



## Setting up the application serving environment

**Note**

Before using this information, be sure to read the general information under “Notices” on page 169.

**Compilation date: December 2, 2004**

**© Copyright International Business Machines Corporation 2004. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>How to send your comments</b> . . . . .	vii
<b>Chapter 1. Overview for setting up application serving environments</b> . . . . .	1
Setting up WebSphere Application Server products . . . . .	1
Introduction: System administration . . . . .	2
Introduction: Administrative console . . . . .	2
Introduction: Administrative scripting (wsadmin) . . . . .	2
Introduction: Administrative commands . . . . .	3
Introduction: Administrative programs. . . . .	3
Introduction: Administrative configuration data . . . . .	4
Introduction: Servers . . . . .	4
Introduction: Application servers . . . . .	4
Introduction: Web servers . . . . .	5
Introduction: Environment . . . . .	6
<b>Chapter 2. How do I administer applications and their environments?</b> . . . . .	7
<b>Chapter 3. Setting up the application serving environment</b> . . . . .	9
<b>Chapter 4. Planning the installation (diagrams)</b> . . . . .	11
Planning to install WebSphere Application Server. . . . .	12
Planning to install Web server plug-ins . . . . .	16
Planning to install WebSphere Application Server Clients . . . . .	19
Planning to create application server environments . . . . .	20
Queuing network. . . . .	20
Queuing and clustering . . . . .	21
Queue configuration tips . . . . .	22
<b>Chapter 5. Configuring the product after installation</b> . . . . .	25
firststeps command. . . . .	25
Using the Profile creation wizard . . . . .	28
Using the Profile creation wizard to create an application server . . . . .	30
Deleting a profile. . . . .	34
wasprofile command . . . . .	34
Introduction to terms that describe Version 6 profiles . . . . .	35
Location of the command file . . . . .	36
Logging . . . . .	37
Required disk space . . . . .	37
Concurrent profile creation . . . . .	37
Entering lengthy commands on more than one line . . . . .	37
wasprofile.sh command syntax . . . . .	37
wasprofile.bat command syntax . . . . .	38
Parameters. . . . .	39
Use case scenarios. . . . .	41
Using the installation verification test . . . . .	45
ivt command . . . . .	46
<b>Chapter 6. Configuring ports</b> . . . . .	49
Port number settings in WebSphere Application Server versions . . . . .	49
<b>Chapter 7. Communicating with Web servers</b> . . . . .	53
Web server plug-in properties settings . . . . .	55
Plug-in log file name . . . . .	55

Plug-in installation location . . . . .	55
Plug-in configuration file name. . . . .	56
Automatically generate plug-in configuration file . . . . .	56
Automatically propagate plug-in configuration file . . . . .	56
Ignore DNS failures during Web server startup. . . . .	57
Refresh configuration interval . . . . .	57
Plug-in logging . . . . .	57
Web server plug-in request and response optimization properties settings. . . . .	58
Web server plug-in caching properties settings. . . . .	60
Web server plug-in request routing properties settings . . . . .	60
Web server plug-in custom properties . . . . .	62
Web server plug-in configuration service properties settings . . . . .	63
Enable automated Web server configuration processing . . . . .	63
Application Server property settings for a Web server plug-in . . . . .	63
Server role . . . . .	63
Connect timeout . . . . .	63
Maximum number of connections that can be handled by the Application Server . . . . .	64
Use extended handshake to check whether Application Server is running . . . . .	64
Send the header "100 Continue" before sending the request content . . . . .	64
Web server plug-in configuration properties . . . . .	65
Web server plug-in connections . . . . .	67
Web server plug-in remote user information processing . . . . .	68
Web server plug-ins . . . . .	68
Checking your IBM HTTP Server version. . . . .	68
Web server tuning parameters. . . . .	69
Gskit install images files . . . . .	73
Plug-ins: Resources for learning . . . . .	73
Web server plug-in tuning tips . . . . .	73
Tuning Web servers . . . . .	74
<b>Chapter 8. Setting up the administrative architecture . . . . .</b>	<b>75</b>
Administration service settings. . . . .	75
Standalone . . . . .	75
Preferred Connector . . . . .	75
Extension MBean Providers collection . . . . .	75
Extension MBean collection. . . . .	76
Java Management Extensions connector properties . . . . .	76
Java Management Extensions connectors . . . . .	81
Repository service settings . . . . .	81
Administrative agents: Resources for learning . . . . .	82
<b>Chapter 9. Configuring the environment . . . . .</b>	<b>83</b>
Virtual hosts . . . . .	83
Why you would use virtual hosting . . . . .	83
The default virtual host (default_host) . . . . .	84
How requests map to virtual host aliases. . . . .	84
Configuring virtual hosts . . . . .	86
Virtual host collection . . . . .	86
Variables . . . . .	89
Configuring WebSphere variables . . . . .	90
WebSphere variables collection . . . . .	91
IBM Toolbox for Java JDBC driver . . . . .	92
Configure and use the jt400.jar file . . . . .	93
Shared library files . . . . .	93
Managing shared libraries . . . . .	93
Creating shared libraries . . . . .	94

Shared library collection . . . . .	94
Associating shared libraries with applications . . . . .	95
Associating shared libraries with servers . . . . .	96
Installed optional packages . . . . .	96
Using installed optional packages . . . . .	97
Library reference collection . . . . .	99
Environment: Resources for learning . . . . .	100
<b>Chapter 10. Working with server configuration files . . . . .</b>	<b>101</b>
Configuration documents . . . . .	101
Configuration document descriptions . . . . .	102
Object names . . . . .	104
Configuration repositories . . . . .	105
Handling temporary configuration files resulting from session timeout . . . . .	105
Changing the location of temporary configuration files . . . . .	105
Changing the location of backed-up configuration files . . . . .	106
Changing the location of temporary workspace files . . . . .	106
Backing up and restoring administrative configurations . . . . .	107
Transformation of configuration files . . . . .	107
Server configuration files: Resources for learning . . . . .	107
<b>Chapter 11. Administering application servers . . . . .</b>	<b>109</b>
Application servers . . . . .	109
Creating application servers . . . . .	109
Configuring application servers for UTF-8 encoding . . . . .	110
Managing application servers. . . . .	111
Server collection . . . . .	111
Starting servers. . . . .	119
Running application servers from a non-root user . . . . .	120
Detecting and handling problems with run-time components . . . . .	121
Stopping servers . . . . .	121
Creating generic servers . . . . .	122
Starting and terminating generic servers . . . . .	123
Configuring transport chains . . . . .	124
Transport chains . . . . .	125
HTTP transport channel custom property . . . . .	126
HTTP Tunnel transport channel custom property . . . . .	126
Troubleshooting transport chain problems . . . . .	127
Configuring HTTP transports . . . . .	127
Transport chains collection . . . . .	133
Transport chain settings . . . . .	133
HTTP tunnel transport channel settings . . . . .	134
HTTP transport channel settings . . . . .	134
TCP transport channel settings . . . . .	136
DCS transport channel settings . . . . .	138
Web container transport channel settings . . . . .	139
Custom services . . . . .	139
Developing custom services . . . . .	140
Custom service collection . . . . .	141
Process definition . . . . .	142
Defining application server processes . . . . .	142
Process definition settings . . . . .	143
Automatically restarting server processes . . . . .	146
Java virtual machines (JVMs) . . . . .	152
Using the JVM . . . . .	152
Java virtual machine settings. . . . .	152

Configuring JVM sendRedirect calls to use context root . . . . .	156
Setting custom JVM properties . . . . .	157
Tuning Java virtual machines. . . . .	159
Preparing to host applications . . . . .	159
Java memory tuning tips . . . . .	160
Configuring multiple network interface card support . . . . .	163
Tuning application servers . . . . .	164
Web services client to Web container optimized communication . . . . .	166
Application servers: Resources for learning . . . . .	167
<b>Notices . . . . .</b>	<b>169</b>
<b>Trademarks and service marks . . . . .</b>	<b>171</b>

---

## How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on articles in the WebSphere Application Server Information Center
  1. Display the article in your Web browser and scroll to the end of the article.
  2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
  3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- To send comments on PDF books, you can e-mail your comments to: **wasdoc@us.ibm.com** or fax them to 919-254-0206.

Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

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





---

## Chapter 1. Overview for setting up application serving environments

This topic summarizes the contents and organization of the administration documentation, including links to conceptual overviews and descriptions of new features.

This publication is for the administrator who is responsible for integrating application serving capabilities into an existing network environment. It looks at the product as part of a larger system, typically a production environment or realistic test environment.

This publication reiterates some installation and customization activities, including topology planning and creating product configurations. It carries the focus into the administrative realm, discussing port configuration and other network concerns. See also the *Installing your application serving environment* PDF.

This information expands the topology planning discussion by describing how to set up and maintain logical administrative domains of cells and nodes, and how to balance workload through clustering and high availability configurations.

---

### Setting up WebSphere Application Server products

IBM WebSphere Application Server products provide a next-generation application server on an industry-standard foundation. Each product addresses a distinct set of scenarios and needs. WebSphere Application Server, Version 6 product offerings are described on the WebSphere Application Server Web site at <http://www.ibm.com/software/webservers/appserv/was/>.

#### Planning

See Chapter 4, “Planning the installation (diagrams),” on page 11 for a description of typical scenarios for each WebSphere Application Server product.

#### Installing

See the *Installing your application serving environment* PDF for a description of installing the WebSphere Application Server product and other installable components on the product disc.

#### Configuring

See “Using the Profile creation wizard” on page 28 for a description of installing other stand-alone Application Servers on your machine.

#### Migrating

See the *Migrating, coexisting, and interoperating* PDF for a description of how to migrate applications and configuration data from a previous version of WebSphere Application Server.

#### Deploying applications

The Information Center describes a way to sample WebSphere Application Server functionality by quickly deploying Web components, such as servlets and JSP files. The method is not recommended as an official development method. See the *Developing and deploying applications* PDF to get started.

---

## Introduction: System administration

A variety of tools are provided for administering the WebSphere Application Server product:

- **Console**

The administrative console is a graphical interface that provides many features to guide you through deployment and systems administration tasks. Use it to explore available management options.

For more information, refer to “Introduction: Administrative console.”

- **Scripting**

The WebSphere administrative (wsadmin) scripting program is a powerful, non-graphical command interpreter environment enabling you to run administrative operations in a scripting language. You can also submit scripting language programs to run. The wsadmin tool is intended for production environments and unattended operations.

For more information, refer to “Introduction: Administrative scripting (wsadmin).”

- **Commands**

Command-line tools are simple programs that you run from an operating system command-line prompt to perform specific tasks, as opposed to general purpose administration. Using the tools, you can start and stop application servers, check server status, add or remove nodes, and complete similar tasks.

For more information, refer to “Introduction: Administrative commands” on page 3.

- **Programming**

The product supports a Java programming interface for developing administrative programs. All of the administrative tools supplied with the product are written according to the API, which is based on the industry standard Java Management Extensions (JMX) specification.

For more information, refer to “Introduction: Administrative programs” on page 3.

- **Data**

Product configuration data resides in XML files that are manipulated by the previously-mentioned administrative tools.

For more information, refer to “Introduction: Administrative configuration data” on page 4.

## Introduction: Administrative console

The administrative console is a graphical interface for performing deployment and system administration tasks. It runs in your Web browser. Your actions in the console modify a set of XML configuration files.

You can use the console to perform tasks such as:

- Add, delete, start, and stop application servers
- Deploy new applications to a server
- Start and stop existing applications, and modify certain configurations
- Add and delete Java 2 Platform, Enterprise Edition (J2EE) resource providers for applications that require data access, mail, URLs, and so on
- Manage variables, shared libraries, and other configurations that can span multiple application servers
- Configure product security, including access to the administrative console
- Collect data for performance and troubleshooting purposes
- Find the product version information. It is located on the front page of the console.

See the *Using the administrative clients* PDF for information on how you begin using the console. See also the **Reference > Administrator > Settings** section of the Information Center navigation. It lists the settings or properties you can configure.

## Introduction: Administrative scripting (wsadmin)

The WebSphere administrative (wsadmin) scripting program is a powerful, non-graphical command interpreter environment enabling you to run administrative operations in a scripting language. The wsadmin

tool is intended for production environments and unattended operations. You can use the wsadmin tool to perform the same tasks that you can perform using the administrative console.

The following list highlights the topics and tasks available with scripting. See the *Administering applications and their environment* PDF for more information on how to perform these tasks.

- Getting started with scripting Provides an introduction to WebSphere Application Server scripting and information about using the wsadmin tool. Topics include information about the scripting languages and the scripting objects, and instructions for starting the wsadmin tool.
- Deploying applications Provides instructions for deploying and uninstalling applications. For example, stand-alone Java archive files and Web archive files, the administrative console, remote enterprise archive (EAR) files, file transfer applications, and so on.
- Managing deployed applications Includes tasks that you perform after the application is deployed. For example, starting and stopping applications, checking status, modifying listener address ports, querying application state, configuring a shared library, and so on.
- Configuring servers Provides instructions for configuring servers, such as creating a server, modifying and restarting the server, configuring the Java virtual machine, disabling a component, disabling a service, and so on.
- Configuring connections to Web servers Includes topics such as regenerating the plug-in, creating new virtual host templates, modifying virtual hosts, and so on.
- Managing servers Includes tasks that you use to manage servers. For example, stopping nodes, starting and stopping servers, querying a server state, starting a listener port, and so on.
- Configuring security Includes security tasks, for example, enabling and disabling global security, enabling and disabling Java 2 security, and so on.
- Configuring data access Includes topics such as configuring a Java DataBase Connectivity (JDBC) provider, defining a data source, configuring connection pools, and so on.
- Configuring messaging Includes topics about messaging, such as Java Message Service (JMS) connection, JMS provider, WebSphere queue connection factory, MQ topics, and so on.
- Configuring mail, URLs, and resource environment entries Includes topics such as mail providers, mail sessions, protocols, resource environment providers, reference tables, URL providers, URLs, and so on.
- Dynamic caching Includes caching topics, for example, creating, viewing and modifying a cache instance.
- Troubleshooting Provides information about how to troubleshoot using scripting. For example, tracing, thread dumps, profiles, and so on.
- Obtaining product information Includes tasks such as querying the product identification.
- Scripting reference material Includes all of the reference material related to scripting. Topics include the syntax for the wsadmin tool and for the administrative command framework, explanations and examples for all of the scripting object commands, the scripting properties, and so on.

## Introduction: Administrative commands

Command-line tools are simple programs that you run from an operating system command-line prompt to perform specific tasks, as opposed to general purpose administration. Using the tools, you can start and stop application servers, check server status, add or remove nodes, and complete similar tasks.

See the *Administering applications and their environment* PDF for the names and syntax of all the commands that are available with the product. A subset of these commands are particular to system administration purposes.

## Introduction: Administrative programs

The product supports a Java programming interface for developing administrative programs. All of the administrative tools supplied with the product are written according to the API, which is based on the industry standard Java Management Extensions (JMX) specification. You can write a Java program that

performs any of the administrative features of the WebSphere Application Server administrative tools. You can also extend the basic WebSphere Application Server administrative system to include your own managed resources.

## Introduction: Administrative configuration data

Administrative tasks typically involve defining new configurations of the product or performing operations on managed resources within the environment. IBM WebSphere Application Server configuration data is kept in files. Because all product configuration involves changing the content of those files, it is useful to know the structure and content of the configuration files.

The WebSphere Application Server product includes an implementation of the Java Management Extension (JMX) specification. All operations on managed resources in the product go through JMX functions. This setup means a more standard framework underlying your administrative operations as well as the ability to tap into the systems management infrastructure programmatically.

---

## Introduction: Servers

### Application servers

Application servers provide the core functionality of the WebSphere Application Server product family. They extend the ability of a Web server to handle Web application requests, and much more. An application server enables a server to generate a dynamic, customized response to a client request.

For additional overview, refer to “Introduction: Application servers.”

## Introduction: Application servers

### Overview

An application server is a Java Virtual Machine (JVM) that is running user applications. The application server collaborates with the Web server to return a dynamic, customized response to a client request. Application code, including servlets, JavaServer Pages (JSP) files, enterprise beans and their supporting classes, runs in an application server. Conforming to the Java 2 platform, Enterprise Edition (J2EE) component architecture, servlets and JSP files run in a Web container, and enterprise beans run in an Enterprise JavaBeans (EJB) container.

To begin creating and managing an application server, see Chapter 11, “Administering application servers,” on page 109.

You can define multiple application servers, each running its own JVM. Enhance the operation of an application server by using the following options:

- Configure transport chains to provide networking services to such functions as the service integration bus component of IBM service integration technologies, WebSphere Secure Caching Proxy, and the high availability manager core group bridge service. See “Configuring transport chains” on page 124 for more information.
- Plug into an application server to define a hook point that runs when the server starts and shuts down. See “Custom services” on page 139 for more information.
- Define command-line information that passes to a server when it starts or initializes. See the *Using the administrative clients* PDF for more information.
- “Tuning application servers” on page 164
- Enhance the performance of the application server JVM. See “Using the JVM” on page 152 for more information.
- Use an Object Request Broker (ORB) for RMI/IIOP communication. See the *Developing and deploying applications* PDF for more information.

### Asynchronous messaging

The product supports asynchronous messaging based on the Java Messaging Service (JMS) of a JMS provider that conforms to the JMS specification version 1.1.

The JMS functions of the default messaging provider in WebSphere Application Server are served by one or more messaging engines (in a service integration bus) that runs within application servers.

## Generic Servers

In distributed platforms, the Generic Servers feature allows you create a generic server as an application server instance within the WebSphere Application Server administration, and associate it with a non-WebSphere server or process. The generic server can be associated with any server or process necessary to support the application server environment, including:

- A Java server
- A C or C++ server or process
- A CORBA server
- A Remote Method Invocation (RMI) server

After you define a generic server, you can use the Application Server administrative console to start, stop, and monitor the associated non-WebSphere server or process when stopping or starting the applications that rely on them.

For more information, refer to “Creating generic servers” on page 122.

## Introduction: Web servers

In the WebSphere Application Server product, an application server works with a Web server to handle requests for dynamic content, such as servlets, from Web applications. Go to <http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html> for the most current information about supported Web servers.

The application server and Web server communicate using “Web server plug-ins” on page 68. Chapter 7, “Communicating with Web servers,” on page 53 describes how to set up your Web server and Web server plug-in environment and how to create a Web server definition. The Web server definition associates a Web server with an application server. After you create a Web server definition, you can use the administrative console to perform the following functions for that Web server:

- Check the status of the Web server
- Generate a plug-in configuration file for that Web server.

If the Web server is an IBM HTTP Server (IHS) and the IHS Administration server is installed and properly configured, you can also:

- Display the IBM HTTP Server Error log (error.log) and Access log (access.log) files.
- Start and stop the server.
- Display and edit the IBM HTTP Server configuration file (httpd.conf).
- Propagate the plug-in configuration file after it is generated.

You can not propagate a plug-in configuration file for a non-IHS Web server. You must manually install an updated plug-in configuration file on that Web server.

After you set up your Web server and Web server plug-in, whenever you deploy a Web application, you must specify a Web server as the deployment target that serves as a router for requests to the Web application. The configuration settings in the plug-in configuration file (plugin-cfg.xml) for each Web server are based on the applications that are routed through that Web server. If the Web server plug-in configuration service is enabled, a Web server plug-in’s configuration file is automatically regenerated whenever a new application is associated with that Web server.

**Note:** Before starting the Web server, make sure you are authorized to run any Application Response Measurement (ARM) agent associated with that Web server.

Refer to your Web server documentation for information on how to administer that Web server. For tips on tuning your Web server plug-in, see “Web server plug-in tuning tips” on page 73.

---

## **Introduction: Environment**

The environment of the product applies to the configuring of Web server plug-ins, variables, and objects that you want consistent throughout a cell.

### **Web servers**

In the WebSphere Application Server product, an application server works with a Web server to handle requests for Web applications. The application Server and Web server communicate using a WebSphere HTTP plug-in for the Web server.

For more information, refer to “Introduction: Web servers” on page 5.

### **Variables**

A variable is a configuration property that can be used to provide a parameter for any value in the system. A variable has a name and a value to use in place of that name wherever the variable name is located within the system.

For more information, refer to “Configuring WebSphere variables” on page 90.

---

## Chapter 2. How do I administer applications and their environments?

- Establish the application serving environment
- Secure the application serving environment. (See the *Securing applications and their environment* PDF.)
- Set up Web access for applications. (See the *Developing and deploying applications* PDF.)
- Set up resources for applications to use. (See the *Developing and deploying applications* PDF.)
- Configure class loaders - see development and deployment. (See the *Developing and deploying applications* PDF.)
- Deploy and administer applications. (See the *Developing and deploying applications* PDF.)
- Use the administrative clients. (See the *Using administrative clients* PDF.)
- Troubleshoot deployment and administration. (See the *Troubleshooting and support* PDF.)

### Establish the application serving environment

The following tasks involve establishing application serving capability in your network environment, whether you use single or clustered application servers. Servers can be grouped into administrative domains known as nodes and cells.

See also the overview:

- Version 6 topology and terminology

---

### Create WebSphere profiles

Profiles are the files that define a stand-alone Application Server node, a managed node, or a deployment manager node. A profile also includes all of the files that the node can change.

---

### Administer configurations

Application server configuration files define the available application servers, their configurations, and their contents. You should periodically save changes to your administrative configuration. You can change the default locations of configuration files, as needed.

---

### Administer application servers

Create, configure, and operate application server processes. An application server configuration provides settings that control how an application server provides services for running enterprise applications and their components.

---

### Administer other server types

One step in the process of creating an application server is to specify a template. A server template is used to define the configuration settings of the new server. You have the option of specifying the default server template or choosing a template that is based on a server that already exists. The default template will be used if you do not specify a different template when you create the server.

You can create other types of servers, to represent Web servers in your topology, or for other purposes. There are two types of *generic* servers: (1) Non-Java applications or processes, or (2) Java applications or processes. A *custom service* provides the ability to plug into a WebSphere application server to define a hook point that runs when the server starts and shuts down.

-----



---

## Chapter 3. Setting up the application serving environment

This topic summarizes the contents of the documentation that helps you set up your application serving environment. This information is for administrators, particularly those performing installation, customization, and maintenance of topologies.

### **Chapter 4, “Planning the installation (diagrams),” on page 11**

In preparation for installation, this topic describes common product topologies that you can install with WebSphere Application Server, Version 6 products.

### **Chapter 5, “Configuring the product after installation,” on page 25**

This topic describes what to do after installing the product.

You can display the First Steps tool, an easy way to get started with the product.

### **Chapter 6, “Configuring ports,” on page 49**

This topic provides information about port number settings for Version 6 and previous versions, for use in coexistence and interoperability situations.

### **Chapter 7, “Communicating with Web servers,” on page 53**

This topic describes how to install and configure WebSphere plug-ins for Web servers, enabling communication between Web servers and application servers.

### **Chapter 8, “Setting up the administrative architecture,” on page 75**

This topic describes how to configure administrative services.

### **Chapter 9, “Configuring the environment,” on page 83**

This topic describes how to configure settings for virtual hosts, variables, and shared libraries to assist in handling requests among Web applications, Web containers, and application servers.

### **Chapter 10, “Working with server configuration files,” on page 101**

This topic describes how to change the default locations of configuration files, as needed. Application server configuration files define the available application servers, their configurations, and their contents.

### **Chapter 11, “Administering application servers,” on page 109**

This topic describes how to configure individual application servers to provide services for running enterprise applications and their components.



---

## Chapter 4. Planning the installation (diagrams)

This topic describes common product topologies that you can install with the product.

Use this topic to understand the capabilities of your product package. Knowing what you can do with the product might influence how you install the product and other installable components on the product disc.

This topic describes topology diagrams and shows you how to create the topologies by showing what components to install for each topology.

### Phased installation roadmap

To install a Version 6 production environment, you install the following components:

- The WebSphere Application Server product on your product CD
- A supported Web server, such as the IBM HTTP Server V6 on the product CD
- A binary plug-in module for your Web server from the product CD

You can also use the product CD to install an application client environment on a client machine. Running Java 2 Platform, Enterprise Edition (J2EE) and thin application clients that communicate with WebSphere Application Server requires that elements of the Application Server are installed on the machine on which the client runs. However, if the machine does not have the Application Server installed, you can install Application Server clients to provide a stand-alone client run-time environment for your client applications.

You can use the Application Server Toolkit CD in the primary packet of discs to install a development environment. Or you can use the Rational Application Developer Trial CD in the supplemental packet of discs to install a fully integrated development environment that includes an exact replica of the Application Server for development testing.

