of that pattern and adding recomposition/decomposition capability. It provides the ability to take a user request and decompose it into multiple requests to route to multiple backend applications. The responses are recomposed into a single response for the user. This action moves some of the business logic into the decomposition tier, but the primary business logic still resides in the back-end application tier. The decomposition and recomposition functions are illustrated in this book using IBM WebSphere Application Server, IBM MQSeries Integrator and the IBM MQSI Aggregator Plug-In. The JMS listener provided by WebSphere Application Server Enterprise Services is also illustrated in this example.

- **Applying the Patterns for e-business to Domino and WebSphere Scenarios, SG24-6255-00** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246255.html

  This Redbook describes Application Integration patterns, and how they form Composite Patterns, together with one or more of the other Patterns for e-business. It looks at run-time patterns for Domino and WebSphere Application Server integration, and identifies Composite Patterns in action. Some of the patterns include Lotus Sametime and Tivoli Policy Director.

  A major part of the book describes three real-life scenarios where Patterns for e-business are applied, with Domino and WebSphere Application Server as part of the run-time topology. Starting from the business requirements phase, the book identifies and applies Business, Application, and Run-time patterns to get to the final run-time topology. It starts with a simple scenario, which becomes increasingly complex in later scenarios. It discusses technology options, as well as design and development guidelines.

- **User-to-Business Pattern Using WebSphere Personalization Patterns for e-business Series, SG24-6213-00** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246213.html

  This Redbook describes what was once referred to as the User-to-Business Topology 7 Personalization Pattern. At the time the book was written, the pattern was an emerging pattern. It focuses on direct interactions between users and a business. The pattern helps guide the design of systems with a consolidated customer-centric view that you can exploit for sophisticated personalization and cross-selling opportunities.

  This Redbook provides examples and guidelines for the User-to-Business Topology 7 Personalization Pattern. It shows how the pattern works and documents the tasks required to build an example of the pattern.

- **Mobile Applications with IBM WebSphere Everyplace Access Design and Development, SG24-6259-00** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246259.html

This Redbook provides application designers and developers with a broad overview of mobile e-business application design and development using the WebSphere Everyplace Access V1R1 offering.

The book gives an overview of the Patterns for e-business and shows how to use the Patterns in the mobile e-business environment. It also discusses the design and development guidelines for mobile e-business applications using the products bundled in the WebSphere Studio and Visual Age for Java offerings.

This book provides detailed information about the Sample application, by discussing scenarios and implementing mobile applications exercising different techniques for several type of clients. It also provides detailed instructions for setting up the development and run-time environment for WebSphere Application Server, WebSphere Transcoding Publisher and WebSphere Voice Server together with the Sample shipped with the Redbook.

- **Access Integration Pattern using IBM WebSphere Portal Server, SG24-6267-00** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246267.html

  This Redbook describes the Access Integration pattern, which is an emerging pattern that describes services and components commonly required to provide users with consistent, seamless, device-independent access to relevant applications and information.

  This Redbook provides an example and guidelines for the Access Integration pattern. It shows how the Pattern works and documents the tasks required to build an example.

- **WebSphere Commerce Suite V5.1 for iSeries, Implementation and Deployment Guide, REDP0159** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpaperAbstracts/redp0159.html

  This Redpaper describes the great benefit of the IBM Framework for e-business, to develop applications on the Windows platform and then deploy the application to one of the many supported WebSphere Application Server platforms, without changing the application.

  This Redpaper introduces the Patterns for e-business and the Electronic Commerce composite pattern used for building e-commerce Web sites. The focus of the Redpaper is on the iSeries unique implementation, deployment and development considerations when using IBM WebSphere Commerce Suite V5.1, Pro Edition for iSeries.

  It includes detailed procedures for implementing WebSphere Commerce Suite V5.1, Pro Edition for iSeries in single-tier and multiple tier run-time environments. Advanced configuration instructions are provided for integrating WebSphere Payment Manager, enabling Secure Socket Layer (SSL) for the HTTP Server, and configuring a secure VPN connection for Open Servlet Engine (OSE) Remote. Once the run-time environment is configured, there is a detailed description of the steps necessary to deploy a WebSphere Commerce Suite store to a WebSphere Commerce Suite V5.1, Pro Edition for iSeries run-time environment.