### Installation features

Installation features in V6 include:

Feature	Description
The WebSphere Application Server product includes Application Server nodes.	You can use the Profile creation wizard to create stand-alone application server nodes after installing the product. You do not have to reinstall the product to create additional application servers. All application servers on a machine share the same core product files.
The product CD includes all of the installable components that are required to create an e-business environment.	You can use the product CD to install the IBM HTTP Server, the Web server plug-ins, and the WebSphere Application Server Clients. You do not have to use separate CDs. Separate installation programs exist within component directories on the product CD.
Each installable component has its own installation program.	You can use the V6 launchpad to install any installable component on the product CD. Or you can install each component directly using the <b>install</b> command in each component directory.
The launchpad can install any installable product in the primary packet of compact discs.	The launchpad can also install the Application Server Toolkit on Windows 2000 and Linux (Intel) systems. The Application Server Toolkit is on a separate disc, which requires you to change discs to launch the installation.

Review topology diagrams for each of the following installable components to determine which topology best fits your needs. The diagrams and their accompanying procedures can serve as a roadmap for installing a similar topology.

This topic describes installation scenarios for the following installable components:

- WebSphere Application Server (base product)
- Web server plug-ins
- Application clients

In addition to product installation diagrams for the installable components, this topic also links to a roadmap for using the Profile creation wizard, which is new for Version 6. The Profile creation wizard lets you create run-time environments for application server processes.

Each of the following installation scenarios includes topology diagrams and associated installation steps. Each step links to a specific procedure for installing a component or to a description of a command or tool.

1. Review the installation scenarios for the base WebSphere Application Server product, as described in “Planning to install WebSphere Application Server.”
2. Review the installation scenarios for the WebSphere Application Server plug-ins, as described in “Planning to install Web server plug-ins” on page 16.
3. Review the installation scenarios for the application clients, as described in “Planning to install WebSphere Application Server Clients” on page 19.
4. Review the installation scenarios for the Profile creation wizard, as described in “Planning to create application server environments” on page 20.
5. **Optional:** Review interoperability and coexistence diagrams to know what is possible with Version 6. WebSphere Application Server V6 can 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* 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 WebSphere Application Server version to a later one when some machines are at the earlier version and some machines are at the later version. 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.

See the *Migrating, coexisting, and Interoperating* PDF for more information.

6. **Optional:** Consider performance when designing your network, as described in “Queuing network” on page 20.

You can review installation scenarios to identify the specific steps to follow when installing more than one component on a single machine or on separate machines.

After determining an appropriate installation scenario, you are ready to install the necessary components and to configure the products for the system that you selected.

---

## Planning to install WebSphere Application Server

This topic describes common installation scenarios and links to component installation procedures for each scenario.

IBM WebSphere Application Server, Version 6.0 is an integrated platform that contains an Application Server, a set of Web development tools, a Web server, and additional supporting software and documentation.

The following information describes scenarios for installing the product in various topologies on one or more machines:

- **Scenario 1:** Single-machine installation of WebSphere Application Server

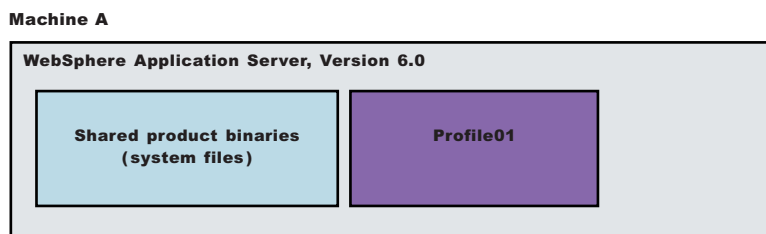
- **Scenario 2:** Single-machine installation of WebSphere Application Server and a Web server
- **Scenario 3:** Two-machine installation of WebSphere Application Server and a Web server
- **Scenario 4:** Creating multiple profiles that use one installation of WebSphere Application Server

Each scenario includes a diagram and a list of detailed installation steps.

- **Scenario 1:** Single-machine installation of WebSphere Application Server

Installing WebSphere Application Server by itself on a single machine creates a stand-alone application server, which is automatically named server1. Installing the base product creates the core product files and a *profile* for application server. The profile is a separate set of files that define the application server environment.

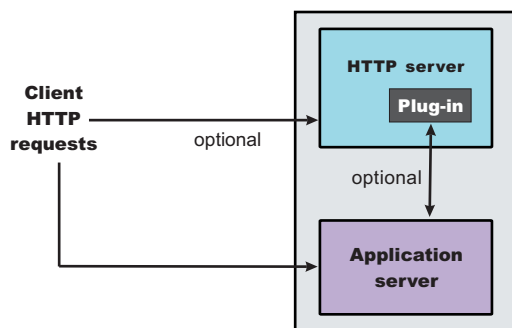
In this scenario, application server uses its internal HTTP transport chain for communication, which is suitable for handling an application with a relatively low request work load. For example, this type of installation can support a simple test environment or a departmental intranet environment.



1. Install WebSphere Application Server.

- **Scenario 2:** Single-machine installation of WebSphere application servers and a Web server

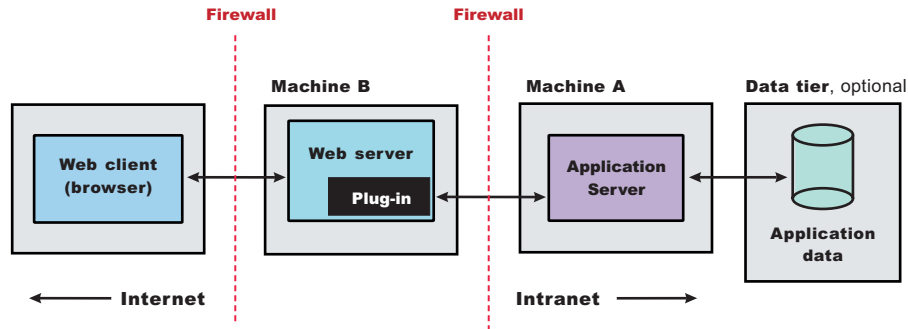
Installing a Web server, such as IBM HTTP Server, on the same machine as the application server provides a more robust Web server environment. Installing a Web server plug-in is a requirement for the Web server to communicate with the application server. This type of installation supports rigorous testing environments or production environments that do not require a firewall. However, this is not a typical production environment.



1. Install WebSphere Application Server.
2. Install IBM HTTP Server or another supported Web server.
3. Install the Web server plug-ins and configure the Web server using the Plug-ins installation wizard.

- **Scenario 3:** Two-machine installation of WebSphere Application Server and a Web server

In the typical production environment, the application server on one machine communicates with a Web server on a separate (remote) machine through the Web server plug-in. Optional firewalls can provide additional security for the application server machine.



1. Install WebSphere Application Server on Machine A.
2. Install IBM HTTP Server or another supported Web server on Machine B.
3. Install the Web server plug-ins and configure the Web server using the Plug-ins installation wizard on Machine B.
4. The Plug-ins installation wizard creates a script named `configureWeb_server_name` in the `plugins_install_root/bin` directory on Machine B. Copy the script to the `install_root/bin` directory on Machine A.
5. Run the `configureWeb_server_name` script to create a Web server definition in the administrative console. You can then use the administrative console to manage the Web server.
6. Propagate the `plugin-cfg.xml` file from the Application Server to the Web server using the administrative console. Click **Servers > Web server > Propagate Plug-in**. (Web servers other than IBM HTTP Server require manual propagation.)

- **Scenario 4:** Creating multiple profiles that use one installation of WebSphere Application Server

A profile is a separate data partition that contains the files that define the run-time environment for an application server. A default profile is created during the installation of the base product. Create additional profiles using the Profile creation wizard. Each profile defines a separate stand-alone application server that has its own administrative interface.

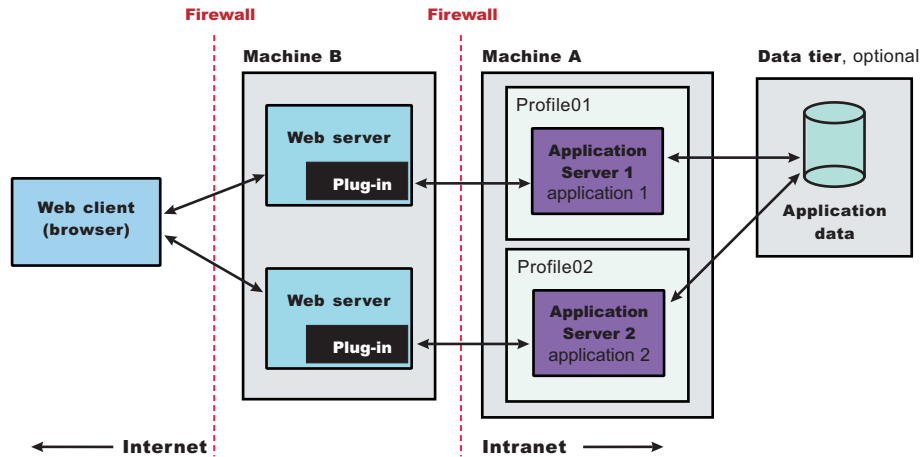
After creating a profile and installing a dedicated Web server, use the Plug-ins installation wizard to install a Web server plug-in and to update the Web server configuration file. The Web server can then communicate with the application server.

With topology, each profile has unique applications, configuration settings, data, and log files, and shares the same set of core product files. Creating multiple profiles creates multiple application server environments that you can dedicate to different purposes.

For example, each application server on a Web site can serve a different application. In another example, each application server can be a separate test environment that you assign to a programmer or to a development team.

### Updating the core product files

Another feature of having multiple profiles is enhanced serviceability. When a refresh pack or a fix pack updates the core product files on a machine, all of the application server profiles that were created from the core product files begin using the updated files. In some situations, you might prefer to not update all of the application servers on a machine. In such situations, simply install the product a second time to create a second set of core product files. Create application server profiles from both installations to manage the product updates incrementally.



1. Install WebSphere Application Server on Machine A.
2. Install IBM HTTP Server or another supported Web server on Machine B.
3. Install the Web server plug-ins and configure the Web server using the Plug-ins installation wizard on Machine B.
4. The Plug-ins installation wizard creates a script named `configureWeb_server_name` in the `plugins_install_root/bin` directory on Machine B. Copy the script to the `install_root/bin` directory on Machine A.
5. Run the `configureWeb_server_name` script to create a Web server definition in the administrative console. You can then use the administrative console to manage the Web server.
6. Propagate the `plugin-cfg.xml` file from the application server to the Web server using the administrative console. Click **Servers > Web server > Propagate Plug-in**. (Web servers other than IBM HTTP Server require manual propagation.)
7. Create the second Application Server profile using the Profile creation wizard on Machine A. Make the profile the default profile during the profile creation by selecting the check box on the appropriate panel.  
The script that the Plug-ins installation wizard creates works on the default profile only. So, this script can only create a Web server definition on the profile that is the default profile at the time that the script runs.
8. Install a second IBM HTTP Server or another supported Web server on Machine B.
9. On Machine B, install the Web server plug-ins to configure the second Web server using the Plug-ins installation wizard. Both Web servers share a single installation of the plug-in binaries but must be configured individually.
10. The Plug-ins installation wizard creates a script named `configureWeb_server_name` for the second Web server. The script is in the `plugins_install_root/bin` directory on Machine B. Copy the script to the `install_root/bin` directory on Machine A.
11. Run the `configureWeb_server_name` script to create a Web server definition in the administrative console. You can then use the administrative console to manage the Web server.
12. Propagate the `plugin-cfg.xml` file from the second application server to the Web server using the administrative console. Click **Servers > Web server > Propagate Plug-in**. (Web servers other than IBM HTTP Server require manual propagation.)

You can review common installation scenarios to find a possible match for the topology that you intend to install. Each product installation diagram provides a high-level procedure for installing the components that comprise the topology.

After determining a possible topology, you are ready to follow the detailed installation instructions for each product that you plan to install.

## Planning to install Web server plug-ins

This topic describes common installation scenarios and links to component installation procedures for each scenario.

The primary production configuration is an application server on one machine and a Web server on a separate machine. This configuration is referred to as a *remote* configuration. Contrast the remote configuration to the local configuration, where the application server and the Web server are on the same machine.

The Plug-ins installation wizard has four main tasks:

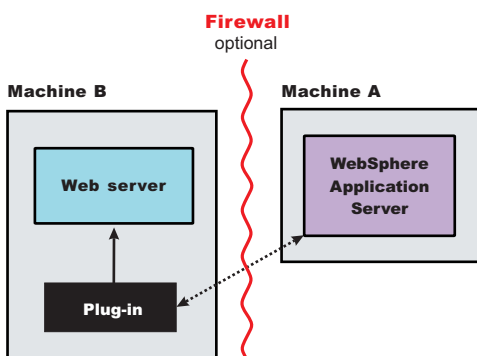
- Installs the binary plug-in module on the Web server machine.
- Configures the Web server configuration file on the Web server machine to point to the binary plug-in module and to the XML configuration file for the binary module.
- Installs a temporary XML configuration file for the binary module (`plugin-cfg.xml`) on the Web server machine in remote scenarios.
- Creates the configuration for a Web server definition on the application server machine. The wizard processes the creation of the Web server definition differently depending on the scenario:
  - Recommended remote stand-alone Application Server installation:  
Creates a configuration script that you run on the application server machine. Install the Web server and its plug-in on a different machine than the application server. This configuration is recommended for a production environment.
  - Local stand-alone Application Server installation:  
Detects the default profile on a local application server machine and creates the Web server definition for it directly. Install the Web server and its plug-in on the same machine with the application server. This configuration is for development and test environments.

Select a link to go to the appropriate steps in the following procedure.

- **Set up a remote Web server installation.**

The remote Web server configuration is recommended for production environments.

The remote installation installs the Web server plug-in on the Web server machine when the application server is on a separate machine, such as shown in the following graphic:




### Remote installation scenario

Table 1. Installation and configuration

Step	Machine	Task
1	A	Install your WebSphere Application Server product. See the <i>Installing your application serving environment</i> PDF.



Table 1. Installation and configuration (continued)

Step	Machine	Task
2	B	Install IBM HTTP Server or another supported Web server. See your Web server documentation.
3	B	Install the binary plug-in module using the Plug-ins installation wizard. See the <i>Installing your application serving environment</i> PDF.  The script for creating and configuring the Web server is created under the <code>plug-ins_install_root/bin</code> directory.
4	B	Copy the <code>configureWeb_server_name</code> script to Machine A. If one machine is running under Linux or UNIX and the other machine is running under Windows, copy the script from the <code>plug-ins_install_root/bin/crossPlatformScripts</code> directory.
5	A	Paste the <code>configureWeb_server_name</code> script from Machine B to the <code>was_install_root/bin</code> directory on Machine A.
6	A	Run the script from a command line.
7	A	Verify that the application server is running. Open the administrative console and save the changed configuration.
8	B	 <code>UNIX</code> Run the <code>plug-ins_install_root/setupPluginCfg.sh</code> script for a Domino Web Server before starting a Domino Web server. Otherwise, start the Web server.
9	B	Run the snoop servlet.  To verify with your own application, regenerate and propagate the <code>plugin-cfg.xml</code> file after installing the application.

#### Regeneration of the plugin-cfg.xml file

During the installation of the plug-ins, the temporary `plugin-cfg.xml` file is installed on Machine B in the `plug-ins_install_root/config/web_server_name` directory.

The Web server plug-in configuration service regenerates the `plugin-cfg.xml` file automatically.

To use the real `plugin-cfg.xml` file from the application server, propagate the `plugin-cfg.xml` file as described in the next section.

#### Propagation of the plugin-cfg.xml file

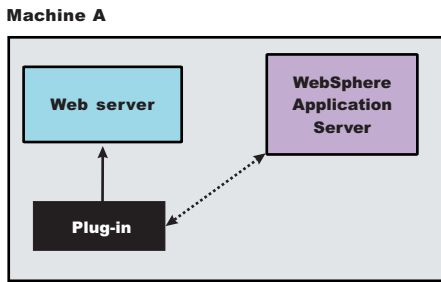
The Web server plug-in configuration service propagates the `plugin-cfg.xml` file automatically for IBM HTTP Server 6.0 only.

For all other Web servers, propagate the plug-in configuration file manually. Copy the `plugin-cfg.xml` file from the `profiles_install_root/config/cells/cell_name/nodes/Web_server_name_node/servers/web_server_name` directory on Machine A. Paste the file into the `plug-ins_install_root/config/web_server_name` directory on Machine B.

- **Set up a local Web server configuration.**


The local Web server configuration is recommended for a development or test environment.

A local installation includes the Web server plug-in, the Web server, and the application server on the same machine:



### Local installation scenario

Table 2. Installation and configuration

Step	Machine	Task
1	A	Install your WebSphere Application Server product. See the <i>Installing your application serving environment</i> PDF.
2	A	Install IBM HTTP Server or another supported Web server. See your Web server documentation.
3	A	Install the binary plug-in module using the Plug-ins installation wizard. See the <i>Installing your application serving environment</i> PDF.  The Web server definition is automatically created and configured during the installation of the plug-ins.
4	A	Verify that the application server is running. Open the administrative console and save the changed configuration.
5	B	 Run the <code>plug-ins_install_root/setupPluginCfg.sh</code> script for a Domino Web Server before starting a Domino Web server. Otherwise, start the Web server.
6	B	Run the snoop servlet.  To verify with your own application, regenerate and propagate the <code>plugin-cfg.xml</code> file after installing the application.

### Regeneration of the plugin-cfg.xml file

The Web server plug-in configuration service regenerates the `plugin-cfg.xml` file automatically.

The `plugin-cfg.xml` file is generated in the `profiles_install_root/profile_name/config/cells/cell_name/nodes/Web_server_name_node/servers/web_server_name` directory. The generation occurs when the Web server definition is created.

### Propagation of the plugin-cfg.xml file

The local file does not require propagation.

You can set up a remote or local Web server by installing Application Server, the Web server, and then the Web server plug-ins.

See your Web server documentation for more information about the files involved in configuring a Web server.

See the *Installing your application serving environment* PDF for information about the logic behind the processing scenarios for the Plug-ins installation wizard.

See the *Installing your application serving environment* PDF for information about how the Plug-ins installation wizard configures supported Web servers.

See See the *Installing your application serving environment* PDF for information about other installation scenarios for installing Web server plug-ins.

---

## Planning to install WebSphere Application Server Clients

This topic helps you examine typical topologies and uses for WebSphere Application Server Clients.

This topic is one in a series of topics described in Chapter 4, “Planning the installation (diagrams),” on page 11. Consider all of the planning scenarios that are mentioned in the parent article to determine the best approach to installing your e-business network. This topic describes installing and using the WebSphere Application Server Clients.

In a traditional client server environment, the client requests a service and the server fulfills the request. Multiple clients use a single server. Clients can also access several different servers. This model persists for Java clients except that now these requests use a client run-time environment.

In this model, the client application requires a servlet to communicate with the enterprise bean, and the servlet must reside on the same machine as the WebSphere Application Server.

The Application Client for WebSphere Application Server, Version 6 now consists of the following models:

- ActiveX application client
- Applet client
- J2EE application client
- Pluggable and thin application clients

The following graphic shows a topology for installing the Application Client and using client applications:

.

The example shows two types of application clients installed in a topology that uses client applications to access applications and data on Machine A:

- The ActiveX application client on Machine B is a Windows only client that uses the Java Native Interface (JNI) architecture to programmatically access the Java virtual machine (JVM) API. The JVM code exists in the same process space as the ActiveX application (Visual Basic, VBScript, or Active Server Pages (ASP) files) and remains attached to the process until that process terminates.
- The J2EE application client on Machine C is a Java application program that accesses enterprise beans, Java Database Connectivity (JDBC) APIs, and Java Message Service message queues. The application program must configure the execution environment of the J2EE application client and use the Java Naming and Directory Interface (JNDI) name space to access resources.

Use the following procedure as a roadmap for installing the Application Client.

1. Install the WebSphere Application Server product from your product CD on Machine A to establish the core product files.
2. Use the Profile creation wizard to create the additional stand-alone application server profile.
3. Use the administrative console of each application server to deploy any user applications.
4. Use the administrative console of each application server to create a Web server configuration for the Web server.
5. Use the administrative console of each application server to regenerate each `plugin-cfg.xml` file in the local Web server configuration.
6. Install the IBM HTTP Server from the product CD on Machine A.
7. Use the Plug-ins installation wizard to install the plug-in for IBM HTTP Server on Machine A.

- The wizard automatically configures the HTTP Server to communicate with the first application server.
8. Install the Application Client from your product CD on Machine B.
    - a. Select the Custom install type.
    - b. Select the ActiveX to EJB Bridge feature.
    - c. Select to add the Java run time to the system path.
    - d. Select the Java run time as the default JRE, which adds the Java run time path to the beginning of the system path.
  9. Install the Application Client from your product CD on Machine C.
    - a. Select the Custom install type.
    - b. Select the J2EE application client feature.

This topic can help you plan run-time environments for client applications.

See the *Using administrative clients* PDF for information about creating client applications.

---

## Planning to create application server environments

Application server profiles are the run-time environments for application server processes. This topic describes common scenarios for creating application server profiles and provides links to profile creation procedures for each scenario.

Install the core product files for a WebSphere Application Server product before using the Profile creation wizard to create additional application server run-time environments.

This topic describes how to use the Profile creation wizard to create application server profiles. Each profile is a run-time environment for the application server that it defines, with data files, configuration files, applications, and an administrative console.

Create a stand-alone application server, as described in “Using the Profile creation wizard to create an application server” on page 30.

The installation procedure creates an application server during installation named server1. However, you can use the Profile creation wizard to create more stand-alone application servers on a machine where server1 or another application server already exists.

You can create additional application server processes using the Profile creation wizard.

After installing the product and optionally adding more application servers with the Profile creation wizard, you are ready to deploy applications to test the environment.

---

## Queuing network

WebSphere Application Server contains interrelated components that must be harmoniously tuned to support the custom needs of your end-to-end e-business application. These adjustments help the system achieve maximum throughput while maintaining the overall stability of the system. This group of interconnected components is known as a queuing network. These queues or components include the network, Web server, Web container, EJB container, data source, and possibly a connection manager to a custom back-end system. Each of these resources represents a queue of requests waiting to use that resource. Various queue settings include:

- IBM HTTP Server: MaxClients for UNIX and ThreadsPerChild for Windows NT and Windows 2000 systems described in “Web server tuning parameters” on page 69.

- Web container: **Maximum size** described in “Thread pool settings” on page 117, **MaxKeepAliveConnections** and **MaxKeepAliveRequests** described in “HTTP transport custom properties” on page 130.
- **Tuning Object Request Brokers** explained in “Tuning application servers” on page 164.
- Data source **connection pooling** and **statement cache size** are explained in the *Developing and deploying applications* PDF.

Most of the queues that make up the queuing network are closed queues. A closed queue places a limit on the maximum number of requests present in the queue, while an open queue has no limit. A closed queue supports tight management of system resources. For example, the Web container thread pool setting controls the size of the Web container queue. If the average servlet running in a Web container creates 10MB of objects during each request, a value of 100 for thread pools limits the memory consumed by the Web container to 1GB.

In a closed queue, requests can be active or waiting. An active request is doing work or waiting for a response from a downstream queue. For example, an active request in the Web server is doing work, such as retrieving static HTML, or waiting for a request to complete in the Web container. A waiting request is waiting to become active. The request remains in the waiting state until one of the active requests leaves the queue.

All Web servers supported by WebSphere Application Server are closed queues, as are WebSphere Application Server data sources. You can configure Web containers as open or closed queues. In general, it is best to make them closed queues. EJB containers are open queues. If there are no threads available in the pool, a new one is created for the duration of the request.

If enterprise beans are called by servlets, the Web container limits the number of total concurrent requests into an EJB container, because the Web container also has a limit. The Web container limits the number of total concurrent requests only if enterprise beans are called from the servlet thread of execution. Nothing prevents you from creating threads and bombarding the EJB container with requests. Therefore, servlets should not create their own work threads.

## Queuing and clustering

Cloning application servers can be a valuable asset in configuring highly-scalable production environments, especially when the application is experiencing bottlenecks that are preventing full CPU utilization of symmetric multiprocessing (SMP) servers. When adjusting the WebSphere Application Server system queues in clustered configurations, remember that when a server is added to a cluster, the server downstream receives twice the load.

Two servlet engines are located between a Web server and a data source. It is assumed that the Web server, servlet engines and data source, but not the database, are all running on a single SMP server. Given these constraints, the following queue considerations must be made:

- Double the Web server queue settings to ensure ample work is distributed to each Web container.
- Reduce the Web container thread pools to avoid saturating a system resource like CPU or another resource that the servlets are using.
- Reduce the data source to avoid saturating the database server.
- Reduce Java heap parameters for each instance of the application server. For versions of the Java virtual machine (JVM) shipped with WebSphere Application Server, it is crucial that the heap from all JVMs remain in physical memory. For example, if a cluster of four JVMs is running on a system, enough physical memory must be available for all four heaps.

## Queue configuration tips

The following section outlines a methodology for configuring the WebSphere Application Server queues. Moving the database server onto another machine or providing more powerful resources, for example a faster set of CPUs with more memory, can dramatically change the dynamics of your system.

There are four tips for queuing:

- **Minimize the number of requests in WebSphere Application Server queues.**

In general, requests wait in the network in front of the Web server, rather than waiting in WebSphere Application Server. This configuration only supports those requests that are ready for processing to enter the queuing network. Specify that the queues furthest upstream or closest to the client are slightly larger, and queues further downstream or furthest from the client are progressively smaller.

Queues in the queuing network become progressively smaller as work flows downstream. When 200 client requests arrive at the Web server, 125 requests remain queued in the network because the Web server is set to handle 75 concurrent clients. As the 75 requests pass from the Web server to the Web container, 25 requests remain queued in the Web server and the remaining 50 are handled by the Web container. This process progresses through the data source until 25 user requests arrive at the final destination, the database server. Because there is work waiting to enter a component at each point upstream, no component in this system must wait for work to arrive. The bulk of the requests wait in the network, outside of WebSphere Application Server. This type of configuration adds stability, because no component is overloaded.

- **Draw throughput curves to determine when the system capabilities are maximized.**

You can use a test case that represents the full spirit of the production application by either exercising all meaningful code paths or using the production application. Run a set of experiments to determine when the system capabilities are fully stressed or when it has reached the saturation point. Conduct these tests after most of the bottlenecks are removed from the application. The goal of these tests is to drive CPUs to near 100% utilization. For maximum concurrency through the system, start the initial baseline experiment with large queues. For example, start the first experiment with a queue size of 100 at each of the servers in the queuing network: Web server, Web container and data source. Begin a series of experiments to plot a throughput curve, increasing the concurrent user load after each experiment. For example, perform experiments with one user, two users, five, 10, 25, 50, 100, 150 and 200 users. After each run, record the throughput requests per second, and response times in seconds per request. The curve resulting from the baseline experiments resembles the following typical throughput curve shown as follows:

The WebSphere Application Server throughput is a function of the number of concurrent requests present in the total system. Section A, the light load zone, shows that the number of concurrent user requests increases, the throughput increases almost linearly with the number of requests. At light loads, concurrent requests face very little congestion within the WebSphere Application Server system queues. At some point, congestion starts to develop and throughput increases at a much lower rate until it reaches a saturation point that represents the maximum throughput value, as determined by some bottleneck in the WebSphere Application Server system. The most manageable type of bottleneck occurs when the WebSphere Application Server machine CPUs become fully utilized because adding CPUs or more powerful CPUs fixes the bottleneck.

In the heavy load zone or Section B, as the concurrent client load increases, throughput remains relatively constant. However, the response time increases proportionally to the user load. That is, if the user load is doubled in the heavy load zone, the response time doubles. At some point, represented by Section C, the buckle zone, one of the system components becomes exhausted. At this point, throughput starts to degrade. For example, the system might enter the buckle zone when the network connections at the Web server exhaust the limits of the network adapter or if the requests exceed operating system limits for file handles.

If the saturation point is reached by driving CPU utilization close to 100%, you can move on to the next step. If the saturation CPU occurs before system utilization reaches 100%, it is likely that another



bottleneck is being aggravated by the application. For example, the application might be creating Java objects causing excessive garbage collection bottlenecks in the Java code.

There are two ways to manage application bottlenecks: remove the bottleneck or clone the bottleneck. The best way to manage a bottleneck is to remove it. You can use a Java-based application profiler, such as Rational Application Developer, Performance Trace Data Visualizer (PTDV), Borland's Optimizeit, JProbe or Jinsight to examine overall object utilization.

- **Decrease queue sizes while moving downstream from the client.**

The number of concurrent users at the throughput saturation point represents the maximum concurrency of the application. For example, if the application saturates WebSphere Application Server at 50 users, using 48 users might produce the best combination of throughput and response time. This value is called the Max Application Concurrency value. Max Application Concurrency becomes the preferred value for adjusting the WebSphere Application Server system queues. Remember, it is desirable for most users to wait in the network; therefore, queue sizes should increase when moving downstream farther from the client. For example, given a Max Application Concurrency value of 48, start with system queues at the following values: Web server 75, Web container 50, data source 45. Perform a set of additional experiments adjusting these values slightly higher and lower to find the best settings.

To help determine the number of concurrent users, view the Servlet Engine Thread Pool and Concurrently Active Threads metric in the Tivoli Performance Viewer.

- **Adjust queue settings to correspond to access patterns.**

In many cases, only a fraction of the requests passing through one queue enters the next queue downstream. In a site with many static pages, a number of requests are fulfilled at the Web server and are not passed to the Web container. In this circumstance, the Web server queue can be significantly larger than the Web container queue. In the previous example, the Web server queue was set to 75, rather than closer to the value of Max Application Concurrency. You can make similar adjustments when different components have different execution times.

For example, in an application that spends 90% of its time in a complex servlet and only 10% of its time making a short JDBC query, on average 10% of the servlets are using database connections at any time, so the database connection queue can be significantly smaller than the Web container queue. Conversely, if the majority of servlet execution time is spent making a complex query to a database, consider increasing the queue values at both the Web container and the data source. Always monitor the CPU and memory utilization for both the WebSphere Application Server and the database servers to verify that the CPU or memory are not saturating.





---

## Chapter 5. Configuring the product after installation

This topic summarizes how to configure the application serving environment.

Use the First steps console to configure and test the WebSphere Application Server environment after installation.

This procedure uses the First steps console to launch the installation verification test (IVT) that tests and verifies your WebSphere Application Server environment. This procedure also uses the First steps console to launch the Profile creation wizard to create an additional Application Server.

1. Start the First steps console by selecting the check box on the last panel of the wizard.

The First steps console can start automatically at the end of the installation. Select the check box on the last panel of the Installation wizard.

The First steps console is an easy way to start using the product. The console provides one-stop access to the administrative console, Samples Gallery, Profile creation wizard, installation verification test, Migration wizard, and other activities.

See the description of the “firststeps command” for more information.

2. Click **Installation verification** on the First steps console.

The installation verification test starts the Application Server process named server1 and runs several tests to verify that the server1 process can start without errors.

See “Using the installation verification test” on page 45 for more information.

3. Click **Profile creation wizard** on the First steps console to create an Application Server profile.

You can create multiple Application Servers on your system without installing the product again.

See “Using the Profile creation wizard to create an application server” on page 30.

4. Start the First steps console by selecting the check box on the last panel of the Profile creation wizard.

This First steps console belongs to the Application Server profile that you just created. Each profile has its own First steps console.

5. Click **Installation verification** on the First steps console.

The installation verification test starts the new Application Server process named server1 and runs several tests to verify that the server1 process can start without error.

This procedure results in configuring and testing the Application Server environment.

See “Planning to install WebSphere Application Server” on page 12 for diagrams of topologies that you can create using the First steps console and the Profile creation wizard.

---

### firststeps command

The **firststeps** command starts the First steps console.

#### The First steps console

The First steps console is a post-installation ease-of-use tool for directing WebSphere Application Server elements from one place. Options display dynamically on the First steps console, depending on features you install. With all of the options present, you can use the First steps console to start or stop the application server, verify the installation, access the information center, access the administrative console, launch the Migration wizard, or access the Samples gallery.

Select the check box to start the First steps console at the end of the product installation.

You can also start the First steps console from the command line as described later.

## Installation verification

This option starts the installation verification test (IVT). The test consists of starting and monitoring the application server during its start up.

If this is the first time that you have used the First steps console since creating an application server profile, click **Installation verification** to verify that all is well with your installation. The verification process starts the application server.

If you select the **Installation verification** option, the **Start the server** option is grayed out while the IVT is running.

The IVT provides the following useful information about the application server:

- The server name: server1
- The name of the profile
- The profile file path
- The type of profile: default
- The cell name
- The node name
- The current encoding
- The port number for the administrative console
- Various informational messages that include the location of the SystemOut.log file and how many errors are listed within the file
- A completion message

## Start the server

This option toggles to **Stop the server** when the application server is running.

After selecting the **Start the server** option, an output screen displays with status messages. The success message informs you that the server is open for e-business. Then the menu item changes to **Stop the server**.

If you select the **Start the server** option, the **Installation verification** option is grayed out while the application server is running.

## Administrative console

This option is grayed out until the application server is running.

The administrative console is a configuration editor that runs in a Web browser. The administrative console lets you work with XML configuration files for the application server. To launch the administrative console, click **Administrative console**. You can also point your browser to <http://localhost:9060/ibm/console> to start the administrative console. Substitute your own host name in the address if the localhost variable does not resolve correctly. As the administrative console opens, it prompts you for a login name. This is not a security item, but merely a tag to identify configuration changes that you make during the session. Secure signon is also available.

## Profile creation wizard

This option starts the Profile creation wizard. The wizard lets you create additional application servers. A *profile* consists of files that define the run-time environment for the application server. Each environment has its own administrative interface. This means that the new application server has its own administrative console.

Each application server has its own First steps console. The location of the command is within the set of files in the profile. A prompt to launch the First steps console displays on the last panel of the Profile creation wizard.

## Samples gallery

This option starts the Samples gallery. The option is grayed out until you start the application server. The option displays when you have installed the Samples during installation. The typical installation includes the Samples by default.

From the First steps console, click **Samples gallery** to explore the application Samples. Alternatively you can point your browser directly to `http://localhost:9080/WSsamples`. Substitute your own host name in the address if the localhost variable does not resolve correctly. The Web address is case sensitive. Substitute your own host name in the address.

### Information center for WebSphere Application Server

This option links you to the online information center at the `http://publib.boulder.ibm.com/infocenter/ws60help/index.jsp` IBM Web address.

### Migration wizard

This option starts the Migration wizard. The Migration wizard is a new graphical interface to the migration tools. The migration tools are described in the *Migrating, coexisting, and interoperating* PDF.

**Exit** This option closes the First steps console.

### Location of the command file

Installing the product creates a default profile for the server1 application server. The location of the First steps console for the default profile is:

- **UNIX** `install_root/profiles/default/firststeps/firststeps.sh`
- **Windows** `install_root\profiles\default\firststeps\firststeps.bat`

The location of the `firststeps.sh` or `firststeps.bat` script for any profile is:

- **UNIX** `install_root/profiles/profile_name/firststeps/firststeps.sh`
- **Windows** `install_root\profiles\profile_name\firststeps\firststeps.bat`

### Parameters

No parameters are associated with this command.

### Syntax for the firststeps command







Use the following syntax for the command:

- **UNIX** `./firststeps.sh`
- **Windows** `firststeps.bat`

### Usage tips

The following links exist on the First steps console for the base WebSphere Application Server product:

Option	Link
<b>Installation verification</b>	<p>Calls the <b>ivt</b> command.</p> <p>The location of the installation verification test varies per platform:</p> <ul style="list-style-type: none"> <li>• <b>UNIX</b> <code>install_root/profiles/profile_name/bin/ivt.sh</code></li> <li>• <b>Windows</b> <code>install_root\profiles\profile_name\bin\ivt.bat</code></li> </ul>
<b>Start the server</b>	<p>Calls the <b>startServer</b> command.</p> <p>The location of the <b>startServer</b> command varies per platform:</p> <ul style="list-style-type: none"> <li>• <b>UNIX</b> <code>install_root/profiles/profile_name/bin/startServer.sh server1</code></li> <li>• <b>Windows</b> <code>install_root\profiles\profile_name\bin\startServer.bat server1</code></li> </ul> <p>When you have more than one application server on the same machine, the command starts the same application server that is associated with the First steps console.</p>

Option	Link
<b>Stop the server</b>	<p>Calls the <b>stopServer</b> command.</p> <p>The location of the <b>stopServer</b> command varies per platform:</p> <ul style="list-style-type: none"> <li>•  <code>install_root/profiles/profile_name/bin/stopServer.sh server1</code></li> <li>•  <code>install_root\profiles\profile_name\bin\stopServer.bat server1</code></li> </ul>
<b>Administrative console</b>	<p>Opens the default browser to the <a href="http://localhost:9060/ibm/console">http://localhost:9060/ibm/console</a> Web address.</p> <p>When you have more than one application server on the same machine, the port varies. The First steps console starts the administrative console that is associated with the First steps console.</p>
<b>Profile creation wizard</b>	<p>Calls the <b>pctplatform</b> command.</p> <p>The command is in the <code>install_root/bin/ProfileCreator</code> directory. The name of the command varies per platform:</p> <ul style="list-style-type: none"> <li>• <code>pctAIX.bin</code></li> <li>• <code>pctHPUX.bin</code></li> <li>• 64-bit platforms: <code>pctHPUXIA64.bin</code></li> <li>• <code>pctLinux.bin</code></li> <li>• 64-bit platforms: <code>pct.bin</code> S/390 platforms: <code>pctLinux390.bin</code></li> <li>• Power platforms: <code>pctLinuxPPC.bin</code></li> <li>• <code>pctSolaris.bin</code></li> <li>•  <code>pctWindows.exe</code></li> <li>•  64-bit platforms: <code>pctWindowsIA64.exe</code></li> </ul>
<b>Samples Gallery</b>	<p>Opens the default browser to the <a href="http://localhost:9080/WSsamples">http://localhost:9080/WSsamples</a> Web address.</p> <p>If you do not install Samples, the option does not appear on the First steps console. If you do not install the Samples during the initial installation of the product, the option does not display on the First steps console. You can perform an incremental installation to add the Samples feature. After adding the Samples, the options displays on the First steps console.</p> <p>When you have more than one profile on the same machine, the port varies. The First steps console starts the Samples gallery that is associated with the First steps console.</p>
<b>Information center for WebSphere Application Server products</b>	<p>Opens the default browser to the online information center at the <a href="http://publib.boulder.ibm.com/infocenter/ws60help/index.jsp">http://publib.boulder.ibm.com/infocenter/ws60help/index.jsp</a> Web address.</p>
<b>Migration wizard</b>	<p>Calls the <b>migration</b> command.</p> <p>The location of the <b>migration</b> command is:</p> <ul style="list-style-type: none"> <li>•  <code>install_root/bin/migration.sh</code></li> <li>•  <code>install_root\bin\migration.bat</code></li> </ul> <p>The migration tools are also in the <code>/migration</code> folder on the product disc.</p>

## Using the Profile creation wizard

This topic describes how to create run-time environments for WebSphere Application Server. Each run-time environment is created within a *profile*. A profile is the set of files that define the run-time environment. The Profile creation wizard creates the profile for each run-time environment.

Before using the Profile creation wizard, install the core product files.

The Profile creation wizard is the wizard interface to the profile creation tool, `wasprofile`. See the description of the “`wasprofile` command” on page 34 for more information.

An error can occur when you have not provided enough system temporary space to create a profile. Verify that you have a minimum of 40 MB of temp space available before creating a profile.

You must have 200 MB of available disk space in the directory where you create an Application Server profile.

Manually verify that the required space for creating a profile is available on AIX. A known problem in the underlying InstallShield for Multiplatforms (ISMP) code prevents proper space checking on AIX systems at the time that the product disc was created.

**Important:** Concurrent profile creation is not supported at this time for one set of core product files. Concurrent attempts to create profiles result in a warning about a profile creation already in progress.

The installation procedure creates one profile named `default` for an application server named `server1`. You can use the Profile creation wizard to create more application server processes. For example, a second profile can allow two different teams in a department to test independently of one another using the same machine.



Each use of the Profile creation wizard or the **`wasprofile`** command line tool creates one profile.

1. Install the product to create the core product files.
2. Start the Profile creation wizard to create a new run-time environment.

Several ways exist to start the wizard.

One way to start the wizard is to issue the command directly from a command line.




The command is in the `install_root/bin/ProfileCreator` directory. The name of the command varies per platform:

- `./pctAIX.bin`
- `./pctHPUX.bin`
- 64-bit platforms: `pctHPUXIA64.bin`
- `./pctLinux.bin`
- 64-bit platforms: `pct.bin` S/390 platforms: `pctLinux390.bin`
- Power platforms: `pctLinuxPPC.bin`
- `./pctSolaris.bin`
-  `pctWindows.exe`
-  64-bit platforms: `pctWindowsIA64.exe`

Another way to start the Profile creation wizard is to select the wizard from the First steps console.

- a. Open a command window.
- b. Change directories to the `firststeps` directory in the installation root directory:

The installation root varies by platform:

- `./usr/IBM/WebSphere/AppServer/firststeps`
  - `.../opt/IBM/WebSphere/AppServer/firststeps`
  -  `C:\Program Files\IBM\WebSphere\AppServer\firststeps`
- c. Issue the **`firststeps`** command to start the console:
    -  `./firststeps.sh`
    -  `firststeps.bat`
  - d. Select the Profile creation wizard option on the console.

The Profile creation wizard is an InstallShield for Multiplatforms application. The wizard loads the Java 2 SDK and then displays its Welcome panel.

See the description of the “firststeps command” on page 25 for more information.

3. Create another stand-alone application server.

See “Using the Profile creation wizard to create an application server.”

The installation procedure creates a stand-alone application server during installation. However, you can use the Profile creation wizard to create additional stand-alone application servers.

See the *Administering applications and their environment* PDF to learn more about the command-line alternative method of creating a profile, and to see examples of using the command.

See Chapter 4, “Planning the installation (diagrams),” on page 11 for examples of configurations that you can create by creating profiles.

## Using the Profile creation wizard to create an application server

The Profile creation wizard can create an application server profile on any machine where the core product files exist.

Before using the Profile creation wizard, install the core product files.

The Profile creation wizard is the wizard interface to the profile creation tool, `wasprofile`. See the description of the “`wasprofile` command” on page 34 for more information.

An error can occur when you have not provided enough system temporary space to create a profile. Verify that you have a minimum of 40 MB of temp space available before creating a profile.

You must have 200 MB of available disk space in the directory where you create an Application Server profile.

Manually verify that the required space for creating a profile is available on AIX. A known problem in the underlying InstallShield for Multiplatforms (ISMP) code prevents proper space checking on AIX systems at the time that the product disc was created.

The Installation wizard creates an application server profile with a server named `server1`. You can create additional profiles. Each additional profile is an application server named `server1`.

This procedure describes creating an application server profile using the graphical user interface provided by the Profile creation wizard.

You can also use the **`wasprofile`** command to create an application server profile. See the description of the “`wasprofile` command” on page 34 for more information.



1. Start the Profile creation wizard to create a new run-time environment.

Several ways exist to start the wizard.

One way to start the wizard is to issue the command directly from a command line.

The command is in the `install_root/bin/ProfileCreator` directory. The name of the command varies per platform:

- `pctAIX.bin`
- `pctHPUX.bin`
- 64-bit platforms: `pctHPUXIA64.bin`
- `pctLinux.bin`
- 64-bit platforms: `pct.bin` S/390 platforms: `pctLinux390.bin`
- Power platforms: `pctLinuxPPC.bin`


- `pctSolaris.bin`
-  `pctWindows.exe`
-  64-bit platforms: `pctWindowsIA64.exe`

Another way to start the Profile creation wizard is to select the wizard from the First steps console.



a. Open a command window.

b. Change directories to the `firststeps` directory in the installation root directory:

The installation root varies by platform:

- `./usr/IBM/WebSphere/AppServer/firststeps`
- `.../opt/IBM/WebSphere/AppServer/firststeps`
-  `C:\Program Files\IBM\WebSphere\AppServer\firststeps`

c. Issue the **firststeps** command to start the console:

-  `./firststeps.sh`
-  `firststeps.bat`

d. Select the Profile creation wizard option on the console.

The Profile creation wizard is an InstallShield for Multiplatforms application. The wizard loads the Java 2 SDK and then displays its Welcome panel.

See the description of the “firststeps command” on page 25 for more information.

2. Click **Next** on the Welcome panel.

The wizard displays the Profile type selection panel.

3. Click **Next**.

The wizard displays the Profile name panel.

Each profile that you create must have a name. The name is the name of the folder that contains all of the files that define the run-time environment for the profile. When you have more than one profile, you can tell them apart at their highest level by this name.

4. Specify a name for the profile, then click **Next**.

**Profile naming guidelines:** The profile name can be any unique name with the following restrictions. Do not use any of the following characters when naming your profile:

- Spaces
- Illegal special characters that are not allowed within the name of a directory on your operating system, such as `*&?`
- Slashes (`/`) or (`\`)

Double-byte characters are allowed.

### The default profile

The first profile that you create on a machine is the default profile. The default profile is the default target for commands issued from the `bin` directory in the product installation root. When only one profile exists on a machine, every command works on the only server process in the configuration.

### Addressing a profile in a multi-profile environment

When two or more profiles exist on a machine, certain commands require that you specify the profile to which the command applies. These commands use the `-profileName` parameter to identify which profile to address. You might find it easier to use the commands that in the `bin` directory of each profile.

A command in the `profiles/profile_name/bin` directory has two lines. The first line sets the `WAS_USER_SCRIPT` environment variable for the command window. The variable sets up the command environment to address the profile. The second line calls the actual command in the `install_root/bin` directory.

The actual command queries the command shell to determine the calling profile and to autonomically address the command to the calling profile.

The wizard then displays the Profile directory panel.



- Accept the default directory or specify a non-default location, then click **Next**. Or click **Browse** to select a different location.

If you click **Back** and change the name of the profile, you must manually change the name on this panel when it displays again.

The wizard displays the Node and host name panel.

- Specify the characteristics for the application server, then click **Next**.

Use unique names for each application server that you create.

**Reserved names:** Avoid using reserved folder names as field values. The use of reserved folder names can cause unpredictable results. The following words are reserved:

- cells
- nodes
- servers
- clusters
- applications
- deployments

Field name	Default value	Constraints	Description
Node name	Name of your machine	Avoid using the reserved words.	Pick any name you want. To help organize your installation, use a unique name if you plan to create more than one application server on the machine.
Host name	DNS name of your machine	Addressable through your network.	Use the actual DNS name or IP address of your machine to enable communication with your machine. See additional information about the host name following this table.

#### Node name considerations:

**Windows** The installation directory path must be no longer than 60 characters.

#### Host name considerations:

The host name is the network name for the physical machine on which the node is installed. The host name must resolve to a physical network node on the server. When multiple network cards exist in the server, the host name or IP address must resolve to one of the network cards. Remote nodes use the host name to connect to and to communicate with this node. Selecting a host name that other machines can reach within your network is extremely important. Do not use the generic localhost identifier for this value.

If you define coexisting nodes on the same computer with unique IP addresses, define each IP address in a domain name server (DNS) look-up table. Configuration files for stand-alone Application Servers do not provide domain name resolution for multiple IP addresses on a machine with a single network address.

The value that you specify for the host name is used as the value of the `hostName` property in configuration documents for the stand-alone Application Server. Specify the host name value in one of the following formats:

- Fully qualified domain name servers (DNS) host name string, such as `xmachine.manhattan.ibm.com`
- The default short DNS host name string, such as `xmachine`
- Numeric IP address, such as `127.1.255.3`

The fully qualified DNS host name has the advantage of being totally unambiguous and also flexible. You have the flexibility of changing the actual IP address for the host system without having to change the Application Server configuration. This value for host name is particularly useful if you plan



to change the IP address frequently when using Dynamic Host Configuration Protocol (DHCP) to assign IP addresses. A format disadvantage is being dependent on DNS. If DNS is not available, then connectivity is compromised.

The short host name is also dynamically resolvable. A short name format has the added ability of being redefined in the local hosts file so that the system can run the Application Server even when disconnected from the network. Define the short name to 127.0.0.1 (local loopback) in the hosts file to run disconnected. A format disadvantage is being dependent on DNS for remote access. If DNS is not available, then connectivity is compromised.

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. A format disadvantage is that the numeric IP address is fixed. You must change the setting of the `hostName` property in Express configuration documents 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. Another format disadvantage is that you cannot use the node if the host is disconnected from the network.

After specifying application server characteristics, the wizard displays the Port value assignment panel.