- **e-commerce Patterns for z/Linux Using WebSphere Commerce Suite V5.1 Patterns for e-business series, REDP0411** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpaperAbstracts/redp0411.html

  This Redpaper describes the installation, configuration and customization of IBM WebSphere Commerce Suite Pro Edition for Linux for e-server z900 and S/390. It is intended for both technicians who need to install and administer Commerce

Suite, and for developers who need to design and customize e-commerce sites for deployment on IBM WebSphere Commerce Suite Pro Edition for Linux for e-server z900 and S/390.

This Redpaper is part of the Patterns for e-business series and reuses information developed in the Redbook B2C e-commerce Composite pattern using WebSphere Commerce Suite V5.1, SG24-6180.

- **Integrating WebSphere Commerce Suite With a backend Order Management Application, REDP0514** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpaperAbstracts/redp0514.html

  This Redpaper describes a scenario that presents a fictitious manufacturing company called MANCO, which produces bicycle parts and accessories. MANCO uses the J.D. Edwards OneWorld application on an iSeries server for the Enterprise Resource Planning (ERP) system. MANCO wanted to take advantage of the Internet global connectivity to give its business customers a new way to buy products and to check order status while augmenting the manual sales processes with direct electronic sales to its business customers. MANCO requirements best fit the user-to-online buying business pattern, which is a subset of the user-to-business pattern.

- **Connect for iSeries with WebSphere Commerce Suite: BtoB Enabling a WebSphere Commerce Suite Web Site, REDP0127** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpaperAbstracts/redp0127.html

  This Redpaper describes another MANCO scenario, from another iSeries development team in the IBM Rochester laboratory.

  This scenario presents a fictitious manufacturing company called MANCO, which produces bicycle parts and accessories.

  The e-marketplace has emerged as a means of connecting buyers and suppliers. Suppliers look for opportunities to grow their business and expand their customer base. An e-marketplace provides that ability. Many suppliers have already invested in e-commerce solutions and would like to leverage their current solution to enter into the e-marketplace arena.

  This Redpaper details the migration of the existing MANCO e-Commerce Web site to an e-marketplace-enabled Web site, by utilizing the WebSphere Commerce Suite and Connect for iSeries products.

- **e-Marketplace Pattern using WebSphere Commerce Suite, MarketPlace Edition Patterns for e-business Series, SG24-6158-00** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246158.html

  This Redbook describes the Business to Business e-Marketplace Pattern, which supports the development of e-Marketplace hub applications that bring multiple buyers and sellers together for efficient electronic trading of goods and services. Subsets of the application topologies for the Business to Business e-Marketplace Pattern are used to describe different parts of the full marketplace topology, and they represent increasing levels of complexity, functionality and integration in the topology, ranging from a simple e-Marketplace to a fully integrated e-Marketplace.

- **Business-to-Business Integration Using MQSeries and MQSI, Patterns for e-business Series, SG24-6010-00** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246010.html

  This Redbook describes what used to be known as Business-to-Business Integration patterns two and three, which form the basis for many complex and more fully functioned B2B patterns. It is relevant to all enterprises dealing with partner integration issues over the Internet.

Application topology 2 describes a scenario in which messages are being passed between two enterprise applications and no routing is performed. Topology 3 extends topology 2 to describe the scenario where routing is required for multiple cross enterprise applications to communicate.

- **Business Process Management using MQSeries and Partner Agreement Manager, SG24-6166-00** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246166.html

  This Redbook describes business process management.

  Business process management is the answer for addressing the business issues of the economic world. The velocity of change, driven by the dynamics of commerce and e-business, force you to have an IT system that is ready to change rapidly, and continually. Another issue is the integration of the Internet within business processes and more general multichannel delivery. Companies must be able to provide a consistent service across all channels. To achieve this, you need to have an integrated business view. Business Process Management is the externalization and formalization of knowledge and expertise within applications and minds. That externalization makes it possible to stay in control of your business.

  This Redbook introduces the concepts of business process management and its relationship with business-to-business technologies. In the second part of the book, it explores an example of a business process built in MQSeries Workflow. The third part of the book extends this intra-enterprise business process to include collaboration with external companies using WebSphere Business-to-Business Partner Agreement Manager.

  The final part of the book takes one step back and looks in more general terms at the patterns of developing and deploying business-to-business solutions.

- **Design for Scalability - An Update** at http://www7b.software.ibm.com/wsdd/library/techarticles/hvws/scalability.html

  This White paper is from the IBM High Volume Web Sites team. The White paper describes component selection and management techniques you can use to make your Web site ready to adapt to increasing traffic. These techniques are the product of IBM experiences while working with customers seeking to improve the performance and availability of some of the largest Web sites in the world.

  Abstract: Optimizing for scalability remains a significant challenge for e-businesses as they balance the demands for availability, reliability, security, and high performance. Vendors are responding with infrastructure options and supporting hardware and software platforms that address these requirements. This update identifies current products and emerging trends that are most likely to improve the scalability of your e-business infrastructure.

- **IBM WebSphere V4.0 Advanced Edition Handbook** at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246176.html

  This Redbook describes base application topologies and product mappings for WebSphere Application Server. Refer to the **IBM Redbooks** Web site at http://www.redbooks.ibm.com/ for the latest update.

- **User centered design (UCD) for different project types, part 1** at http://www-106.ibm.com/developerworks/usability/library/us-ucd/

  This Web page is the first of two articles posted to the IBM develperWorks domain that describes useful application design activities for different types of projects.

- **User centered design (UCD) for different project types, part 2** at http://www.ibm.com/developerworks/library/us-ucd2/index.html?dwzone=usability

  This Web page is the latest of two articles that describes design activities that IBM scientists have found most useful in various types of projects. This article defines user interface design elements, including the design prototype, use case model, and design specification document.

Programming model and decisions

- **Designing e-business Solutions for Performance** at http://www.ibm.com/developerworks/patterns/ebusiness-performance-customer-v2.pdf

  This White paper describes how the design or implementation of an e-business application can affect performance.
- **Managing Web Site Performance** at http://www.ibm.com/developerworks/patterns/guidelines/ HTTP_Session_Best_Practice.pdf

  This White paper contains tips and techniques for developers building applications that use session persistence. It also helps administrators to tune the WebSphere Application Server product appropriately for these applications.

Programming instructions and examples

- **IBM developerWorks** at http://www.ibm.com/developerworks/

  IBM developerWorks contains many excellent resources for developers, including tutorials on Web development-related topics. There is an excellent tutorial on the JDBC API.
- **IBM Redbooks** at http://www.redbooks.ibm.com/

  The IBM Redbooks site contains many WebSphere Application Server related documents.
- **Servlets and JavaServer Pages - A Tutorial** at http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/

  Tutorial from the author of Core Servlets and JavaServer Pages.

Programming specifications

- **J2EE information** at http://java.sun.com

  For more information about J2EE specifications, visit the Sun site.
- **sun.net.inetaddr.ttl property** at http://java.sun.com/j2se/1.4/docs/guide/net/properties.html

  The following Java 2 SDK, Standard Edition 1.4 Web site describes the private sun.net.inetaddr.ttl property, which also works in Java 2 SDK, Standard Edition 1.3.
- **java.net.URLConnection class** at http://java.sun.com/j2se/1.4.1/jcp/beta/

  The *Networking* section of this Java 2 SDK, Standard Edition 1.4 Web site describes a change in the behavior of the java.net.URLConnection class.

Administration

- **Best Practices Zone on WSDD** at http://www7b.boulder.ibm.com/wsdd/zones/bp/

The WebSphere Best Practices Zone is a collection of best practices for administering WebSphere Application Server. Over time, the zone is intended to grow to include best practices for using other WebSphere software products, and to cover more topics. Use the feedback mechanism to submit your best practice suggestions.

- **The IBM Glossary of Computing Terms** at http://www.ibm.com/ibm/terminology/goc/gocmain.htm

  This glossary defines technical terms used in many IBM products. It is not a comprehensive resource of all IBM computing terms. This resource is provided for information purposes only and is updated periodically. IBM takes no responsibility for the accuracy of the information it contains.