7. Verify that the ports specified for the stand-alone application server are unique, then click **Next**.

**Windows** After specifying port assignments, the wizard displays the Windows service definition panel, if you are installing on a Windows platform.

8. **Windows** Choose whether to run the application server as a Windows service on a Windows platform and click **Next**.

Version 6 attempts to start Windows services for application server processes started by a **startServer** command. For example, if you configure an application server as a Windows service and issue the **startServer** command, the **wasservice** command attempts to start the defined service.

If you chose to install a local system service, you do not have to specify your user ID or password. If you create a specified user type of service, you must specify the user ID and the password for the user who is to run the service. The user must have *Log on as a service* authority for the service to run properly.

To perform this installation task, the user ID must not have spaces in its name. The ID must also belong to the administrator group and must have the advanced user rights *Act as part of the operating system* and *Log on as a service*. The Installation wizard grants the user ID the advanced user rights if it does not already have them, if the user ID belongs to the administrator group.

You can also create other Windows services after the installation is complete, to start other server processes. See “Automatically restarting server processes” on page 146 for more information.

The installation wizard shows which components are selected for installation in a pre-installation summary panel.

9. Click **Next** to create the application server or click **Back** to change the characteristics of the application server.

The wizard displays the Installation status panel that shows which components are installing.

When the installation is complete, the wizard displays the Profile creation is complete panel.

10. Click **Finish** to exit, then click **Profile creation wizard** on the First steps console to start the wizard again to create other application servers.

You can create an application server profile. The node within the profile has an application server named `server1`.

Refer to the description of the “`wasprofile` command” on page 34 to learn about creating this type of profile using a command instead of a wizard.

Deploy an application to get started!

## Deleting a profile

This topic describes how to manually delete a profile.

Before using the manual procedure to remove a profile, try the **wasprofile** command with the **-delete** option. For example, issue one of the following commands:

```
UNIX  
./wasprofile.sh -delete  
                -profileName profile_name | -profilePath profile_path
```

```
Windows  
wasprofile.bat -delete  
               -profileName profile_name | -profilePath profile_path
```

See “wasprofile command.”

If the command does not work, use this procedure to delete the profile.

This procedure describes how to manually delete a profile when the **wasprofile -delete** command results in the following message:

INSTCONFFAILED: Cannot delete profile

1. Delete the *profiles\_install\_root/profile\_name* directory.
2. If the *install\_root/properties/profileRegistry.xml* file exists, edit the file in a flat-file editor to delete the entry for the profile, if the entry is present.

The entry resembles the following example:

```
<profile isDefault="true"  
        name="BadProfile"  
        path="E:\IBM\WebSphere\AppServer\profiles\BadProfile"  
        template="E:\IBM\WebSphere\AppServer\profileTemplates\default"/>
```

3. **UNIX** Compare the two batch files, *install\_root/properties/fsdb/\_was\_profile\_default/default.sh* and *install\_root/properties/fsdb/bad\_profile\_name.sh*.  
If the files are identical, delete the *install\_root/properties/fsdb/\_was\_profile\_default* directory and the *install\_root/properties/fsdb/bad\_profile\_name.sh* file.  
If the files are not identical, delete only the *install\_root/properties/fsdb/bad\_profile\_name.sh* file.
4. **Windows** Compare the two batch files, *install\_root\properties\fsdb\\_was\_profile\_default\default.bat* and *install\_root\properties\fsdb\bad\_profile\_name.bat*.  
If the files are identical, delete the *install\_root\properties\fsdb\\_was\_profile\_default* directory and the *install\_root\properties\fsdb\bad\_profile\_name.bat* file.  
If the files are not identical, delete only the *install\_root\properties\fsdb\bad\_profile\_name.bat* file.

See the description of the “wasprofile command” to learn more about the command-line method of working with profiles.

See “Using the Profile creation wizard” on page 28 for more information about creating profiles with the Profile creation wizard.

---

## wasprofile command

The **wasprofile** command line tool creates all Application Server run-time environments in Version 6. The command creates a profile, which is the set of files that define the run-time environment for a stand-alone Application Server.

The **wasprofile** command is also referred to as the *profile creation tool*.

## Introduction to terms that describe Version 6 profiles

The **wasprofile** command creates the run-time environment for a WebSphere Application Server process in a set of files called a *profile*. The profile defines the run-time environment and includes all of the files that the server processes in the run-time environment can change. The *profile creation tool* and its graphical user interface, the *Profile creation wizard*, are the only ways to create run-time environments in V6.

The *Profile creation wizard* is an InstallShield for Multiplatforms (ISMP) application. You can use the wizard to enter most of the parameters that are described in this topic. Some parameters, however, require you to use the **wasprofile** command. You must use the **wasprofile** command to delete a profile, for instance, because the Profile creation wizard does not provide a deletion function.

However, the Profile creation wizard also performs tasks that the wasprofile command does not. For instance, the wizard can create a Windows service for each profile that it creates. It can also assign non-conflicting ports based on previous Version 6 port assignments.

### Core product files

The core product files are the shared product binaries. The binary files are shared by all profiles.

The directory structure for V6 has two major divisions of files in the installation root directory for the product:

- The core product files are shared product binary files that do not change unless you install a refresh pack, a fix pack, or an interim fix. Some log information is also updated.
- The *profiles* directory is the default directory for creating profiles. The configuration for every defined Application Server process is within the *profiles* directory unless you specify a new directory when you create a profile. These files change as often as you create a new profile, reconfigure an existing profile, or delete a profile.

All of the folders except for the *profiles* directory and a few others such as the *logs* directory and the *properties* directory do not change unless you install service fixes. The *profiles* directory, however, changes each time you add, change, or delete a profile. The *profiles* directory is the default repository for profiles. However, you can put a profile anywhere on the machine provided there is enough available disk space.

If you put a profile in another existing folder in the installation root directory, a risk exists that the profile might be affected by the installation of a service fix that applies maintenance to the folder. Use a directory outside of the installation root directory when using a directory other than the *profiles* directory for creating profiles.

### WebSphere Application Server profile

The **wasprofile** command line tool defines each Application Server instance of a Version 6 product.

You must run the wizard or the command line tool each time that you want to create a stand-alone Application Server. A need for more than one stand-alone Application Server on a machine is common.

Administration is greatly enhanced when using V6 profiles instead of multiple product installs. Not only is disk space saved, but updating the product is simplified when you only maintain a single set of product core files. Also, creating new profiles is faster and less prone to error than full product installs, allowing a developer to create new disposable profiles of the product for development and testing.

You can run the Profile creation wizard or the profile creation tool to create a new Application Server environment on the same machine as an existing one. Simply define unique characteristics (such as profile name and node name) for the new profile. Each profile has its own administrative console and administrative scripting interface. Each Application Server process shares all run-time scripts, libraries, the Software Development Kit, and other core product files.

The installation program for the base WebSphere Application Server product uses the profile creation tool to create an Application Server profile named default.

## Installed file set

You decide where to install the files that define a profile. The default location is in the `profiles` directory in the installation root directory. But you can change the location on the Profile creation wizard or in a parameter when using the command line tool. For example, assume that you create two profiles on a Linux platform with host name `devhost1`. The profile directories resemble the following example if you do not relocate them:

```
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile02
```

Suppose that you specify a different directory, such as `/opt/profiles`, for the profile directory field in the wizard. The profile directories resemble the following example:

```
/opt/profiles/devhost1Profile01
/opt/profiles/devhost1Profile02
```

The following directories exist within a profile. This example assumes that a profile named `devhost1Profile01` exists:

```
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/bin
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/config
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/etc
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/firststeps
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/installableApps
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/installedApps
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/installedConnectors
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/installedFilters
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/logs
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/properties
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/samples
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/temp
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/tranlog
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01/wstemp
```

## The profile repository

The profile repository is the default location of profile-related metadata. The repository is the default location for new profiles, which is often referred to as the profiles installation root directory.

However, you can decide where to install a profile. The default location of the profile repository is the `install_root/profiles` directory. In the earlier example, creating two profiles on a Linux platform with host name `devhost1` results in the following example directories in the profile repository:

```
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile01
/opt/IBM/WebSphere/AppServer/profiles/devhost1Profile02
```

When you specify a directory, such as `/opt/profiles`, the profiles are no longer in the default repository, which is not a problem. For example, the following locations are valid:

```
/opt/profiles/devhost1Profile01
/opt/profiles/devhost1Profile02
```

## Location of the command file

The command file is located in the `install_root/bin` directory. The command file is a script named `wasprofile.sh` for Linux and UNIX platforms or `wasprofile.bat` for Windows platforms.

The Profile creation wizard is the graphical user interface to the command line tool. The file name of the command that calls the Profile creation wizard varies per operating system platform. See “Using the Profile creation wizard” on page 28 for more information.

## Logging

The **wasprofile** command creates a log for every profile that it creates. The logs are in the *install\_root/logs/wasprofile* directory. The files are named in this pattern:  
*wasprofile\_create\_profile\_name.log*.

The command also creates a log for every profile that it deletes. The logs are in the *install\_root/logs/wasprofile* directory. The files are named in this pattern:  
*wasprofile\_delete\_profile\_name.log*.

## Required disk space

Manually verify that the required space for creating a profile is available on AIX. A known problem in the underlying InstallShield for Multiplatforms (ISMP) code prevents proper space checking on AIX systems at the time that the product disc was created.

An error can occur when you have not provided enough system temporary space to create a profile. Verify that you have a minimum of 40 MB of temp space available before creating a profile.

You must have 200 MB of available disk space in the directory where you create an Application Server profile.

## Concurrent profile creation

**Important:** Concurrent profile creation is not supported at this time for one set of core product files. Concurrent attempts to create profiles result in a warning about a profile creation already in progress.

## Entering lengthy commands on more than one line

The length of the **wasprofile** command can exceed the normal shell window limit for one line of 256 characters. If your command is longer than the limit, issue the command on multiple lines by ending a line with a backward slash, pressing **Enter**, and continuing the command on the next line.

For example, on a Solaris system, the following command requires input on multiple lines:

```
./wasprofile.sh \  
-create -profileName bladetcb6profile \  
-profilePath /usr/IBM/WebSphere/AppServer/profiles/bladetcb6profile \  
-templatePath /usr/WebSphere/AppServer/profileTemplates/default \  
-nodeName bladetcb6node \  
-cellName bladetcb6Cell \  
-hostName bladetcb6.rtp.raleigh.ibm.com
```

Omit the line continuation character from the last line to signal the end of the command to the operating system.

## wasprofile.sh command syntax

List existing profiles:

```
# ./wasprofile.sh -listProfiles  
[-debug]
```

Delete profiles:

```
# ./wasprofile.sh -delete
    -profileName profile_name | -profilePath profile_path
    [-debug]
```

#### Create new profiles:

```
wasprofile.sh -create
    -profileName profile_name
    -profilePath fully_qualified_profile_path
    -templatePath template_path
    -nodeName node_name
    -cellName cell_name
    -hostName host_name
    -server iSeries_server_name
    [-startingPort starting_port | -portsFile filepath]
    -winserviceCheck true | false
    -winserviceAccountType specifieduser | localsystem
    -winserviceUserName yourusername
    -winservicePassword yourpassword
    -winserviceStartupType manual | automatic | disabled
    [-debug]
```

#### Get name of existing profile from path:

```
# ./wasprofile.sh -getName
    -profilePath profile_path
    [-debug]
```

#### Get path of existing profile from name:

```
# ./wasprofile.sh -getPath
    -profileName profile_name
    [-debug]
```

#### Check the integrity of the profile registry:

```
# ./wasprofile.sh -validateRegistry
    [-debug]
```

#### Check the integrity of the profile registry, removing profiles that are not found:

```
# ./wasprofile.sh -validateAndUpdateRegistry
    [-backup file_name]
    [-debug]
```

## wasprofile.bat command syntax

#### List existing profiles:

```
wasprofile.bat -listProfiles
    [-debug]
```

#### Delete profiles:

```
wasprofile.bat -delete
    -profileName profile_name | -profilePath profile_path
    [-debug]
```

#### Create new profiles:

```
wasprofile.bat -create
    -profileName profile_name
    -profilePath fully_qualified_profile_path
    -templatePath template_path
    -nodeName node_name
    [-cellName cell_name]
    -hostName host_name
    -server iSeries_server_name
    [-startingPort starting_port | -portsFile filepath]
    -winserviceCheck true | false
```

```
-winserviceAccountType specifieduser | localsystem
-winserviceUserName yourusername
-winservicePassword yourpassword
-winserviceStartupType manual | automatic | disabled
[-debug]
```

When the `-startingPort` parameter is not used, the profile creation tool uses the default port settings specified in the `serverindex.xml` file.

Get name of existing profile from path:

```
wasprofile.bat -getName
                -profilePath fully_qualified_profile_path
                [-debug]
```

Get path of existing profile from name:

```
wasprofile.bat -getPath
                -profileName profile_name
                [-debug]
```

Check integrity of profile registry:

```
wasprofile.bat -validateRegistry
                [-debug]
```

Check integrity of profile registry, removing unfound profiles:

```
wasprofile.bat -validateAndUpdateRegistry
                [-backup file_name]
                [-debug]
```

## Parameters

Supported arguments include:

### **-augment**

Refreshes or augments the given profile using the template in the `templatePath` parameter.

### **-backup** *file\_name*

Backs up the profile registry file to a file with the file name specified.

### **-cellname** *file\_name*

Specifies the cell name of the profile.

This is an optional parameter for WebSphere Application Server.

If you omit the parameter, a default cell name is assigned.

### **-create**

Creates the profile.

### **-debug**

Turns on the debug function of the Ant utility, which the **wasprofile** command uses.

### **-delete**

Deletes the profile.

### **-getName**

Gets the name for a profile registered at a given file system path. Requires the `-profilePath` parameter.

### **-getPath**

Gets the file system location for a profile of a given name. Requires the `-profileName` parameter.

### **-hostname** *host\_name*

Specifies the host name where you are creating the profile. This should match the host name that you specified during installation of the initial product.



**-listProfiles**

Lists all defined profiles.

**-nodeName** *node\_name*

Specifies the node name for the node that is created with the new profile. Use a unique value or on the machine. Each profile that shares the same set of product binaries must have a unique node name.

**-portsFile** *file\_path*

An optional parameter that specifies the path to a file that defines port settings for the new profile. When omitted, the wasprofile tool looks for the *install\_root /profileTemplates/profile\_type /actions/portsUpdate/bin/portdef.props* file.

Do not use this parameter when using the startingPort parameter.

**-profileName** *profile\_name*

Specifies the name of the profile. Use a unique value when creating a profile. Each profile that shares the same set of product binaries must have a unique name.

**-profilePath** *profile\_path*

Specifies the fully qualified path to the profile.

**Windows** If the fully qualified path contains spaces, enclose the value in quotation marks.

**-startingPort** *startingPort*

Specifies the starting port number for generating all ports for the profile. If not specified, the **wasprofile** command uses default ports specified in the *serverindex.xml* file.

**-templatePath** *template\_path*

Specifies the path to the templates in the shared binaries.

**-validateAndUpdateRegistry** *registry\_file backup\_file*

Checks all of the profiles that are listed in the profile registry to see if the profiles are present on the file system. Removes any missing profiles from the registry. Returns a list of the missing profiles that were deleted from the profile.

**-validateRegistry** *registry\_file*

Checks all of the profiles that are listed in the profile registry to see if the profiles are present on the file system. Returns a list of missing profiles.

**Windows** **-winserviceAccountType** *type\_of\_owner\_account*

The type of the owner account of the Windows service created for the profile can be either *specifieduser* or *localsystem*. The Windows service can run under the local account of the user who is creating the profile.

**Windows** **-winserviceCheck** *value*

The value can be either *true* or *false*. Specify *true* to create a Windows service for the server process that is created within the profile. Specify *false* to not create the Windows service.

**Windows** **-winservicePassword** *yourpassword*

Specify the password for the specified user or the local account that is to own the Windows service.

**Windows** **-winserviceStartupType** *startup\_type*

Possible *startup\_type* values are:

- manual
- automatic
- disabled

See “WASService command” on page 148 for more information about Windows services.

**Windows** **-winserviceUserName** *user\_ID*

Specify your user ID so that Windows can verify you as an ID that is capable of creating a Windows



service. Your user ID must belong to the administrator group and have the following advanced user rights, *Act as part of the operating system* and *Log on as a service*

## Use case scenarios

Use cases are a description of common tasks for which the tool is used.

### Scenario: Deleting a profile

The following command is on more than one line for clarity. Enter the command on one line to delete the profile named shasti:

```
wasprofile.sh -delete
              -profileName shasti
```

### Scenario: Using predefined port numbers

When you use the wasprofile tool without the `-startingPort` parameter, the tool uses the `/profileTemplates/profile_type/actions/portsUpdate/bin/portdef.props` file to set the initial ports.

### Example of using the `-portsFile` parameter

Copy the file, edit the port settings, and use your copy by using the `-portsFile` parameter as shown in the following example:

```
wasprofile.bat
  -create
  -profileName Wow_Profile
  -profilePath
    C:\ExpressV6\IBM\WebSphere\AppServer\profiles\Wow_Profile
  -templatePath
    C:\ExpressV6\IBM\WebSphere\AppServer\profileTemplates\default
  -nodeName Wow_node
  -cellName Wow_cell
  -hostName 1oyAllen
  -portsFile C:\temp\ports\portdef.props
```

Suppose that the `portdef.props` file has the following values:

```
WC_defaulthost=39080
WC_adminhost=39060
WC_defaulthost_secure=39443
WC_adminhost_secure=39043
BOOTSTRAP_ADDRESS=32809
SOAP_CONNECTOR_ADDRESS=38880
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=39401
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=39403
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=39402
ORB_LISTENER_ADDRESS=39100
DCS_UNICAST_ADDRESS=39353
SIB_ENDPOINT_ADDRESS=37276
SIB_ENDPOINT_SECURE_ADDRESS=37286
SIB_MQ_ENDPOINT_ADDRESS=35558
SIB_MQ_ENDPOINT_SECURE_ADDRESS=35578
```

As you run the command, messages similar to the following appear in the output stream:

```
replaceRegExpAllInstancesOfGivenTokenWithGivenValueForTheGivenFile:
  [echo] File C:\ExpressV6\IBM\WebSphere\AppServer\profiles\
  Wow_Profile/config/templates/default/serverentry-template.xml:
  setting CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS to 39403
...
replaceRegExpAllInstancesOfGivenTokenWithGivenValueForTheGivenFile:
  [echo] File C:\ExpressV6\IBM\WebSphere\AppServer\profiles\
  Wow_Profile/config/templates/default/serverentry-template.xml:
  setting CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS to 39402
...
```

The resulting serverindex.xml file looks similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<serverindex:ServerIndex xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
...
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="BOOTSTRAP_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="IBMmachine" port="32809"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="SOAP_CONNECTOR_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="IBMmachine" port="38880"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="SAS_SSL_SERVERAUTH_LISTENER_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="IBMmachine" port="39401"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="IBMmachine" port="39403"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="IBMmachine" port="39402"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="WC_adminhost">
    <endPoint xmi:id="EndPoint_..." host="*" port="39060"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="WC_defaulthost">
    <endPoint xmi:id="EndPoint_..." host="*" port="39080"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="DCS_UNICAST_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="IBMmachine" port="39353"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="WC_adminhost_secure">
    <endPoint xmi:id="EndPoint_..." host="*" port="39043"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="WC_defaulthost_secure">
    <endPoint xmi:id="EndPoint_..." host="*" port="39443"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="SIB_ENDPOINT_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="*" port="37276"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="SIB_ENDPOINT_SECURE_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="*" port="37286"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="SIB_MQ_ENDPOINT_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="*" port="35558"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="SIB_MQ_ENDPOINT_SECURE_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="*" port="35578"/>
  </specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_..."
                    endPointName="ORB_LISTENER_ADDRESS">
    <endPoint xmi:id="EndPoint_..." host="IBMmachine" port="39100"/>
  </specialEndpoints>
</serverEntries>
</serverindex:ServerIndex>
```

The **wasprofile** command creates a copy of the current portdefs.props file in the `install_root\profiles\profile_name\logs` directory.

Do not use the portsFile parameter when using the startingPort parameter. The two parameters are mutually exclusive.

### Scenario: Incrementing default port numbers from a starting point

The **wasprofile** command can assign port numbers based on a starting port value that you give on the command line, using the `-startingPort` parameter. The tool assigns port numbers sequentially from the starting port number value.

The order of port assignments is arbitrary. Predicting assignments is not possible.

For example, ports created with `-startingPort 20002` would appear similar to the following example:

#### Assigned ports for an Application Server profile

```
WC_defaulthost=20002
WC_adminhost=20003
WC_defaulthost_secure=20004
WC_adminhost_secure=20005
BOOTSTRAP_ADDRESS=20006
SOAP_CONNECTOR_ADDRESS=20007
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=20008
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=20009
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=20010
ORB_LISTENER_ADDRESS=20011
DCS_UNICAST_ADDRESS=20012
SIB_ENDPOINT_ADDRESS=20013
SIB_ENDPOINT_SECURE_ADDRESS=20014
SIB_MQ_ENDPOINT_ADDRESS=20015
SIB_MQ_ENDPOINT_SECURE_ADDRESS=20016
```

#### Example of startingPort parameter use

The following example of using the **wasprofile** command creates ports from an initial value of 20002, with the content shown in the previous example:

```
wasprofile.bat -create
               -profileName shasti
               -profilePath G:\shasti\WebSphere
               -templatePath template_path
               -nodeName W2K03
               -cellName W2K03_Cell01
               -hostName planetnt
               -startingPort 20002
```

### Scenario: Setting up and using the profile environment

Most tasks that you perform in a profile are done using the `-profileName` attribute on the command line tools that you use. Such a scenario might be:

1. Create the server process using the `install_root/bin/wasprofile.sh` (or `wasprofile.bat`) script from the original installation. Assume that you create the Profile02 profile.
2. In that command window or in another, change directories to the `/bin` directory of the new server process.
3. Establish a temporary override for the normal WebSphere Application Server environment by using the `-profileName` attribute on a command you issue. In the same window, start server1 by changing directories to the `install_root/bin` directory of the original installation and issuing the command. There is no such command in the `/bin` directory of the server process:

```
startServer.sh server1 -profileName Profile02
```

4. Notice the changes in the environment. Display all of the ports for the machine to see the ports that you set for the server process. For example, in a Linux bash shell or in the command window on a Windows platform, type:

```
netstat -a
```

5. Open a browser window and point it at the port defined for the HTTP\_TRANSPORT\_ADMIN port of the new process. For example, suppose the setting is HTTP\_TRANSPORT\_ADMIN=20003. Open the administrative console for server1 by pointing your browser at:

```
http://hostname_orIP_address:20003/ibm/console/
```

## Scenario: Profile creation for a non-root user

Two methods exist for a non-root user to create a profile:

- The root user creates the profile and assigns ownership to the non-root user.
- A non-root user creates a profile after getting write permission to the appropriate directories.

**Remember:** An ease-of-use limitation exists for non-root users who create profiles. Mechanisms within the Profile creation wizard that suggest unique names and port values are disabled for non-root users. The non-root user must change the default field values in the Profile creation wizard for the profile name, node name, and port assignments. Consider assigning non-root users a range of values for each of the fields. You can assign responsibility to the non-root profilers for adhering to their proper value ranges and for maintaining the integrity of their own definitions.

**Root creates the profile and assigns ownership to a non-root user:** The root user can create a profile and assigns ownership of the profile directory to a non-root user.

1. The root user creates the profile with the following command:

```
./wasprofile.sh -create -profileName profile01 -profilePath install_root/profiles/profile01
```

2. The root user changes ownership of the directory for the profile to the non-root user with the following command:

```
chown -R user1 install_root/profiles/profile01
```

**A non-root user creates the profile (advanced option):** The root user can grant write permission to the appropriate files and directories to a non-root user. The non-root user can then create the profile. You can create a group for users who are authorized to create profiles. Or you can give everyone the ability to create profiles. The following example shows how to create a group that is authorized to create profiles.

1. Log on to the Application Server system as root.
2. Create the profilers group that you can use to create profiles.
3. Create a user named user1 to create profiles.
4. Add users root and user1 to the profilers group.
5. Log off and back on as root to pick up the new group.
6. As root, use operating system tools to change file permissions.

The following example assumes that the installation root directory is `/opt/IBM/WebSphere/AppServer`:

```
mkdir /opt/IBM/WebSphere/AppServer/logs/wasprofile
chgrp profilers /opt/IBM/WebSphere/AppServer/logs/wasprofile
chmod g+wr /opt/IBM/WebSphere/AppServer/logs/wasprofile
chgrp profilers /opt/IBM/WebSphere/AppServer/properties
chmod g+wr /opt/IBM/WebSphere/AppServer/properties
chmod g+wr /opt/IBM/WebSphere/AppServer/properties/profileRegistry.xml
```

You might have to change the permissions on additional `/opt/IBM/WebSphere/AppServer` directories if you encounter permission problems.

7. The non-root user who belongs to the profilers group can then create a profile in any directory to which the non-root user has write permission.

If the non-root user does not have write access to any directories, it is up to the root user to change that situation. If the non-root user does not have write access to the /tmp directory, it is up to the root user to change that as well.

The commands listed in step 6 give users assigned to the profilers group the ability to write to the /opt/IBM/WebSphere/AppServer/logs/wasprofile directory and to the /opt/IBM/WebSphere/AppServer/properties directory. It is not necessary to write to any other directories in the installation root of your WebSphere Application Server product.

Have non-root users create a profiles directory in their own area, not in the installation root directory of the product.

---

## Using the installation verification test

This topic describes how to use the installation verification test (IVT). The IVT verifies that the installation of the application server profile was successful. A *profile* consists of files that define the run-time environment for an application server. Each profile has its own IVT command.

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.

After installing the product, the Installation wizard displays a prompt for starting the First steps console.



Installation verification is the first option on the First steps console.

The test consists of starting and monitoring the application server during its start up.

1. Select **Installation verification** on the First steps console after installing the product.

You can also start the First steps console from the command line, as described in “firststeps command” on page 25.

You can also start the “ivt command” on page 46 directly from the bin directory of the profile:

-  `install_root/profiles/default/bin/ivt.sh`
-  `install_root\profiles\default\bin\ivt.bat`

If you create additional profiles, the ivt script location is within the *profile\_home*/bin directory.

2. Observe the results in the First steps status window.

The log file for installation verification is the *install\_root/profiles/default/logs/ivtClient.log* file. If you create additional profiles, the file path is *profile\_home/logs/ivtClient.log*.

The IVT provides the following useful information about the application server:

- The application server name
- The name of the profile
- The profile file path
- The type of profile
- The node name
- The current encoding
- The port number for the administrative console
- Various informational messages that include the location of the SystemOut.log file and how many errors are listed within the file
- A completion message

As the IVT starts the application server on a Windows platform, the IVT attempts to start the Windows service for the application server, if a Windows service exists. This is true even though the Windows service might have a manual startup type.

See “Automatically restarting server processes” on page 146 for more information.

3. If the log shows that errors occurred during the installation verification, correct the errors and run the IVT again. If necessary, create a new profile after correcting the error, and run the IVT on the new profile.

The IVT program starts the server process automatically if the server is not running. Once the server initializes, the IVT runs a series of verification tests. The tool displays pass or fail status in a console window. The tool also logs results to the `profile_home/logs/ivtClient.log` file. As the IVT verifies your system, the tool reports any detectable errors in the `SystemOut.log` file.

Return to the *Installing your application serving environment* PDF to continue.

## ivt command

The `ivt` command starts the installation verification test (IVT) program. The IVT verifies that the installation of the application server profile was successful. A *profile* consists of files that define the run-time environment for an application server. Each profile has its own IVT command.

The IVT program starts the application server automatically if the server process is not already running. After the server process initializes, the IVT runs a series of verification tests and displays pass or fail status in a console window.

The IVT program scans the `SystemOut.log` file for errors and verifies core functionality of the profile.

You can start the IVT program from the command line or from the First steps console.

### Location of the command file

Installing the product creates a default profile for the `server1` application server. The location of the installation verification test program for the default profile is:

- **UNIX** `install_root/profiles/default/bin/ivt.sh`
- **Windows** `install_root\profiles\default\bin\ivt.bat`

The location of the `ivt.sh` or `ivt.bat` script for any profile is:

- **UNIX** `install_root/profiles/profile_name/bin/ivt.sh`
- **Windows** `install_root\profiles\profile_name\bin\ivt.bat`

### Parameters

The following parameters are associated with this command.

#### **server\_name**

Required parameter that identifies the name of the server process, such as `server1`.

#### **profile\_name**

Required parameter that identifies the name of the profile that contains the server definition.

#### **-p server\_port\_number**

Optional parameter that identifies the default\_host port when the port is not 9080, which is the default.

#### **-host machine\_host\_name**

Optional parameter that identifies the host machine of the profile to test. The default is `localhost`.

### Syntax for the ivt command

Use the following syntax for the command:

- **UNIX** `install_root/profiles/profile_name/bin/ivt.sh`
- **Windows** `install_root\profiles\profile_name\bin\ivt.bat`

## Logging

The `ivt` command logs results to the `install_root/profiles/profile name/logs/ivtClient.log` file.

## Example

The following examples test the `server1` process in the `profile01` profile on the `myhost` machine using the `default_host` on port `9081`.

### Windows

```
ivt.bat server1 profile01 -p 9081 -host myhost
```

### UNIX

```
ivt.sh server1 profile01 -p 9081 -host myhost
```





---

## Chapter 6. Configuring ports

This topic discusses configuring ports, particularly in coexistence scenarios.

1. Review “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 with Version 6.

2. You can change port settings on the port assignment panel while using the Installation wizard or the Profile creation wizard.

See “Using the Profile creation wizard” on page 28 and the *Installing your application serving environment* PDF for more information.

3. After installation, edit the `profiles_install_root/profile_name/config/cells/cell_name/nodes/node_name/serverindex.xml` file to change the port settings, or use scripting to change the values.

See the *Administering applications and their environment* PDF for more information.

---

### 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 6.

#### Version 6 port numbers

Table 3. Port definitions for WebSphere Application Server Version 6

Port name	WebSphere Application Server	File
	Value	
HTTP_TRANSPORT	9080	
HTTP Admin Console Port (HTTP_TRANSPORT_ADMIN)	9060	serverindex.xml and virtualhosts.xml
HTTPS Transport Port (HTTPS_TRANSPORT)	9443	
HTTPS Admin Console Secure Port (HTTPS_TRANSPORT_ADMIN)	9043	

Table 3. Port definitions for WebSphere Application Server Version 6 (continued)

Port name	WebSphere Application Server	File
	Value	
BOOTSTRAP_ADDRESS	2809	
SOAP_CONNECTOR_ADDRESS	8880	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9401	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9403	
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9402	
ORB_LISTENER_ADDRESS	9100	
DCS_UNICAST_ADDRESS	9353	serverindex.xml
SIB_ENDPOINT_ADDRESS	7276	
SIB_ENDPOINT_SECURE_ADDRESS	7286	
SIB_MQ_ENDPOINT_ADDRESS	5558	
SIB_MQ_ENDPOINT_SECURE_ADDRESS	5578	
Internal JMS Server (JMSSERVER_SECURITY_PORT)	5557	
DRS_CLIENT_ADDRESS	7873	
IBM HTTP Server Port	80	virtualhosts.xml, plugin-cfg.xml, and <i>IHSinstall_root/conf/</i> <i>httpd.conf</i>
IBM HTTP Server Admin Port	8008	<i>IHSinstall_root/conf/</i> admin.conf
NODE_MULTICAST_IPV6_DISCOVERY_ADDRESS	5001	serverindex.xml

### Version 5.x port numbers

Table 4. Port definitions for WebSphere Application Server Version 5.1

Port name	WebSphere Application Server	File
	Value	
HTTP_TRANSPORT	9080	
HTTPS Transport Port (HTTPS_TRANSPORT)	9443	
HTTP Admin Console Port (HTTP_TRANSPORT_ADMIN)	9090	server.xml and virtualhosts.xml
HTTPS Admin Console Secure Port (HTTPS_TRANSPORT_ADMIN)	9043	
Internal JMS Server (JMSSERVER_SECURITY_PORT)	5557	server.xml
JMSSERVER_QUEUED_ADDRESS	5558	serverindex.xml
JMSSERVER_DIRECT_ADDRESS	5559	
BOOTSTRAP_ADDRESS	2809	serverindex.xml
SOAP_CONNECTOR_ADDRESS	8880	serverindex.xml
DRS_CLIENT_ADDRESS	7873	serverindex.xml
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	0	serverindex.xml

Table 4. Port definitions for WebSphere Application Server Version 5.1 (continued)

Port name	WebSphere Application Server	File
	Value	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	0	serverindex.xml
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	0	
ORB_LISTENER_ADDRESS	0	serverindex.xml
IBM HTTP Server Port	80	virtualhosts.xml, plugin-cfg.xml, and <i>IHSinstall_root/conf/httpd.conf</i>
IBM HTTP Server Admin Port	8008	<i>IHSinstall_root/conf/admin.conf</i>

### 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.

Table 5. Port definitions for WebSphere Application Server V4.0.x

Port name	Value	Advanced Edition	IBM WebSphere Business Integration Server Foundation Edition	Advanced Single Server Edition
		File		
bootstrapPort	900			
lsdPort	9000	admin.config	admin.config	
LSDSSLPort	9001			
HTTP transport port	9080			
HTTPS transport port	9443			server-cfg.xml
Admin Console HTTP transport port	9090	database	database	
ObjectLevelTrace	2102			
diagThreadPort	7000			



---

## Chapter 7. Communicating with Web servers

The WebSphere Application Server works with a Web server to route requests for dynamic content, such as servlets, from Web applications. The Web servers are necessary for directing traffic from browsers to the applications that run in WebSphere Application Server. The Web server plug-in uses the XML configuration file to determine whether a request is from the Web server or the Application Server.

The Web server plug-ins for distributed platform Web servers are provided on a separate CD from the WebSphere Application Server products. A Web Server Plug-in Installation Wizard is also provided on that CD. The *Installing your application serving environment* PDF describes how to install a Web server plug-in and create a Web server definition.

1. Install your Web server if it is not already installed. See the installation information provided with your Web server.
2. Ensure that your Web server is configured to perform operations required by Web applications, such as GET and POST. Typically, this involves setting a directive in the Web server configuration file (such as the `httpd.conf` file for an IBM HTTP Server). Refer to the Web server documentation for instructions. If an operation is not enabled when a servlet or JSP file requiring the operation is accessed, an error message displays, such as this one from the IBM HTTP Server:

HTTP method POST is not supported by this URL.

3. Use the Plug-in Installation wizard to install the appropriate plug-in file to your Web server and run the script `configureWeb_server_name` created by the wizard to create and configure the Web server definition in the WebSphere configuration repository. The following substeps are performed during the plug-in installation process. See the Plug-in Installation Roadmap for additional information.
  - a. A Web server definitions is created. You can also use either use the administrative console or use the `ConfigureWebServerDefintion.jacl` script to create a Web server definition.

If you use the administrative console:

    - 1) Select the node that was created in the preceding step, and in the Server name field, enter the local name of the Web server for which you are creating a Web server definition.
    - 2) Use the wizard to complete the Web server definition.
  - b. An application or modules are mapped to a Web server. If an application that you want to use with this Web server is already installed, the application is automatically mapped to the Web server. If the application is not installed, select this Web server during the Map modules to servers step of the application installation process.
  - c. Master repository is updated and saved.
4. **Optional:** Configure the plug-in. Use either the administrative console, or issue the `GenPluginCfg` command to create your `plugin-cfg.xml` file.

When setting up your Web server plug-in, you must decide whether or not to have the configuration automatically generated in response to a configuration change. When the Web server plug-in configuration service is enabled and any of the following conditions occur, the plug-in configuration file is automatically generated:

- When the Web server is created or saved.
- When an application is installed.
- When an application is uninstalled.
- When the virtual host definition is updated

Generating or regenerating the configuration file might take a while to complete. After it finishes, all objects in the administrative cell use their newest settings, which the Web server can access. If the Application Server is on the same physical machine as the Web server, the regeneration usually takes about 30 to 60 seconds to complete. The regeneration takes longer if they are not both on the same machine.

**Important:** When the plug-in configuration file is first generated, it does not include `admin_host` on the list of virtual hosts. The *Installing your application serving environment* PDF describes how to add it to the list.

To use the administrative console:

- a. Select **Servers > Web Servers > *webserver* > plug-in properties**.
- b. Select **Automatically generate plug-in configuration file** or click on one or more of the following topics to manually configure the `plugin-cfg.xml` file:
  - Caching
  - Request and response
  - Request routing
  - Service

Web server plug-in configuration properties maps each property to one of these topics.

- c. Click **OK**.
  - d. You might need to stop the application server and then start the application server again to enable the Web server to locate the `plugin-cfg.xml` file.
5. If you want to use Secure-socket layer (SSL) with this configuration, use the plug-in's installation wizard to install the appropriate GSKIT installation image file on your workstation. See the *Securing applications and their environment* PDF for information on how to configure GSKIT.
  6. If you want to enable the Web server plug-in to use private headers, define an SSL configuration repertoire that defines a trust file. Then in the administrative console, select **Application servers > server1 > Web Container Settings > Web Container Transport Chains > *transport\_chain* > SSL Inbound Channel (SSL\_2)** and specify this repertoire for that transport chain. If you try to use private headers without setting up an SSL configuration repertoire that does not include a trust file definition, the private headers will be ignored. If the private headers are ignored, the application server might not locate the requested application.

After you enable the use of private headers, the transport chain's SSL inbound channel trusts all private headers it receives. Therefore, you must ensure that all paths to the transport chain's SSL inbound channel are trusted.

7. Tune your Web server with Web server tuning parameters.
8. Propagate the plug-in configuration. The plug-in configuration file (`plugin-cfg.xml`) is automatically propagated to the Web server if the Web server plug-in configuration service is enabled, and one of the following is true:
  - The Web server is a local Web server. (It are located on the same machine as an application server.)
  - The Web server is a remote IBM HTTP Server (IHS) Version 6.0 that has a running IHS Administrative server.

If neither of these conditions is true, the `plugin-cfg.xml` file must be manually copied to the remote Web server's installation location.

The configuration is complete. To activate the configuration, stop and restart the Web server. If you encounter problems restarting your Web server, check the `http_plugin.log` file for information on what portion of the `plugin-cfg.xml` file contains an error. The log file states the line number on which the error occurred along with other details that might help you diagnose why the Web server did not start. You can then use the administrative console to update the `plugin-cfg.xml` file.

If applications are infrequently installed or uninstalled, which is usually the situation in a production environment, or if you can tolerate the performance impact of generating and distributing the plug-in configuration file each time any of the previously listed actions occur, you should consider enabling this service.

If you are making a series of simultaneous changes, like installing numerous applications, you might want the configuration service disabled until after you make the last change. The Web server plug-in

configuration service is enabled by default. To disable this service, in the administrative console click **Servers > Application Servers > server\_name > Administration Services > Web server plug-in configuration service** and then unselect the Enable automated Web server configuration processing option.

**Tip:** If your installation uses a firewall, make sure you configure the Web server plug-in to use a port that has been opened. (See your security administrator for information on how to obtain an open port.)

---

## Web server plug-in properties settings

Use this page to view or change the settings of a Web server plug-in configuration file. The plug-in configuration file, `plugin_cfg.xml`, provides properties for establishing communication between the Web server and the Application Server.

To view this administrative console page, click **Servers > Web Servers > Web\_server\_name Plug-in Properties**.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information.

The **Runtime** tab is available only when this Web server has accessed applications running on application servers and there is an `http_plugin.log` file.

### Plug-in log file name

Specifies the fully qualified path to the log file to which the plug-in will write error messages. The default file path is `plugin_install_root/logs/web_server_name/http_plugin.log`.

If the file does not exist then it will be created. If the file already exists, it will be opened in append mode and the previous plug-in log messages will remain.

This field corresponds to the `RequestMetrics loggingEnabled` element in the `plugin-cfg.xml` file.

<b>Data type</b>	String
<b>Default for Linux and UNIX platforms</b>	<code>plugin_install_root/logs/web_server_name/http_plugin.log</code>
<b>Default for Windows platforms</b>	<code>plugin_install_root/logs/web_server_name/http_plugin.log</code>

### Plug-in installation location

Specifies the fully qualified path to where the plug-in configuration file is installed.

<b>Data type</b>	String
<b>Default</b>	The default value is the installation root directory.

If you select a Web server plug-in during installation, the installer program configures the Web server to identify the location of the `plugin-cfg.xml` file, if possible. The Web server is considered installed on a local machine if it is on the same machine as the application server. It is considered installed on a remote machine if the Web server and the application server are on different machines.

- If the Web server is installed on a remote machine, the plug-in configuration file, by default, will be installed in the `plugin_install_root/config/web_server_name` directory.
- If the Web server is installed on a local standalone machine, the plug-in configuration file, by default will be installed in the `profile_install_root/config/cells/cell_name/nodes/web_server_name_node/servers/web_server_name` directory.

The installer program adds a directive to the Web server configuration that specifies the location of the `plugin-cfg.xml` file.

For remote Web servers, you must copy the file from the local directory where the Application Server is installed to the remote machine. This is known as propagating the plug-in configuration file. If you are using an IBM HTTP Server (IHS) V6 for your Web server, WebSphere Application Server can automatically propagate the plug-in configuration file for you to remote machines provided there is a working HTTP transport mechanism to propagate the file.

## Plug-in configuration file name

Specifies the file name of the configuration file for the plug-in. The Application Server generates the `plugin-cfg.xml` file by default. The configuration file identifies applications, Application Servers, clusters, and HTTP ports for the Web server. The Web server uses the file to access deployed applications on various Application Servers.

<b>Data type</b>	String
<b>Default</b>	<code>plugin-cfg.xml</code>

If you select a plug-in during installation, the installer program configures the Web server to identify the location and name of the `plugin-cfg.xml` file, if possible.

You can change the name of the plug-in configuration file. However, if you do change the file name, you must also change the Web server configuration to point to the new plug-in configuration file.

## Automatically generate plug-in configuration file

To automatically generate a plug-in configuration file to a remote Web server:

- This field must be checked.
- The plug-in configuration service must be enabled

When the plug-in configuration service is enabled, a plug-in configuration file is automatically generated for a Web server whenever:

- The WebSphere Application Server administrator defines new Web server.
- An application is deployed to an Application Server.
- An application is uninstalled.
- A virtual host definition is updated and saved.

Clear the check box if you want to manually generate a plug-in configuration file for this Web server.

**Important:** When the plug-in configuration file is generated, it does not include `admin_host` on the list of virtual hosts. The *Installing your application serving environment* PDF describes how to add it to the list.

## Automatically propagate plug-in configuration file

To automatically propagate a copy of a changed plug-in configuration file to a Web server:

- This field must be checked.
- The plug-in configuration service must be enabled
- A WebSphere Application Server node agent must be on the node that hosts the Web server associated with the changed plug-in configuration file.

**Note:** The plug-in configuration file can only be automatically propagated to a remote Web server if that Web server is an IHS V6.0 Web server and its administration server is running.



## Ignore DNS failures during Web server startup

Specifies whether the plug-in ignores DNS failures within a configuration when starting.

This field corresponds to the IgnoreDNSFailures element in the plugin-cfg.xml file.

When set to **true**, the plug-in ignores DNS failures within a configuration and starts successfully if at least one server in each ServerCluster is able to resolve the host name. Any server for which the host name can not be resolved is marked **unavailable** for the life of the configuration. No attempts to resolve the host name are made later on during the routing of requests. If a DNS failure occurs, a log message is written to the plug-in log file and the plug-in initialization continues rather than causing the Web server not to start. When **false** is specified, DNS failures cause the Web server not to start.

<b>Data type</b>	String
<b>Default</b>	false

## Refresh configuration interval

Specifies the time interval, in seconds, at which the plug-in should check the configuration file to see if updates or changes have occurred. The plug-in checks the file for any modifications that have occurred since the last time the plug-in configuration was loaded.

In a development environment in which changes are frequent, a lower setting than the default setting of 60 seconds is preferable. In production, a higher value than the default is preferable because updates to the configuration will not occur so often. If the plug-in reload fails for some reason, a message is written to the plug-in log file and the previous configuration is used until the plug-in configuration file successfully reloads. If you are not seeing the changes you made to your plug-in configuration, check the plug-in log file for indications of the problem.

<b>Data type</b>	Integer
<b>Default</b>	60 seconds.

## Plug-in logging

Specifies the location and name of the http\_plugin.log file. Also specifies the scope of messages in the log.

This field corresponds to the RequestMetrics traceLevel element in the plugin-cfg.xml file.

The log describes the location and level of log messages that are written by the plug-in. If a log is not specified within the configuration file, then, in some cases, log messages are written to the Web server error log.

On a distributed platform, if the log file does not exist then it will be created. If the log file already exists, it will be opened in append mode and the previous plug-in log messages will remain.

**Log file name** - The fully qualified path to the log file to which the plug-in will write error messages.

<b>Data type</b>	String
<b>Default</b>	<i>plugin_install_root/logs/web_server_name/http_plugin.log</i>

Specify the file path of the http\_plugin.log file.

**Log level**- The level of detail of the log messages that the plug-in should write to the log. You can specify one of the following values for this attribute:

- Trace. All of the steps in the request process are logged in detail.

- Stats. The server selected for each request and other load balancing information relating to request handling is logged.
- Warn. All warning and error messages resulting from abnormal request processing are logged.
- Error. Only error messages resulting from abnormal request processing are logged.

If a Log level is not specified, the default value **Error** is used.

Be careful when setting the level to **Trace**. A lot of messages are logged at this level which can cause the disk space/file system to fill up very quickly. A **Trace** setting should never be used in a normally functioning environment as it adversely affects performance.

<b>Data type</b>	String
<b>Default</b>	Error

## Web server plug-in request and response optimization properties settings

Use this page to view or change the request and response optimization properties for a Web server plug-in.

To view this administrative console page, click **Servers > Web Servers > *web\_server\_name* Plug-in Properties > Request and Response**.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information.

The **Runtime** tab is available only when this Web server has accessed applications running on application servers and there is an `http_plugin.log` file.

### Maximum chunk size used when reading the response body

Specifies the maximum chunk size the plug-in can use when reading the response body.

This field corresponds to the `ResponseChunkSize` element in the `plugin-cfg.xml` file.

The plug-in reads the response body in 64K chunks until all of the response data is read. This approach causes a performance problem for requests whose response body contains large amounts of data.

If the content length of the response body is unknown, the values specified for this property is used as the size of the buffer that is allocated. The response body is then read in this size chunks, until the entire body is read. If the content length is known, then a buffer size of either the content length or the specified size (whichever is less) is used to read the response body.

<b>Data type</b>	Integer
<b>Default</b>	64 kilobytes

Specify the size in kilobytes (1024 byte blocks).

### Enable Nagle algorithm for connections to the Application Server

When checked, the Nagle algorithm is enabled for connections between the plug-in and the Application Server.

This field corresponds to the `ASDisableNagle` element in the `plugin-cfg.xml` file.

The Nagle algorithm is named after engineer John Nagle, who invented this standard part of the transmission control protocol/internet protocol (TCP/IP). The algorithm reduces network overhead by adding a transmission delay (usually 20 milliseconds) to a small packet, which lets other small packets

arrive and be included in the transmission. Because communications has an associated cost that is not as dependent on packet size as it is on frequency of transmission, this algorithm potentially reduces overhead with a more efficient number of transmissions.

Clear the check box to disable the Nagle algorithm.

### **Enable Nagle Algorithm for the IIS Web Server**

When checked, the Nagle algorithm is used for connections from the Microsoft Internet Informations Services (IIS) Web Server to the Application Server.

This field corresponds to the IHSDisableNagle element in the plugin-cfg.xml file. It only appears if you are using the Microsoft Internet Informations Services (IIS) Web server. Clear the check box to disable the Nagle algorithm for this connection.

### **Chunk response to the client**

When checked, responses to the client are broken into chunks if a Transfer-Encoding : Chunked response header is present in the response.

This field corresponds to the ChunkedResponse element in the plugin-cfg.xml file. It only appears if you are using a Microsoft Internet Informations Services (IIS) Web Server, a Java System Web server, or a Domino Web server. The IHS Web server automatically handles breaking the response into chunks to send to the client.

Clear the check box to if you do not want responses broken into chunks.

### **Accept content for all requests**

This field corresponds to the AcceptAllContent element in the plugin-cfg.xml file.

When selected, users can include content in POST, PUT, GET, and HEAD requests when a Content-Length or Transfer-encoding header is contained in the request header.

### **Virtual host matching**

When selected, virtual host mapping is performed by physically using the port number for which the request was received.

This field corresponds to the VHostMatchingCompat element in the plugin-cfg.xml file.

If this field is not selected, matching is done logically using the port number contained in the host header.

Use the radio buttons to make your physical or logical port selection.

### **Application server port preference**

Specifies which port number the Application Server should use to build URI's for a sendRedirect.

This field corresponds to the AppServerPortPreference element in the plugin-cfg.xml file.

Specify:

- webserverPort if the port number from the host header of the HTTP request coming in is to be used.
- hostHeader if the port number on which the Web server received the request is to be used.

The default is webserverPort.

### **Priority used by the IIS Web server when loading the plug-in configuration file**

Specifies the priority in which the Microsoft Internet Informations Services (IIS) Web server loads the WebSphere Web server plug-in.

This field corresponds to the IISPluginPriority element in the plugin-cfg.xml file. It only appears if you are using the IIS Web server. Because the IIS Web server uses this value during startup, the Web server must be restarted before a change to this field takes effect.

Select one of the following priorities:

- High
- Medium
- Low

The default value of **High** ensures that all requests are handled by the Web server plug-in before they are handled by any other filter/extensions. If problems occur while using a priority of **Medium** or **Low**, you will have to rearrange the order or change the priority of the interfering filter/extension.

## Web server plug-in caching properties settings

Use this page to view or change the caching properties for a Web server plug-in.

To view this administrative console page, click **Servers > Web Servers > Web\_server\_name Plug-in Properties > Caching Properties**.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information.

The **Runtime** tab is available only when this Web server has accessed applications running on application servers and there is an http\_plugin.log file.

### Enable Edge Side Include (ESI) processing to cache the responses

Specifies whether to enable Edge Side Include processing to cache the responses.

This field corresponds to the esiEnable element in the plugin-cfg.xml file.

When selected, Edge Side Include (ESI) processing is used to cache responses. If ESI processing is disabled for the plug-in, the other ESI plug-in properties are ignored. Clear the checkbox to disable Edge Side Include processing.

### Enable invalidation monitor to receive notifications

When checked, the ESI processor receives invalidations from the Application Server.

This field corresponds to the ESIInvalidationMonitor element in the plugin-cfg.xml file. It is ignored if Edge Side Include (ESI) processing is not enabled for the plug-in.

### Maximum cache size

Specifies, in 1K byte units, the maximum size of the cache. The default maximum size of the cache is 1024K bytes (1 megabyte). If the cache is full, the first entry to be evicted from the cache is the entry that is closest its expiration time.

This field corresponds to the esiMaxCacheSize element in the plugin-cfg.xml file.

**Data type**  
**Default**

Integer  
1024 kilobytes

Specify the size in kilobytes (1024 byte blocks).

## Web server plug-in request routing properties settings

Use this page to view or change the request routing properties for a Web server plug-in.

To view this administrative console page, click **Servers > Web Servers > *Web\_server\_name* Plug-in Properties > Plug-in *server\_cluster\_name* Properties**.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information.

The **Runtime** tab is available only when this Web server has accessed applications running on application servers and there is an `http_plugin.log` file.

### Load balancing option

Specifies the load balancing option that the plug-in uses in sending requests to the various application servers associated with that Web server.

This field corresponds to the `LoadBalanceWeight` element in the `plugin-cfg.xml` file.

Select the appropriate load balancing option:

- Round robin
- Random

The Round Robin implementation has a random starting point. The first application server is picked randomly. Round Robin is then used to pick application servers from that point forward. This implementation ensures that in multiple process based Web servers, all of the processes don't start up by sending the first request to the same Application Server.

The default load balancing type is Round Robin.

### Retry interval

Specifies the length of time, in seconds, that should elapse from the time an application server is marked down to the time that the plug-in retries a connection.

This field corresponds to the `ServerWaitforContinue` element in the `plugin-cfg.xml` file.

<b>Data type</b>	Integer
<b>Default</b>	60 seconds

### Maximum size of request content

Select whether there is a limit on the size of request content. If limited, this field also specifies the maximum number of bytes of request content allowed in order for the plug-in to attempt to send the request to an application server.

This field corresponds to the `PostSizeLimit` element in the `plugin-cfg.xml` file.

When a limit is set, the plug-in fails any request that is received that is greater than the specified limit.

Select whether to limit the size of request content:

- No limit
- Set limit

If **Set limit** is selected, specify a limit size.

<b>Data type</b>	Integer
<b>Default</b>	-1, which indicates there is no limit for the post size.

Specify the size in kilobytes (1024 byte blocks).

## Remove special headers

When checked, the plug-in will remove any headers from incoming requests before adding the headers the plug-in is supposed to add before forwarding the request to an application server.

This field corresponds to the `RemoveSpecialHeaders` element in the `plugin-cfg.xml` file.

The plug-in adds special headers to the request before it is forwarded to the application server. These headers store information about the request that will need to be used by the application. Not removing the headers from incoming requests introduces a potential security exposure.

Clear the check box to retain special headers.

## Clone separator change

When checked, the plug-in expects the plus character (+) as the clone separator.

This field corresponds to the `ServerCloneID` element in the `plugin-cfg.xml` file.

Some pervasive devices cannot handle the colon character (:) used to separate clone IDs in conjunction with session affinity. If this field is checked, you must also change the configurations of the associated application servers such that the application servers separates clone IDs with the plus character as well.

Clear the checkbox to use the colon character to separate clone IDs.

---

## Web server plug-in custom properties

If you are using a Web server plug-in, you can add the following custom property to the configuration settings for that plug-in.

To add a custom property:

1. In the administrative console, click **In the administrative console, click `Servers > Web Servers > Web_server_name > Plug-in properties > Custom properties > New`**
2. Under **General Properties** specify the name of the custom property in the Name field and a value for this property in the Value field. You can also specify a description of this property in the Description field.
3. Click **Apply** or **OK**.
4. Click **Save** to save your configuration changes.
5. Restart the server.

Following is a list of Web server plug-in custom properties that are provided with the Application Server. These properties are not shown on the properties settings pages for the plug-in.

### StashfileLocation

Use this element to set a value for the stashfile initialization parameter.

**Data type** String

### KeyringLocation

Use this element to set a value for the keyring initialization parameter.

**Data type** String

---

## Web server plug-in configuration service properties settings

Use this page to view or change the configuration settings for the Web server plug-in configuration service.

To view this administrative console page, click **Application Servers** > *server\_name* > **Administration Services** > **Web server plug-in configuration service**.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information.

### Enable automated Web server configuration processing

When selected, the Web server plug-in configuration service automatically generates the plug-in configuration file whenever the Web server environment changes. For example, the plug-in configuration file is regenerated whenever one of the following activities occurs:

- A new application is deployed on an associated application server.
- The Web server definition is saved.
- An application is removed from an associated application server.
- A new virtual host is defined.

Whenever a virtual host definition is updated, the plug-in configuration file is automatically regenerated for all of the Web servers.

---

## Application Server property settings for a Web server plug-in

Use this page to view or change application server settings for a Web server plug-in.

To view this administrative console page, click **Application Servers** > *server\_name* > **Web Server Plug-in**.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information.

The **Runtime** tab is available only when this Web server has accessed applications running on application servers and there is an `http_plugin.log` file.

### Server role

Specifies the role this application server is assigned.

Select **Primary** to add this application server to the list of primary application servers. The plug-in initially attempts to route requests to the application servers on this list.

Select **Backup** to add this application server to the list of backup application servers. The plug-in does not load balance across the backup application servers. A backup server is only used if a primary server is not available. When the plug-in determines that a backup application server is required, it goes through the list of backup servers, in order, until no servers are left in the list or until a request is successfully sent and a response received from one of the servers on this list.

### Connect timeout

Specifies whether or not there is a limited amount of time the Application Server will maintain a connection with the Web server.

You can select either **No timeout** or **Set timeout**. If you select **Set timeout** you, must specify, in seconds, the length of time a connection with the Web server is to be maintained.

This property enables the plug-in to perform non-blocking connections with the application server. Non-blocking connections are beneficial when the plug-in is unable to contact the destination to determine whether or not the port is available. If no value is specified for this property, the plug-in performs a blocking connect in which the plug-in sits until an operating system times out (which could be as long as 2 minutes depending on the platform) and allows the plug-in to mark the server unavailable.

A value of 0 causes the plug-in to perform a blocking connect. A value greater than 0 specifies the number of seconds you want the plug-in to wait for a successful connection. If a connection does not occur after that time interval, the plug-in marks the server unavailable and fails over to another application server defined for the requested application.

**Data type** Integer

## Maximum number of connections that can be handled by the Application Server

Specifies the maximum number of pending connections to an Application Server that can be flowing through a Web server process at any point in time.

This field corresponds to the ServerMaxConnections element in the plugin-cfg.xml file.

You can select either **No limit** or **Set limit**. If you select **Set limit** you, must specify the maximum number of connections that can exist between the Web server and the Application Server at any given point in time.

If this attribute is set to either zero or -1, there is no limit to the number of pending connections to the Application Servers.

**Data type** Integer  
**Default** -1

## Use extended handshake to check whether Application Server is running

When selected, the Web server plug-in will use an extended handshake to check whether or not the Application Server is running.

This field corresponds to the ServerExtendedHandshake element in the plugin-cfg.xml file.

Select this property if a proxy firewall is between the plug-in and the application server.

The plug-in marks a server as down when the connect() fails. However, when a proxy firewall is in between the plug-in and the application server, the connect() will succeed, even though the back end application server is down. This causes the plug-in to not failover correctly to other application servers.

If the plug-in performs some handshaking with the application server to ensure that it is started before it sends a request it can failover to another application server if it detects that the application server with which it is attempting to perform a handshake is down.

## Send the header "100 Continue" before sending the request content

This field corresponds to the WaitForContinue element in the plugin-cfg.xml file.

When selected, the Web server plug-in will send the header "100 Continue" to the Application Server before it sends the request content.



## Web server plug-in configuration properties

The following table indicates which panel in the administrative console you need to use to manually configure a Web server plug-in property.

Table 6. Web server plug-in configuration properties

Administrative console panel	Field name	Configuration property name
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties</b>	Refresh configuration interval	RefreshInterval
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties</b>	Plug-in log file name	Log->name
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties</b>	Plug-in logging	Log->LogLevel
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties</b>	Ignore DNS failures during Web server startup	IgnoreDNSFailures
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Custom properties &gt; New</b>	KeyringLocation	Keyring
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Custom properties &gt; New</b>	StashfileLocation	Stashfile
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request routing</b>	Load balancing option	LoadBalance
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request Routing</b>	Clone separator change	CloneSeparatorChange
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request Routing</b>	Retry interval	RetryInterval
In the administrative console, click <b>Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request routing</b>	Maximum size of request content	PostSizeLimit
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request routing</b>	Remove special headers	RemoveSpecialHeaders

Table 6. Web server plug-in configuration properties (continued)

In the administrative console, click <b>Application Servers &gt;server_name &gt; Web server plug-in properties</b>	Server role	PrimaryServers and BackupServers list
In the administrative console, click <b>Application Servers &gt;server_name &gt; Web server plug-in properties</b>	Connect timeout	Server ConnectTimeout
In the administrative console, click <b>Application Servers &gt;server_name &gt; Web server plug-in properties</b>	Use extended handshake to check whether Application Server is running	Server Extended Handshake
In the administrative console, click <b>Application Servers &gt;server_name &gt; Web server plug-in properties</b>	Send the header "100 Continue" before sending the request content	WaitForContinue
In the administrative console, click <b>Application Servers &gt;server_name &gt; Web server plug-in properties</b>	Maximum number of connections that can be handled by the Application Server	Server MaxConnections
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request and Response</b>	Application server port preference	AppServerPortPreference
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request and Response</b>	Enable Nagle algorithm for connections to the Application Server	ASDisableNagle
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request and Response</b>	Enable Nagle Algorithm for the IIS Web Server	IISDisableNagle
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request and Response</b>	Virtual host matching	VHostMatchingCompat
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request and Response</b>	Maximum chunk size used when reading the response body	ResponseChunkSize
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request and Response</b>	Accept content for all requests	AcceptAllContent
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request and Response</b>	Chunk response to the client	ChunkedResponse

Table 6. Web server plug-in configuration properties (continued)

In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Request and Response</b>	Priority used by the IIS Web server when loading the plug-in configuration file	IISPluginPriority
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Caching</b>	Enable Edge Side Include (ESI) processing to cache the responses	ESIEnable
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Caching</b>	Maximum cache size	ESIMaxCacheSize
In the administrative console, click <b>Servers &gt; Web Servers &gt; Web_server_name &gt; Plug-in properties &gt; Caching</b>	Enable invalidation monitor to receive notifications	ESIInvalidationMonitor

## Web server plug-in connections

The WebSphere Application Server Web server plug-ins are used to establish and maintain persistent HTTP and HTTPS connections to Application Servers .

When the plug-in is ready to send a request to the application server, it first checks its connection pool for existing connections. If an existing connection is available the plug-in checks its connection status. If the status is still good, the plug-in uses that connection to send the request. If a connection does not exist, the plug-in creates one. If a connection exists but has been closed by the application server, the plug-in closes that connection and opens a new one.

After a connection is established between a plug-in and an application server, it will not be closed unless the application server closes it for one of the following reasons:

- If the **Use Keep-Alive** property is selected and the time limit specified on the **Read timeout** or **Write timeout** property for the HTTP inbound channel has expired.
- The maximum number of persistent requests which can be processed on an HTTP inbound channel has been exceeded. (This number is set using the HTTP inbound channel's **Maximum persistent requests** property.)
- The Application Server is shutting down.

Even if the application server closes a connection, the plug-in will not know that it has been closed until it tries to use it again. The connection will be closed if one of the following events occur:

- The plug-in receives a new HTTP request and tries to reuse the existing connection.
- The number of **httpd** processes drop because the Web server is not receiving any new HTTP requests. (For the IHS Web server, the number of **httpd** processes that are kept alive depends on the value specified on the Web server's **MinSpareServers** directive.)
- The Web server is stopped and all **httpd** processes are terminated, and their corresponding sockets are closed.

**Note:** Sometimes, if a heavy request load is stopped or decreased abruptly on a particular application server, a lot of the plug-in's connections to that application server will be in **CLOSE\_WAIT** state. Because these connections will be closed the first time the plug-in tries to reuse them, having a large number of connections in **CLOSE-WAIT** state should not affect performance

---

## Web server plug-in remote user information processing

You can configure your Web server with a third-party authentication module and then configure the Web server plug-in to route requests to the Application Server. If an application calls the `getRemoteUser()` method, it relies on a private HTTP header that contains the remote user information and is parsed by the plug-in. The plug-in sets the private HTTP header value whenever a Web server authentication module populates the remote user in the Web server data structure. If the private HTTP header value is not set, the application's call to `getRemoteUser()` returns a null value.

- In the case of an Apache and IBM HTTP Server (IHS) Web server, the plug-in builds the private header from the information contained in the associated request record.
- In the case of a Sun One Web server, the plug-in builds the private header from the information contained in the **auth\_user** property associated with the request. The private header is usually set to the name of the local HTTP user of the Web browser, if HTTP access authorization is activated for the URL.
- In the case of a Domino Web server, the plug-in builds the private header from the information contained in the **REMOTE\_USER** environment variable. The plug-in sets this variable to **anonymous** for users who have not logged in and to the *username* for users who are logged into the application.
- In the case of an Internet Information Services (IIS) Web server, the plug-in builds the private header from the information contained in the **REMOTE\_USER** environment variable. The plug-in sets this variable to the name of the user as it is derived from the authorization header sent by the client.

If the private header is not being set in the Sun One, IIS, or Domino Web server plug-in, make sure the request record includes information about the user requesting the data.

**Note:** If an application's call to `getRemoteUser()` returns a null value, or if the correct remote user information is not being added to the Web server plug-in's data structure, make sure the remote user parameter within the WebAgent is still set to **YES**. (Sometimes this parameter gets set to **NO** when service is applied.)

---

## Web server plug-ins

Web server plug-ins enable the Web server to communicate requests for dynamic content, such as servlets, to the application server. A Web server plug-in is associated with each Web server definition. The configuration file (`plugin-cfg.xml`) that is generated for each plug-in is based on the applications that are routed through the associated Web server.

A Web server plug-in is used to forward HTTP requests from a supported Web server to an application server. Using a Web server plug-in to provide communication between a Web server and an application server has the following advantages:

- XML-based configuration file
- Standard protocol recognized by firewall products
- Security using HTTPS, replacing proprietary Open Servlet Engine (OSE) over Secure Sockets Layer (SSL)

Each of the supported distributed platform Web server plug-ins run on a number of operating systems. See Supported Hardware and Software at

<http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html> for the product for the most current information about supported Web servers.

---

## Checking your IBM HTTP Server version

At times, you might need to determine the version of your IBM HTTP Server installation.

1. Change directory to the installation root of the Web server. For example, this is `/opt/IBMIHS` on a Solaris machine.

2. Find the subdirectory that contains `apache.exe` (on a Windows platforms) or `apachectl` (on a UNIX-based platforms, such as z/OS, Solaris, Linux, and HP-UX)
3. On a Windows platform, issue:
 