Support

- **AIX Fix Distribution Service Web site** at http://techsupport.services.ibm.com/rs6k/fixdb.html

  A Web facility for downloading AIX Version 4 and AIX Version 3 fixes, with a limited search engine designed with the assumption that you know what fix you need. If you do not know what fix you need, there is a pointer at the Web site to the APAR Database Facility. You can also contact your authorized IBM business partner or IBM Support Center.
- **Ten Steps to Getting Support for WebSphere Application Server** at http://www7b.boulder.ibm.com/wsdd/support/appserver_support.html

  If you are new to a product, you might have difficulty finding all the information you need. And if you come across a problem, where do you go for help? Whether you are a new user looking for introductory information, or an experienced user looking for a workaround for a specific defect, you can benefit immediately from extensive Web-based support from IBM. It enables you to download FixPaks, search on keywords, look up FAQs, Hints and Tips, and so forth. Always use this Web resource before contacting IBM Support directly.
- **WebSphere Application Server Support page** at http://www-3.ibm.com/software/webservers/appserv/support.html

  Take advantage of the Web-based Support and Service resources from IBM to quickly find answers to your technical questions. You can easily access this extensive Web-based support through the IBM Software Support portal at URL `http://www-3.ibm.com/software/support/` and search by product category, or by product name. For example, if you are experiencing problems specific to WebSphere Application Server, click **WebSphere Application Server** in the product list. The WebSphere Application Server Support page appears.

Chapter 4. Quickly deploying Web components - Try it out!

Here is a quick way to deploy Web components, such as servlets and JSP files. This is not recommended as an official development method. It is provided so that you can sample the product functionality.

In summary, deploy Web components quickly by dropping the individual files into the directory structure of the default application installed by the product. This procedure relies on the Invoker servlet provided by the product. This servlet, enabled by default, lets you access deployed servlets by classname.

For *recommended* methods of developing and deploying Web application components, see the *Using Web applications* topic (tweb_aovr) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

1. If deploying a servlet, first compile your servlet.
2. Copy the servlet or JSP class file into the directory of the default application.

   A servlet should be placed in the directory (split for publication):

   ```
   install_root/
      installedApps/
         cell_name/
            DefaultApplication.ear/
               DefaultApplication.war/
                  WEB-INF/
                    classes
   ```

   If your servlet has a package statement, then create a subdirectory in the above directory for each level in your package statement.

   A JSP file should be placed in the directory:

   ```
   install_root/
      installedApps/
         cell_name/
            DefaultApplication.ear/
               DefaultApplication.war
   ```
3. Open a browser window and request your servlet or JSP file.

   The URL is:

   ```
    http://your_host_name:9080/servlet/class_name
   ```

   where *class_name* is the Java class implementing the servlet or JSP file.

Chapter 5. Samples Gallery

The Samples Gallery available with the Application Server offers a set of samples that demonstrate common Web application tasks.

The Samples Gallery includes 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.
- The message-driven beans Samples demonstrate message-driven beans receiving messages from the Point-to-Point and Publish Subscribe messaging models. It also demonstrates Java Message Service (JMS) inside the client container.

Finding the Samples Gallery. Once the Samples are installed on your local machine, they are available to try out. Locate them at http://localhost:9080/WSsamples/. The default port is 9080. If you do not find the Samples on your localhost, confirm their installation and the port number for the internal HTTP server. On Windows platforms, you can also find the Samples by clicking **Start > Programs > IBM WebSphere > Application Server v5.0 > Samples Gallery**.

Client Samples. A separate Samples Gallery is available for the client Samples. To view these Samples, install the WebSphere Application Server client. The Client Samples Gallery demonstrates the following:

- J2EE application client.
- Java thin client.
- Applet client.
- ActiveX to EJB Bridge client.
- CORBA C++ SDK Client.

Code Examples. In addition to the Samples in the Samples Gallery, you can find other code examples in the InfoCenter by clicking **Quick reference > Examples** in the InfoCenter navigation.

Note: The Samples are for demonstration purposes only. The code provided is not intended to run in a secured production environment. The Samples support Java 2 Security, therefore the Samples implement policy-based access control that checks for permissions on protected system resources, such as file I/O. The Samples do not support global security, which is described in the *Global security settings* topic (usec_rgsp) in the IBM WebSphere Application Server, Version 5 InfoCenter, which is available at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.

The Samples are not supported in a multi-server, clustered environment. Many of the Samples use Cloudscape as a persistent data store on the server. Only one instance of Cloudscape is supported per Java Virtual Machine (JVM). As a result, the second server in the node will fail to start the Sample applications, because an instance of Cloudscape has already been created with the first server in the node.

Additional WebSphere Application Server Samples are available on the IBM WebSphere Developer Domain, which is available from the IBM WebSphere Application Server, Version 5 InfoCenter at http://www-3.ibm.com/software/webservers/appserv/infocenter.html.