```
apache.exe -V
```
4. On a UNIX-based platforms issue:
 

```
./apachectl -V 4
```

The version is shown in the "Server version:" field and will look something like the following:

```
Server version: IBM_HTTP_Server/2.0.47 Apache/2.0.47
Server built: July 2 2004 20:38:36
Server's Module Magic Number: 20020903:4.
```

---

## Web server tuning parameters

WebSphere Application Server provides plug-ins for several Web server brands and versions. Each Web server operating system combination has specific tuning parameters that affect the application performance.

- **IBM HTTP Server**

The IBM HTTP Server V6.0 is a multi-process, multi-threaded server.

- **Access logs**

- **Description:** Collects all incoming HTTP requests. Logging degrades performance because IO operation overhead causes logs to grow significantly in a short time.
- **How to view or set:**
  1. Open the IBM HTTP Server `httpd.conf` file, located in the directory `IBM_HTTP_Server_root_directory/conf`.
  2. Search for a line with the text **CustomLog**.
  3. Comment out this line by placing `#` in front of the line.
  4. Save and close the `httpd.conf` file.
  5. Stop and restart the IBM HTTP Server.
- **Default value:** Logging of every incoming HTTP request is enabled.
- **Recommended value:** Disable the access logs.

- **MaxClients**

- **Description:** The `MaxClients` directive controls the maximum number of simultaneous connections or users that the web server can service at any one time. If, at peak usage, your web server needs to support 200 active users at once, you should set `MaxClients` to 220 (200 plus an extra 10% for load growth). Setting `MaxClients` too low could cause some users to believe the web server is not responding. You should have sufficient RAM in your web server machines to support each connected client. For IBM HTTP Server V6.0 on UNIX, you should allocate around 1.5MB `MaxClients` of RAM for use by the IBM HTTP Server. For IBM HTTP Server V6.0 on Windows, you should allocate around 300KB `MaxClients` of RAM for use by the IBM HTTP Server. Some third party modules can significantly increase the amount of RAM used per connected client.
- **How to view or set:** Edit the `MaxClients` directive in the IBM HTTP Server `httpd.conf` file, located in the directory `IBM_HTTP_Server_root_directory/conf`.
- **Default value:** 150
- **Recommended value:** The maximum number of users normally simultaneously connected to your web server, plus an additional 10% for buffer. Note: The `KeepAliveTimeout` setting can affect how long a user is connected to the webserver.

- **MinSpareServers, MaxSpareServers, and StartServers**

- **Description:** Pre-allocates and maintains the specified number of processes so that few processes are created and destroyed as the load approaches the specified number of processes. Specifying similar values reduces the CPU usage for creating and destroying HTTPD processes. Adjust this parameter if the time waiting for IBM HTTP Server to start more servers, so that it can handle HTTP requests, is not acceptable.
- **How to view or set:** Edit the `MinSpareServers`, `MaxSpareServers` and `StartServers` directives in the `httpd.conf` file located in the `IBM_HTTP_Server_root_directory/conf` directory.

- **Default value:** MinSpareServers 5, MaxSpareServers 10, StartServers 5
- **Recommended value:** For optimum performance, specify the same value for the MinSpareServers and the StartServers parameters. If MaxSpareServers is set to less than MinSpareServers, IBM HTTP Server resets MaxSpareServer=MinSpareServer+1. Setting the StartServers too high can cause swapping if memory is not sufficient, degrading performance.
- **ListenBackLog**
  - **Description:** Sets the length of a pending connections queue. When several clients request connections to the IBM HTTP Server, and all threads used, a queue exists to hold additional client requests. However, if you use the default Fast Response Cache Accelerator (FRCA) feature of IBM HTTP Server V6.0 on Windows, the ListenBackLog directive is not used since FRCA has its own internal queue.
  - **How to view or set:** For non-FRCA: Edit the IBM HTTP Server httpd.conf file. Then, add or view the ListenBackLog directive.
  - **Default value:** For HTTP Server V6.0: 1024 with FRCA enabled, 511 with FRCA disabled
  - **Recommended value:** Use the defaults.
- **IBM HTTP Server - Linux**
  - **MaxRequestsPerChild**
    - **Description:** Sets the limit on the number of requests that an individual child server process handles. After the number of requests reaches the value set for the MaxRequestsPerChild parameter, the child process dies. Adjust this parameter if destroying and creating child processes is degrading your Web server performance.
    - **How to view or set:**
      1. Edit the IBM HTTP server httpd.conf file located in the *IBM\_HTTP\_Server\_root\_directory/conf* directory.
      2. Change the value of the parameter.
      3. Save the changes and restart the IBM HTTP server.
    - **Default value:** 500
    - **Recommended value:** Should normally be set to 0. Non-zero settings can be useful if child memory usage is observed to steadily increase over time. Memory leaks have occasionally been observed in third party modules and various OS runtime libraries used by the IBM HTTP Server.
- **IBM HTTP Server - Windows 2000 and Windows 2003**
  - **ThreadsPerChild**
    - **Description:** Sets the number of concurrent threads running at any one time within the IBM HTTP Server.
    - **How to view or set:** Edit the IBM HTTP Server file httpd.conf file located in the directory *IBM\_HTTP\_Server\_root\_directory/conf*. Change the value of the parameter. Save the changes and restart the IBM HTTP Server.

There are two ways to find how many threads are used under load:

1. Use the Windows 2000 and Windows 2003 Performance Monitor under the desktop Start menu:
  - a. Right-click the status bar on your desktop. Click **Task Manager**.
  - b. Select the **Processes** tab.
  - c. Click **View > Select Columns**.
  - d. Select **Thread Count**.
  - e. Click **OK**.

The **Processes** tab shows the number threads for each process under the column name **Threads**, including Apache.

2. Use the IBM HTTP Server server-status (this choice works on all platforms, not just Windows):
  - a. Edit the IBM HTTP Server httpd.conf file as follows: Remove the comment character # from the following lines: #LoadModule status\_module, modules/ApacheModuleStatus.dll, #<Location/server-status>, #SetHandler server-status, and #</Location>.
  - b. Save the changes and restart the IBM HTTP Server.
  - c. In a Web browser, go to the URL: <http://yourhost/server-status>. Alternatively,
  - d. Click **Reload** to update status.



- e. (Optional) If the browser supports refresh, go to `http://your_host/server-status?refresh=5` to refresh every five seconds. You will see five requests currently processing 45 idle servers.
- **Default value:** 250 for IBM HTTP Server V6.0.
- **Recommended value:** Set this value to prevent bottlenecks, allowing just enough traffic through to the application server.
- **Web server configuration reload interval**
  - **Description:** Tracks a variety of configuration information about WebSphere Application Server resources. The Web server needs to understand some of this information, such as Uniform Resource Identifiers (URIs) pointing to WebSphere Application Server resources. This configuration data is pushed to the Web server through the WebSphere Application Server plug-in at intervals specified by this parameter. Periodic updates add new servlet definitions without having to restart any of the WebSphere Application Server servers. However, the dynamic regeneration of this configuration information is costly in terms of performance. Adjust this parameter in a stable production environment.
  - **How to view or set:** Use the Refresh configuration interval Web server plug-in property to change the current setting for this parameter. In the administrative console, click **Servers > Web Servers > Web\_server\_name > Plug-in properties**.
  - **Default value:** The default reload interval is 60 seconds.
  - **Recommended value:** Increase the reload interval to a value that represents an acceptable wait time between the servlet update and the Web server update.

For more information about the `plugin-cfg.xml` file see the topic “Web server plug-ins” on page 68.

- **Sun Java System Web server, Enterprise Edition (formerly Sun ONE) - Solaris operating environment**

The default configuration of the Sun ONE Web server, Enterprise Edition provides a single-process, multi-threaded server.

- **Active threads**

- **Description:** Specifies the current number of threads active in the server. After the server reaches the limit set with this parameter, the server stops servicing new connections until it finishes old connections. If this setting is too low, the server can become throttled, resulting in degraded response times. To tell if the Web server is being throttled, consult its perfdump statistics. Look at the following data:
  - **WaitingThreads count:** If WaitingThreads count is getting close to zero, or is zero, the server is not accepting new connections.
  - **BusyThreads count:** If the WaitingThreads count is close to zero, or is zero, BusyThreads is probably very close to its limit.
  - **ActiveThreads count:** If ActiveThreads count is close to its limit, the server is probably limiting itself.
- **How to view or set:** Use the Maximum number of simultaneous requests parameter in the Enterprise Server Manager interface to control the number of active threads within Sun ONE Web server, Enterprise Edition. This setting corresponds to the `RqThrottle` parameter in the `magnus.conf` file.
- **Default value:** 512
- **Recommended value:** Increase the thread count until the active threads parameters show optimum behavior.

- **Microsoft Internet Information Server (IIS) - Windows NT and Windows 2000**

- **IIS permission properties**

- **Description:** The Web server has several properties that dramatically affect the performance of the application server. The default settings are usually acceptable. However, because other products can change the default settings without user knowledge, make sure to check the IIS settings for the Home Directory permissions of the Web server. The permissions should be set to Script and not to Execute. If the permissions are set to Execute, no error messages are returned, but the performance of WebSphere Application Server is decreased.
- **How to view or set:** To check or change these permissions, perform the following procedure in the Microsoft management console:

1. Select the Web site (usually default Web site).
  2. Right-click and select the **Properties** option.
  3. Click the **Home Directory** tab. To set the permissions of the Home Directory:
    - a. In the **Application** settings, select the **Script** check box in the **Permissions** list and clear the **Execute** check box.
    - b. (Optional) Check the permissions of the sePlugin:
      - 1) Expand the Web server.
      - 2) Right-click the sePlugin and select **Properties**.
      - 3) Confirm that the **Execute** permissions are set to **Execute**.
- **Default value:** Script
  - **Recommended value:** Script
- **Number of expected hits per day**
- **Description:** Controls the memory that IIS allocates for connections.
  - **How to view or set:** Using the performance window, set the parameter to More than 100000 in the Web site properties panel of the Microsoft management console.
  - **Default value:** Fewer than 100000
  - **Recommended value:** More than 100000
- **ListenBackLog parameter**
- **Description:** Alleviates failed connections under heavy load conditions, if you are using IIS on Windows NT and Windows 2000. Failure typically occurs when you are using more than 100 clients. ListenBackLog increases the number of requests that IIS keeps in its queue. Consider raising this value if you see intermittent Unable to locate server errors in the Netscape browser.
  - **How to view or set:**
    1. Use a command prompt to issue the **regedit** command to access the operating system registry.
    2. In the registry window, locate the parameter in the HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\ListenBackLog directory.
    3. Right-click the parameter to adjust the setting according to the server load.
  - **Default value:** 25 (decimal)
  - **Recommended value:** You can set the ListenBackLog parameter can be set as high as 200, without negative impact on performance and with an improvement in load handling.

### Modifying the WebSphere plug-in to improve performance

You can improve the performance of IBM HTTP Server V6.0 (with the WebSphere Web server plug-in) by modifying the plug-in's RetryInterval configuration. The RetryInterval is the length of time to wait before trying to connect to a server that has been marked temporarily unavailable. Making this change can help the IBM HTTP Server V6.0 to scale higher than 400 users.

The plug-in marks a server temporarily unavailable if the connection to the server fails. Although a default value is 60 seconds, it is recommended that you lower this value in order to increase throughput under heavy load conditions. Lowering the RetryInterval is important for IBM HTTP Server V6.0 on UNIX operating systems that have a single thread per process, or for IBM HTTP Server 2.0 if it is configured to have fewer than 10 threads per process.

How can lowering the RetryInterval affect throughput? If the plug-in attempts to connect to a particular application server while the application server threads are busy handling other connections, which happens under heavy load conditions, the connection times out and the plug-in marks the server temporarily unavailable. If the same plug-in process has other connections open to the same server and a response is received on one of these connections, the server is marked again. However, when you use the IBM HTTP Server V6.0 on a UNIX operating system, there is no other connection since there is only one thread and one concurrent request per plug-in process. Therefore, the plug-in waits for the RetryInterval before attempting to connect to the server again.



Since the application server is not really down, but is busy, requests are typically completed in a small amount of time. The application server threads become available to accept more connections. A large RetryInterval causes application servers that are marked temporarily unavailable, resulting in more consistent application server CPU utilization and a higher sustained throughput.

**Note:** Although lowering the RetryInterval can improve performance, if all the application servers are running, a low value can have an adverse affect when one of the application servers is down. In this case, each IBM HTTP Server V6.0 process attempts to connect and fail more frequently, resulting in increased latency and decreased overall throughput.

---

## Gskit install images files

The Global Security Kit (GSKit) installation image files for the WebSphere Web server plug-ins are packaged on the CD with the Web server plug-in files. You can download the appropriate GSKIT file to the workstation on which your Web server is running. Use the following table to assist you in selecting the correct GSKIT installation image file.

Operating system	GSKit 7 Installation image file
Windows	No image name
AIX	gskta.rte
HP-UX	gsk7bas
Solaris Operating Environment	gsk7bas
Linux	gsk7bas_7.0.3.1.i386.rpm
Linux390	gsk7bas-7.0.3.1.s390.rpm
LinuxPPC	gsk7bas-7.0.3.1.ppc.rpm

---

## Plug-ins: Resources for learning

See this topic in the V6 Information Center to find links links to relevant supplemental information about Web server plug-ins. 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.

---

## Web server plug-in tuning tips

### Balancing workloads:

In a distributed environment, you can limit the number of connections that can be handled by an applications server. To do this, go to the **Servers > Web Servers > webserver > Plug-in properties** page in the administrative console and select **Set limit** for the **Minimum number of connections that can be handled by the Application Server** field. Then specify in the **Connections** field the maximum number of connections you want to allow. When this maximum number of connections is reached, the plug-in returns an HTTP 503 response code to the client. This code indicates that the server is currently unable to handle the request because it is experiencing a temporary overloading or because maintenance is being performed.

When this maximum number of connections is reached, the plug-in, when establishing connections, automatically skips that application server, and tries the next available application server. If no application

servers are available, an HTTP 503 response code will be returned to the client. This code indicates that the server is currently unable to handle the request because it is experiencing a temporary overloading or because maintenance is being performed.

Limiting the number of connections that can be established with an application server works best for Web servers that follow the threading model instead of the process model, and only one process is started.

The IBM HTTP Server V6.0.x follows the threading model. To prevent the IBM HTTP Server from starting more than one process, change the following properties in the Web server configuration file (`httpd.conf`) to the indicated values:

```
ServerLimit      1
ThreadLimit     4000
StartServers    1
MaxClients      1024
MinSpareThreads 1
MaxSpareThreads 1024
ThreadsPerChild 1024
MaxRequestsPerChild 0
```

### **Windows Improving performance in a high stress environment:**

If you use the default settings for a Microsoft Windows operating system, you might encounter Web server plug-in performance problems if you are running in a high stress environment. To avoid these problems, consider tuning the the TCP/IP setting for this operating system. Two of the keys setting to tune are `TcpTimedWaitDelay` and `MaxUserPort`.

To tune the `TcpTimedWaitDelay` setting, change the value of the `tcp_time_wait_interval` parameter from the default value of 240 seconds, to 30 seconds:

1. Locate in the Windows Registry:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tcpip\Parameters\TcpTimedWaitDelay
```

If this entry does not exist in your Windows Registry, create it by editing this entry as a new `DWORD` item.

2. Specify, in seconds, a value between 30 and 300 inclusive for this entry. (It is recommended that you specify a value of 30. )

To tune the `MaxUserPort` setting:

1. Locate in the Windows Registry:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tcpip\Parameters\MaxUserPort
```

If this entry does not exist in your Windows Registry, create it by editing this entry as a new `DWORD` item.

2. Set the maximum number of ports to a value between 5000 and 65534 ports, inclusive. (It is recommended that you specify a value of 65534,)

See the Microsoft Web site at <http://www.microsoft.com> for more information about these settings.

---

## **Tuning Web servers**

“Web server tuning parameters” on page 69 lists tuning parameters specific to Web servers. The listed parameters may not apply to all of the supported Web servers. Check your Web server documentation before using any of these parameters.

---

## Chapter 8. Setting up the administrative architecture

If your system uses administrative services, you can specify settings for the service.

Use the settings page for an administrative service to configure administrative services.

---

### Administration service settings

Use this page to view and change the configuration for an administration service.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Administration > Administration Services**

#### Standalone

Specifies whether the server process is a participant in a Network Deployment cell or not. If the box is checked (true), the server does not participate in distributed administration. If the box is unchecked (false), the server participates in the Network Deployment system.

The default value for base WebSphere Application Server installations is true. When addNode runs to incorporate the server into a Network Deployment cell, the value switches to false.

<b>Data type</b>	Boolean
<b>Default</b>	true

#### Preferred Connector

Specifies the preferred JMX Connector type. Available options, such as SOAPConnector or RMIConnector, are defined using the JMX Connectors page.

<b>Data type</b>	String
<b>Default</b>	SOAP

#### Extension MBean Providers collection

Use this page to view and change the configuration for JMX extension MBean providers.

You can configure JMX extension MBean providers to be used to extend the existing WebSphere managed resources in the core administrative system. Each MBean provider is a library containing an implementation of a JMX MBean and its MBean XML Descriptor file.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Administration > Administration Services > Extension MBean Providers**

**Name** The name used to identify the Extension MBean provider library.

**Description**

An arbitrary descriptive text for the Extension MBean Provider configuration.

**Classpath**

The path to the Java archive (JAR) file that contains the Extension MBean provider library. This class path is automatically added to the Application Server class path.

#### Extension MBean Provider settings

Use this page to view and change the configuration for a JMX extension MBean provider.

You can configure a library containing an implementation of a JMX MBean, and its MBean XML Descriptor file, to be used to extend the existing WebSphere managed resources in the core administrative system

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Administration > Administration Services > Extension MBean Providers > *provider\_library\_name***

**Classpath:** The path to the Java archive (JAR) file that contains the Extension MBean provider library. This class path is automatically added to the Application Server class path. The class loader needs this information to load and parse the Extension MBean XML Descriptor file.

**Description:** An arbitrary descriptive text for the Extension MBean Provider configuration. Use this field for any text that helps identify or differentiate the provider configuration.

**Name:** The name used to identify the Extension MBean provider library.

## Extension MBean collection

You can configure Java Management Extension (JMX) MBeans to extend the existing WebSphere Application Server managed resources in the administrative console. Use this page to register JMX MBeans. Any MBeans that are listed have already been registered.

To view this administrative console page, click **Servers > Application Servers > *server name* > Administration > Administration Services > Extension MBean Providers > *provider library name* > Extension MBean**

### DescriptorURI

Specifies the location, relative to the provider class path, where the MBean XML descriptor file is located.

**Type** Specifies the type to use for registering this MBean. The type must match the type that is declared in the MBean descriptor file.

## Extension MBean settings

Use this page to view and configure Java Management Extension (JMX) MBeans.

To view this administrative console page, click **Servers > Application Servers > *server name* > Administration > Administration Services > Extension MBean Providers > *provider library name* > Extension MBean > *Extension MBean name***

**descriptorURI:** Specifies the location, relative to the provider class path, where the MBean XML descriptor file is located.

**type:** Specifies the type to use for registering this MBean. The type must match the type that is declared in the MBean descriptor file.

## Java Management Extensions connector properties

A Java Management Extensions (JMX) connector can either be a Remote Method Invocation (RMI) connector or a Simple Object Access Protocol (SOAP) connector.

Depending on the property, you can specify or set a property in the administrative console, the wsadmin tool, Application Server commands, scripts run from a command line interface, or a custom Java administrative client program that you write. You can also set SOAP connector properties in the `soap.client.props` file.

For specific information on how to code the JMX connector properties for the wsadmin tool, the Application Server commands, or scripts, see the particular tool or command. For specific information on how to code the JMX connector properties for a custom Java administrative client program, see the "Java API documentation for Application Server" topic in the Information Center.

For the administrative console, this article specifies the coding of the particular setting or property. Coding of properties in the `soap.client.props` file that are specific to JMX connectors is specified. These

properties begin with `com.ibm.SOAP`. Other properties in the `soap.client.props` file that contain information that can be set elsewhere in the Application Server are not documented here. The coding for the `com.ibm.ssl.contextProvider` property, which can be set only in the `soap.client.props` file, is specified.

Each profile has a property file at `installation root/profiles/profile name/properties/soap.client.props`. These property files allow you to set different properties, including security and timeout properties. These properties are the default for all administrative connections that use the SOAP JMX connector between processes executing in a particular profile. For instance, the `wsadmin` program executing under a particular profile uses the property values from that file for the SOAP connector behavior unless the properties are overridden by some other programmatic means.

To view the JMX connector custom properties administrative console panel that goes with this article, click one of the following paths:

- **Servers -> Application servers -> *server name* -> Server Infrastructure -> Administration -> Administration Services -> Additional properties -> JMX Connectors->*connector type* -> Additional Properties -> Custom properties**
- **System administration -> Deployment manager ->Additional Properties -> Administration Services -> Additional Properties -> JMX Connectors->*connector type*-> Additional Properties -> Custom properties**
- **System administration -> Node agents ->*node agent name* -> Additional Properties -> Administration Services -> Additional Properties -> JMX Connectors->*connector type*-> Additional Properties -> Custom properties**

## SOAP connector properties

This section discusses JMX connector properties that pertain to SOAP connectors.

### SOAP Request timeout

Specifies the SOAP client request timeout. The value that you choose depends on a number of factors such as the size and the number of the applications that are installed on the server, the speed of your machine, and the level of usage of your machine.

The program default value for the request timeout is 600 seconds. However, other components that connect to the SOAP client can override the default. Components that use the `soap.client.props` file have a default value of 180 seconds.

You can set the property by using one of the following options:

- Scripts run from a command line interface.
- The `soap.client.props` file.

<b>Property</b>	<code>com.ibm.SOAP.requestTimeout</code>
<b>Data type</b>	Integer
<b>Range in seconds</b>	0 to n
<b>Default</b>	If the property is zero (0), the request never times out. 180

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

<b>Property</b>	<code>requestTimeout</code>
<b>Data type</b>	Integer

<b>Range in seconds</b>	0 to n
<b>Default</b>	If the property is zero (0), the request never times out. 600

- A Java administrative client. The property is AdminClient.CONNECTOR\_SOAP\_REQUEST\_TIMEOUT.

### Configuration URL

Specifies the Universal Resource Locator (URL) of the soap.client.props file. Specify the configuration URL property if you want a program to read SOAP properties from this file. You can set the property by using one of the following options:

- Scripts run from a command line interface. Scripts can pass the Configuration URL property to the Application Server on the com.ibm.SOAP.ConfigURL system property.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

<b>Property</b>	ConfigURL
<b>Data type</b>	String
<b>Valid Value</b>	http://Path/soap.client.props
<b>Default</b>	None

- A Java administrative client. The property is AdminClient.CONNECTOR\_SOAP\_CONFIG.

### Security context provider

Specifies the Secure Sockets Layer (SSL) implementation to use between the Application Server and the SOAP client. You can specify either IBM Java Secure Sockets Extension (IBMJSSE) or IBM Java Secure Sockets Extension that has undergone Federal Information Processing Standards certification (IBMJSSEFIPS).

You can set the property by using the soap.client.props file.

<b>Property</b>	com.ibm.ssl.contextProvider
<b>Data type</b>	String
<b>Valid Values</b>	IBMJSSE IBMJSSEFIPS IBMJSSE2
<b>Default</b>	IBMJSSE2

### Secure Sockets Layer (SSL) security

Use this property to enable SSL security between Application Server and the SOAP client. You can set the property by using one of the following options:

- Scripts run from a command line interface.
- The soap.client.props file.

<b>Property</b>	com.ibm.SOAP.securityEnabled
<b>Data type</b>	Boolean
<b>Default</b>	False

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

<b>Property</b>	securityEnabled
-----------------	-----------------

<b>Data type</b>	Boolean
<b>Default</b>	False

- A Java administrative client. The property is AdminClient.CONNECTOR\_SECURITY\_ENABLED.

## SOAP and RMI connector properties

This section discusses JMX connector properties that pertain to both SOAP connectors and RMI connectors.

### Connector type

Specify a connector type of SOAP or RMI, depending on whether Application Server connects to a SOAP server or an RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command line interface.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

<b>Property</b>	Type
<b>Data type</b>	String
<b>Valid values</b>	SOAPConnector RMIConnector
<b>Default</b>	SOAP

- A Java administrative client. The property is AdminClient.CONNECTOR\_TYPE. Specify by using the AdminClient.CONNECTOR\_TYPE\_RMI or the AdminClient.CONNECTOR\_TYPE\_SOAP constants.

### Host

Use the host property to specify the host name or the IP address of the server to which Application Server connects. The server can be a SOAP server or an RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command line interface.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

<b>Property</b>	host
<b>Data type</b>	String
<b>Valid values</b>	Host name or IP address
<b>Default</b>	None

- A Java administrative client. The property is AdminClient.CONNECTOR\_HOST.

### Port

Use the port property to specify the port number of the server to which Application Server connects. The server can be a SOAP server or an RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command line interface.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.



<b>Property</b>	port
<b>Data type</b>	Integer
<b>Valid value</b>	Port number
<b>Default</b>	None

- A Java administrative client. The property is AdminClient.CONNECTOR\_PORT.

### User name

Specifies the user name that Application Server uses to access the SOAP server or the RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command line interface.
- The soap.client.props file.

<b>Property</b>	com.ibm.SOAP.loginUserid
<b>Data type</b>	String
<b>Valid value</b>	The value must match the global SSL settings for SOAP or RMI.
<b>Default</b>	None

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

<b>Property</b>	username
<b>Data type</b>	String
<b>Valid value</b>	The value must match the global SSL settings for SOAP or RMI.
<b>Default</b>	None

- A Java administrative client. The property is AdminClient.USERNAME.

### Password

Specifies the password that Application Server uses to access the SOAP server or the RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command line interface.
- The soap.client.props file.

<b>Property</b>	com.ibm.SOAP.loginPassword
<b>Data type</b>	String
<b>Valid values</b>	The value must match the global SSL settings for SOAP or RMI.
<b>Default</b>	None

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

<b>Property</b>	password
<b>Data type</b>	String
<b>Valid values</b>	The value must match the global SSL settings for SOAP or RMI.
<b>Default</b>	None



- A Java administrative client. The property is AdminClient.PASSWORD.

## Java Management Extensions connectors

Use this page to view and change the configuration for Java Management Extensions (JMX) connectors.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Administration > Administration Services > JMX Connectors**

Java Management Extensions (JMX) connectors communicate with WebSphere Application Server when you invoke a scripting process. There is no default for the type and parameters of a connector. The `wsadmin.properties` file specifies the Simple Object Access Protocol (SOAP) connector and an appropriate port number. You can also use the Remote Method Invocation (RMI) connector.

Use one of the following methods to select the connector type and attributes:

- Specify properties in a properties file.
- Indicate options on the command line.

### Type

Specifies the type of the JMX connector.

**Data type**

Enum

**Default**

SOAPConnector

**Range**

**SOAPConnector**

For JMX connections using Simple Object Access Protocol (SOAP).

**RMIConnector**

For JMX connections using Remote Method Invocation (RMI).

## JMX connector settings

Use this page to view the configuration for a Java Management Extensions (JMX) connector.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Administration Services > JMX Connectors > *connector\_type***

### Type:

Specifies the type of the JMX connector.

**Data type**

Enum

**Default**

SOAPConnector

**Range**

**SOAPConnector**

For JMX connections using Simple Object Access Protocol (SOAP).

**RMIConnector**

For JMX connections using Remote Method Invocation (RMI).

## Repository service settings

Use this page to view and change the configuration for an administrative service repository.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Administration Services > Repository Service.**

## Audit Enabled

Specifies whether to audit repository updates in the log file. The default is to audit repository updates.

<b>Data type</b>	Boolean
<b>Default</b>	true

---

## Administrative agents: Resources for learning

Use the following links to find relevant supplemental information about WebSphere Application Server administrative agents and distributed administration. 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.

View links to additional information:

### Administration

- IBM WebSphere Application Server Redbooks at <http://publib-b.boulder.ibm.com/Redbooks.nsf/portals/WebSphere>.

This site contains a listing of all WebSphere Application Server Redbooks.

- IBM developerWorks WebSphere at <http://www.software.ibm.com/wsdd/>.

This site is the home of technical information for developers working with WebSphere products. You can download WebSphere software, take a fast path to developerWorks 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.

- WebSphere Application Server Support page at <http://www.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 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 9. Configuring the environment

To assist in handling requests among Web applications, Web containers, and application servers, you can configure settings for virtual hosts, variables and shared libraries.

1. Configure virtual hosts.
2. Configure variables.
3. If your deployed applications use shared library files, define the shared library files needed.  
See “Managing shared libraries” on page 93.

---

### Virtual hosts

A virtual host is a configuration that enables 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.

Each virtual host has a logical name and a list of one or more DNS aliases by which it is known. A DNS alias is the TCP/IP hostname and port number that is used to request the servlet, for example `yourHostName:80`. When no port number is specified, 80 is assumed.

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

An application server provides a default virtual host with some common aliases, such as the machine’s IP address, short host name, and fully qualified host name. The alias comprises the first part of the path for accessing a resource such as a servlet. For example, it is `localhost:80` in the request `http://localhost:80/myServlet`.

A virtual host is not associated with a particular node (machine). It is a configuration, rather than a “live object,” explaining why you can create it, but cannot start or stop it. For many users, creating virtual hosts is unnecessary because the `default_host` is provided.

Adding a `localhost` to the virtual hosts adds the host name and IP address of the localhost machine to the alias table. This allows a remote user to access the administrative console.

### Why you would use virtual hosting

Virtual hosts let you manage a single application server on a single machine as if the application server were multiple application servers each on their own host machine. Resources associated with one virtual host cannot share data with resources associated with another virtual host. This is true even though the virtual hosts share the same application server on the same physical machine.

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

Suppose an Internet service provider (ISP) has two customers with Internet sites hosted on the same machine. The ISP keeps the two sites isolated from one another, despite their sharing a machine, by using virtual hosts. The ISP associates the resources of the first company with `VirtualHost1` and the resources of the second company with `VirtualHost2`. Both virtual hosts map to the same application server.

Further suppose that both company sites offer the same servlet. Each site has its own instance of the servlet, and is unaware of the same servlet on the other site. If the company whose site is organized on

VirtualHost2 is past due in paying its account with the ISP, the ISP can refuse all servlet requests that are routed to VirtualHost2. Even though the same servlet is available on VirtualHost1, the requests directed at VirtualHost2 do not go to the other virtual host.

The servlets on one virtual host do not share their context with the servlets on the other virtual host. Requests for the servlet on VirtualHost1 can continue as usual. This is true even though VirtualHost2 is refusing to fill requests for the servlet with the same name.

You associate a servlet or other application with a virtual host instead of the actual DNS address.

## The default virtual host (default\_host)

The product provides a default virtual host (named default\_host).

The virtual host configuration uses wildcard entries with the ports for its virtual host entries.

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

Unless you specifically want to isolate resources from one another on the same node (physical machine), you probably do not need any virtual hosts in addition to the default host.

## How requests map to virtual host aliases

Virtual hosts let you manage a single application server on a single machine as if the application server were multiple application servers that are each on their own host machine. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even though the virtual hosts share the same application server on the same physical machine.

When you request a resource, WebSphere Application Server tries to map the request to an alias of a defined virtual host.

Mappings are both case sensitive and insensitive. For example, the portion "http://host:port/" is case insensitive, but the URL that follows is case sensitive. The match must be alphanumerically exact. Also, different port numbers are treated as different aliases.

For example, the request `http://www.myhost.com/myservlet` maps successfully to `http://WWW.MYHOST.COM/myservlet` but not to `http://WWW.MYHOST.COM/MYSERVLET` or `Www.Myhost.Com/Myservlet`. In the latter two cases, these mappings fail due to case sensitivity. The request `http://www.myhost.com/myservlet` does not map successfully to `http://myhost/myservlet` or to `http://myhost:9876/myservlet`. These mappings fail because they are not alphanumerically correct.

You can use wildcard entries for aliases by port and specify that all valid host name and address combinations on a particular port map to a particular virtual host.

If you request a resource using an alias that cannot be mapped to an alias of a defined virtual host, you receive a 404 error in the browser that was used to issue the request. A message states that the virtual host could not be found.

Two sets of associations occur for virtual hosts. Application deployment associates an application with a virtual host. Virtual host definitions associate the network address of the machine and the HTTP transport or Web server port assignment of the application server with the virtual host. Looking at the flow from the Web client request for the snoop servlet, for example, the following actions occur:

1. The Web client asks for the snoop servlet: at Web address `http://www.some_host.some_company.com:9080/snoop`

2. The some\_host machine has the 9080 port assigned to the stand-alone WebSphere Application Server, *server1*.
3. The server1 Application Server looks at the virtual host assignments to determine the virtual host that is assigned to the alias some\_host.some\_company.com:9080.
4. The application server finds that no explicit alias for that DNS string exists. However, a wild card assignment for host name \* at port 9080 does exist. This is a match. The virtual host that defines the match is default\_host.
5. The application server looks at the applications deployed on the default\_host and finds the snoop servlet.
6. The application server serves the application to the Web client and the requester is able to use the snoop servlet.

You can have any number of aliases for a virtual host. You can even have overlapping aliases, such as:

Virtual host	Alias	Port
default_host	*	9080
	localhost	9080
	my_machine	9080
	my_machine.my_company.com	9080
	localhost	80

The Application Server looks for a match using the explicit address specified on the Web client address. However, it might resolve the match to any other alias that matches the pattern before matching the explicit address. Simply defining an alias first in the list of aliases does not guarantee the search order when WebSphere Application Server is looking for a matching alias.

A problem can occur if you use the same alias for two different virtual hosts. For example, assume that you installed the default application and the snoop servlet on the default\_host. You also have another virtual host called the admin\_host. However, you have not installed the default application or the snoop servlet on the admin\_host.

Assume that you define overlapping aliases for both virtual hosts because you accidentally defined port 9080 for the admin\_host instead of port 9060:

Virtual host	Alias	Port
default_host	*	9080
	localhost	9080
admin_host	*	9060
	my_machine.com	9080

Assume that a Web client request comes in for `http://my_machine.com:9080/snoop`.

If the application server matches the request against \*:9080, the application is served from the default\_host. If the application server matches the request to my.machine.com:9080, the application cannot be found. A 404 error occurs in the browser that issues the request. A message states that the virtual host could not be found.

This problem is the result of not finding the requested application in the first virtual host that has a matching alias. The correct way to code aliases is for the alias name on an incoming request to match only one virtual host in all of your virtual host definitions. If the URL can match more than one virtual host, you can see the problem just described.

---

## Configuring virtual hosts

Virtual hosts enable you to isolate and independently manage multiple sets of resources on the same physical machine.

1. Create a virtual host using the “Virtual host collection” of the administrative console. Click **Environment > Virtual Hosts** from the navigation tree of the console. Click **New**. On the “Virtual host settings” on page 87 page that displays, specify an administrative name for the virtual host. When you create a virtual host, a default set of 90 MIME entries is created for the virtual host.
2. There must be a virtual host alias corresponding to each port used by an HTTP transport. There is one HTTP transport in each Web container, usually assigned to the virtual host named `default_host`. You can change the default assignment to any valid virtual host.

You must create a virtual host for each HTTP port in the following cases:

- You are using the internal HTTP transport with a port other than the default of 9080. You must define the port that you are using.
- You are using the default port 9080, but the port is no longer defined. You must define port 9080.
- You have created multiple Application Servers (either stand-alone servers or cluster members) that use the same virtual host. Because each server must be listening on a different HTTP transport port, you must define a virtual host alias for the transport port of each server.

If you define new virtual host aliases, identify the port values that the aliases use on the “HTTP transport collection” on page 128 page.

3. If necessary, create a virtual host alias for each HTTP transport port.  
From the “Virtual host collection” page, click your virtual host. On the “Virtual host settings” on page 87 page for the virtual host, click **Host aliases**. To define a virtual host alias on the “Host alias collection” on page 87 page, click **New**. On the “Host alias settings” on page 88 page for the virtual host alias, specify a host name and a port. Configure the virtual host to contain an alias for the port number. For example, specify an alias of `*:9082` if 9082 is the port number in use by the transport.
4. When you enter the URL for the application into a Web browser, include the port number in the URL.  
For example, if 9082 is the port number, specify a URL such as  
`http://localhost:9082/wlm/SimpleServlet`
5. If MIME entries are not specified at the Web module level, define MIME object types and their file name extensions. For each needed MIME entry on the “MIME type collection” on page 89 page, click **New**. On the “MIME type settings” on page 89 page, specify a MIME type and extension.
6. After you configure a virtual host alias or change a configuration, you must regenerate the Web server plug-in configuration and restart WebSphere Application Server.

## Virtual host collection

Use this page to create and manage configurations that each let a single host machine resemble multiple host machines. Such configurations are known as *virtual hosts*.

To view this administrative console page, click **Environment > Virtual Hosts**.

Each virtual host has a logical name (which you define on this panel) and is known by its list of one or more domain name system (DNS) aliases. A DNS alias is the TCP/IP host name and port number used to request the servlet, for example `yourHostName:80`. (Port 80 is the default.)

You define one or more alias associations by clicking an existing virtual host or by adding a new virtual host.

When a servlet request is made, the server name and port number entered into the browser are compared to a list of all known aliases in an effort to locate the correct virtual host to serve the servlet. No match returns an error to the browser.

An application server profile provides a default virtual host with some common aliases, such as the internet protocol (IP) address, the DNS short host name, and the DNS fully qualified host name. The alias comprises the first part of the path for accessing a resource such as a servlet.

For example, the alias is localhost:80 in the request `http://localhost:80/myServlet`.

A virtual host is not associated with a particular profile or node (machine), but is associated with a particular server instead. It is a configuration, rather than a "live object." You can create a virtual host, but you cannot start or stop it.

For many users, creating virtual hosts is unnecessary because the default\_host that is provided is sufficient.

Adding the host name and IP address of the localhost machine to the alias table lets a remote user access the administrative console.

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.

## Name

Specifies a logical name for configuring Web applications to a particular host name. The default virtual host is suitable for most simple configurations.

Virtual hosts enable you to isolate, and independently manage, multiple sets of resources on the same physical machine. Determine whether you need a virtual host alias for each port associated with an HTTP transport channel or an HTTP transport. There must be a virtual host alias corresponding to each port used by an HTTP transport channel or an HTTP transport. There is one HTTP transport channel or HTTP transport associated with each Web container, and there is one Web container in each application server.

When you create a virtual host, a default set of 90 MIME entries is created for the virtual host.

You must create a virtual host for each HTTP port in the following cases:

- You use the internal HTTP transport with a port other than the default value of 9080, or for some reason the virtual host does not contain the usual entry for port 9080.
- You create multiple application servers (stand-alone servers, managed servers, or cluster members) that are using the same virtual host. Because each server must be listening on a different HTTP port, you need a virtual host alias for the HTTP port of each server.

## Virtual host settings

Use this page to configure a virtual host instance.

To view this administrative console page, click **Environment > Virtual Hosts > virtual\_host\_name**.

### Name:

Specifies a logical name for configuring Web applications to a particular host name. The default virtual host is suitable for most simple configurations.

<b>Data type</b>	String
<b>Default</b>	default_host

## Host alias collection

Use this page to manage host name aliases defined for a virtual host. An alias is the DNS host name and port number that a client uses to form the URL request for a Web application resource.



To view this administrative console page, click **Environment > Virtual Hosts > virtual\_host\_name > Host Aliases**.

**Host Name:**

Specifies the IP address, DNS host name with domain name suffix, or just the DNS host name, used by a client to request a Web application resource (such as a servlet, JavaServer Pages (JSP) file, or HTML page). For example, the host alias name is myhost in a DNS name of myhost:8080.

The product provides a default virtual host (named default\_host). The virtual host configuration uses the wildcard character \* (asterisk) along with the port number for its virtual host entries. Unless you specifically want to isolate resources from one another on the same node (physical machine), you probably do not need any virtual hosts in addition to the default host.

**Port:**

Specifies the port for which the Web server has been configured to accept client requests. For example, the port assignment is 8080 in a DNS name of myhost:8080. A URL refers to this DNS as: http://myhost:8080/servlet/snoop.

**Host alias settings:**

Use this page to view and configure a host alias.

To view this administrative console page, click **Environment > Virtual Hosts > virtual\_host\_name > Host Aliases > host\_alias\_name**.

*Host name:*

Specifies the IP address, domain name system (DNS) host name with domain name suffix, or the DNS host name that clients use to request a Web application resource, such as a servlet, JSP file, or HTML page.

For example, when the DNS name is myhost, the host alias is myhost:8080, where 8080 is the port. A URL request can refer to the snoop servlet on the host alias as: http://myhost:8080/servlet/snoop.

When there is no port number specified for a host alias, the default port is 80. For existing virtual hosts, the default host name and port reflect the values specified at product installation or configuration. For new virtual hosts, the default can be \* to allow any value or no specification.

**Data type**

String

**Default**

\*

You can also use the IP address or the long or short DNS name.

*Port:*

Specifies the port where the Web server accepts client requests. Specify a port value in conjunction with the host name.

The default reflects the value specified at product setup. The default might be 80, 81, 9060 or a similar value.

**Data type**

Integer

**Default**

9060



## MIME type collection

Use this page to view and configure multi-purpose internet mail extensions (MIME) object types and their file name extensions.

The list shows a collection of MIME type extension mappings defined for the virtual host. Virtual host MIME entries apply when you do not specify MIME entries at the Web module level.

To view a list of current virtual host Mime types in the administrative console, click **Environment > Virtual Hosts > *virtual\_host\_name* > MIME Types**.

### ***MIME Type:***

Specifies a MIME type, which can be application, audio, image, text, video, www, or x-world. An example value for MIME type is `text/html`.

### ***Extensions:***

Specifies file extensions of files that map the MIME type. Do not specify the period before the extension. Example extensions for a `text/html` MIME type are `htm` and `html`.

### ***MIME type settings:***

Use this page to configure a multi-purpose internet mail extensions (MIME) object type.

To view this administrative console page, click **Environment > Virtual Hosts > *virtual\_host\_name* > MIME Types > *MIME\_type***.

### ***MIME Type:***

Specifies a MIME type, which can be application, audio, image, text, video, www, or x-world. An example value for MIME type is `text/html`.

An example value for MIME type is `text/html`. A default value appears only if you are viewing the configuration for an existing instance.

**Data type** String

### ***Extensions:***

Specifies file extensions of files that map the MIME type. Do not specify the period before the extension. Example extensions for a `text/html` MIME type are `htm` and `html`.

File extensions for a `text/html` MIME type are `.htm` and `.html`. A default value appears only if you are viewing the configuration for an existing MIME type.

**Data type** String

---

## Variables

A variable is a configuration property that can be used to provide a parameter for some values in the system. A variable has a name and a value.

WebSphere variables are used for:

- Configuring WebSphere Application Server path names, such as `JAVA_HOME`, and `APP_INSTALL_ROOT`.

- Configuring certain customization values.

Each variable has a scope. A scope is the range of locations in the WebSphere Application Server network where the variable is applicable.

- A variable with a node-level scope is available only on the node and the servers on that node. If a node-level variable has the same name as a cell-wide variable, the node-level variable value takes precedence.
- A server variable is available only on the one server process. A server variable takes precedence over a variable with the same name that is defined at a higher level.

You can use variables in configuration values such as file system path settings. Use the following syntax to refer to a variable:

```
${variable_name}
```

The value of a variable can contain a reference to another variable. The value of the variable is computed by substituting the value of the referenced variable recursively.

Variables are useful when concatenating two path variables when the specification does not accept the *AND* operator. For example, suppose that the following variables exist:

Variable name	Variable value
<b>ROOT_DIR</b>	/
<b>HOME_DIR</b>	\${ROOT_DIR}home
<b>USER_DIR</b>	\${HOME_DIR}/myuserdir

The variable reference `${USER_DIR}` resolves to the value `/home/myuserdir`.

---

## Configuring WebSphere variables

This topic describes how to create a WebSphere Application Server variable.

You can define a WebSphere Application Server variable to provide a parameter for some values in the system. After you define the name and value for a variable, the value is used in place of the variable name. Variables most often specify file paths. However, some system components also support the use of variables.

WebSphere variables are used for:

- Configuring WebSphere Application Server path names, such as `JAVA_HOME`, and `APP_INSTALL_ROOT`.
- Configuring certain customization values.

The scope of a variable can be cell-wide, node-wide, or applicable to only one server process.

Define variables on the **Environment > WebSphere Variables** console page.

Define the scope to apply a variable node-wide or to only one server process. A variable resolves to its new value when used in a component that supports the use of variables.

1. Click **Environment > WebSphere Variables** in the administrative console to define a new variable.
2. Specify the scope of the variable. Declare the new variable for the **Node** or **Server** and click **Apply**.  
The variable exists at the level you specify. Define a variable at multiple levels to use multiple values. The more granular definition overrides the higher level setting.  
For instance, if you specify the same variable on a node and a server, the server setting overrides the node setting.

Scoping variables is particularly important when testing data source objects. Variable scoping can cause a data source to fail the test connection, but to succeed at run time, or to pass the test connection, but fail at run time.

See the *Developing and deploying applications* PDF for more information.

3. Click **New** on the WebSphere Variables page.
4. Specify a name, a value, and a description on the Variable page. Click **OK**.
5. Verify that the variable is displayed in the list of variables. The administrative console does not pick up typing errors. The variable is ignored if it is referred to incorrectly.
6. Save your configuration.
7. Stop the server and start the server again to put the variable configuration into effect.

## WebSphere variables collection

Use this page to view and change a list of substitution variables with their values and scope.

To view this administrative console page, click **Environment > WebSphere Variables**.

For information on a variable, click the variable and read the value in the **Description** field.

### Name

Specifies the symbolic name for a WebSphere Application Server variable. For example, a variable name might represent a physical path or URL root used by WebSphere Application Server.

### Value

Specifies the value that the symbolic name represents. For example, the value might be an absolute path value for a file or URL root.

### Scope

Specifies the level at which a WebSphere Application Server variable is visible on the administrative console panel.

A resource can be visible in the administrative console collection table at the node or server scope.

## Variable settings

Use this page to define the name and value of a WebSphere Application Server substitution variable.

To view this administrative console page, click **Environment > WebSphere Variables > WebSphere\_variable\_name**.

### Name:

Specifies the symbolic name for a WebSphere Application Server variable. For example, a variable name might represent a physical path or URL root that is used by WebSphere Application Server.

WebSphere variables are used for:

- Configuring WebSphere Application Server path names, such as JAVA\_HOME, and APP\_INSTALL\_ROOT.
- Configuring certain customization values.

WebSphere Application Server substitutes the symbolic name wherever its value displays in the system.

For example, "JAVA\_HOME" is the symbolic name representing the file system path to the installation directory for the Java Virtual Machine (JVM). For example, the value is /opt/IBM/WebSphere/AppServer/java for the WebSphere Application Server product on a Linux machine.

You can create new variables for use in WebSphere Application Server components that support the use of variables.

**Data type** String

**Value:**

Specifies the value that the symbolic name represents. For example, the value might be an absolute path value for a file or URL root.

For example, `/opt/IBM/WebSphere/AppServer/java` is the value on a Linux machine for a variable named `JAVA_HOME`.

**Data type** String

**Description:**

Documents the purpose of a variable.

**Data type** String

## IBM Toolbox for Java JDBC driver

WebSphere Application Server supports the **IBM Toolbox for Java** JDBC driver. The IBM Toolbox for Java JDBC driver is included with the IBM Toolbox for Java product.

IBM Toolbox for Java is a library of Java classes that are optimized for accessing iSeries and AS/400 data and resources. You can use the IBM Toolbox for Java JDBC driver to access local or remote **DB2 UDB for iSeries 400** databases from server-side and client Java applications that run on any platform that supports Java.

IBM Toolbox for Java is available in these versions:

### IBM Toolbox for Java licensed program

The licensed program is available with every OS/400 release, starting with Version 4 Release 2 (V4R2). You can install the licensed program on your iSeries 400 system, and then either copy the IBM Toolbox for Java JAR file (*jt400.jar*) to your system or update your system *classpath* to locate the server installation. Product documentation for IBM Toolbox for Java is available from the iSeries information center: <http://publib.boulder.ibm.com/iseries/v5r1/ic2924/index.htm>. Locate the documentation by traversing the following path in the left-hand navigation window of the iSeries information center: **Programming > Java > IBM Toolbox for Java**.

### JTOpen

JTOpen is the open source version of IBM Toolbox for Java, and is more frequently updated than the licensed program version. You can download JTOpen from <http://www-1.ibm.com/servers/eserver/iseries/toolbox/downloads.htm>. You can also download the *JTOpen Programming Guide*. The guide includes instructions for installing JTOpen and information about the JDBC driver.

The JDBC driver for both versions supports JDBC 2.0. For more information about IBM Toolbox for Java and JTOpen, see the product Web site at <http://www-1.ibm.com/servers/eserver/iseries/toolbox/index.html>.

**Note:** If you are using WebSphere Application Server on platforms other than iSeries, use the **JTOpen** version of the Toolbox JDBC driver.

## Configure and use the jt400.jar file

1. Download the *jt400.jar* file from the **JTOpen** URL at <http://www-1.ibm.com/servers/eserver/iseries/toolbox/downloads.htm>. Place it in a directory on your workstation such as *C:\JDBC\_Drivers\Toolbox*.
2. Open the administrative console.
3. Select **Environment**.
4. Select **Managed WebSphere Variables**.
5. Set the managed variable *OS400\_TOOLBOX\_JDBC\_DRIVER\_PATH* at the **Node** level.
6. Double click **OS400\_TOOLBOX\_JDBC\_DRIVER\_PATH**.
7. Set the value to the full directory path to the *jt400.jar* file downloaded in step one. Do not include *jt400.jar* in this value. For example,

```
OS400_TOOLBOX_JDBC_DRIVER_PATH == "C:\JDBC_Drivers\Toolbox"
```

When you choose a Toolbox driver from the list of possible resource providers the **Classpath** field looks like:

```
Classpath == ${OS400_TOOLBOX_JDBC_DRIVER_PATH}/jt400.jar
```

---

## Shared library files

Shared library files in WebSphere Application Server consist of a symbolic name, a Java class path, and a native path for loading Java Native Interface (JNI) libraries.

You can define a shared library at the cell, node, or server level. Defining a library at one of the three levels does not cause the library to be placed into the application server's class loader. You must associate the library to an application or server in order for the classes represented by the shared library to be loaded in either a server-wide or application-specific class loader.

A separate class loader is used for shared libraries that are associated with an application server. This class loader is the parent of the application class loader, and the WebSphere Application Server extensions class loader is its parent. Shared libraries that are associated with an application are loaded by the application class loader.

---

## Managing shared libraries

Shared libraries are files used by multiple applications. Using the administrative console, you can define a shared library at the cell, node, or server level. You can then associate the library to an application or server to load the classes represented by the shared library in either a server-wide or application-specific class loader. Using an installed optional package, you can associate a shared library to an application by declaring the dependent library *.jar* file in the *MANIFEST.MF* file of the application. Refer to the Java 2 Platform, Enterprise Edition (J2EE) 1.4 specification, section 8.2 for an example.

If your deployed applications use shared library files, define shared libraries for the library files and associate the libraries with specific applications or with an application server. Associating a shared library file with a server associates the file with all applications on the server. Use the Shared Libraries page to define new shared library files to the system and remove them.

- Use the administrative console to define a shared library.
  1. Create a shared library for each library file that your applications need.
  2. Associate each shared library with an application or a server.
    - Associate a shared library with an application that uses the shared library file.
    - Associate a shared library with an application server so every application on the server can use the shared library file.
- Use an installed optional package to declare a shared library for an application.
- Remove a shared library.

1. Click **Environment > Shared Libraries** in the console navigation tree to access the Shared Libraries page.
2. Select the library to be removed.
3. Click **Delete**.

The list of shared libraries is refreshed. The library file no longer displays in the list.

## Creating shared libraries

Shared libraries are files used by multiple applications.

The first step for making a library file available to multiple applications deployed on a server is to create a shared library for each library file that your applications need. When you create the shared libraries, set variables for the library files.

Use the Shared Libraries page to create and configure shared libraries.

1. Go to the Shared Libraries page. Click **Environment > Shared Libraries** in the console navigation tree.
2. Change the scope of the collection table to see what shared libraries are in a cell, node, or server.
  - a. Select the cell, a node, or a server.
  - b. Click **Apply**.
3. Click **New**.
4. Configure the shared library.
  - a. On the settings page for a shared library, specify the name, class path, and any other variables for the library file that are needed.
  - b. Click **Apply**.
5. Repeat steps 1 through 4 until you define a shared library instance for each library file that your applications need.

Using the administrative console, associate your shared libraries with specific applications or with the class loader of an application server. Associating a shared library file with a server class loader associates the file with all applications on the server.

Alternatively, you can use an installed optional package to associate your shared libraries with an application.

## Shared library collection

Use this page to define a list of shared library files that deployed applications can use.

To view this administrative console page, click **Environment > Shared Libraries**.

By default, a shared library is accessible to applications deployed (or installed) on the same node as the shared library file. Use the **Scope** field to change the scope to a different node or to a specific server.

### Name

Specifies a name for the shared library.

### Description

Describes the shared library file.

### Shared library settings

Use this page to make a library file available to deployed applications.

To view this administrative console page, click **Environment > Shared Libraries > *shared\_library\_name***.

**Name:**

Specifies a name for the shared library.

**Data type** String

**Description:**

Describes the shared library file.

**Data type** String

**Classpath:**

Specifies the class path used to locate the JAR files for the shared library support.

**Data type** String  
**Units** Class path

**Native Library Path:**

Specifies the class path for locating platform-specific library files for shared library support; for example, .dll, .so, or \*SRVPGM objects.

**Data type** String  
**Units** Class path

## Associating shared libraries with applications

You can associate a shared library with an application. Classes represented by the shared library are then loaded in the application's class loader, making the classes available to the application.

This article assumes that you have defined a shared library at the cell, node, or server level. The shared library represents a library file used by multiple deployed applications.

This article also assumes that you want to use the administrative console, and not an installed optional package, to associate a shared library with an application.

To associate a shared library with an application, create and configure a library reference using the administrative console. A library reference specifies the name of the shared library file.

If you associate a shared library with an application, do not associate the same shared library with a server class loader.

1. Click **Applications > Enterprise Applications > *application\_name* > Libraries** in the console navigation tree to access the Library Ref page.
2. Click **Add**.
3. On the settings page for a library reference, specify variables for the library reference as needed. The variables identify the shared library file that your application uses.
4. Click **Apply**.

The name of the library reference is shown in the list on the Library Ref page.



Repeat steps 2 through 4 until you define a library reference instance for each shared library that your application requires.

## Associating shared libraries with servers

You can associate shared libraries with the class loader of a server. Classes represented by the shared library are then loaded in a server-wide class loader, making the classes available to all applications deployed on the server.

This article assumes that you have defined a shared library at the cell, node, or server level. The shared library represents a library file used by multiple deployed applications.

To associate a shared library with the class loader of a server, create and configure a library reference using the administrative console. A library reference specifies the name of the shared library file.

If you associate a shared library with a server class loader, do not associate the same shared library with an application.

1. Configure class loaders for applications deployed on the server.
  - a. Click **Servers > Application Servers > *server\_name*** to access the settings page for the application server.
  - b. Set values for the application **Class loader policy** and **Class loading mode** of the server. For information on these settings, see “Application server settings” on page 112 and class loaders.
2. Create a library reference for each shared library file that your application needs.
  - a. Go to the settings page for a class loader.
  - b. Click **Libraries** to access the Library Ref page.
  - c. Click **Add**.
  - d. On the settings page for a library reference, specify variables for the library reference as needed. The variables identify the shared library file that your application uses.
  - e. Click **Apply**. The name of the library reference is shown in the list on the Library Ref page.

Repeat the previous steps until you define a library reference for each shared library that your application needs.

## Installed optional packages

*Installed optional packages* enable applications to use the classes in Java archive (.jar) files without having to include them explicitly in a class path. An installed optional package is a .jar file containing specialized tags in its manifest file that enable the application server to identify it. An installed optional package declares one or more shared library .jar files in the manifest file of an application. When the application is installed on a server, the classes represented by the shared libraries are loaded in the class loader of the application, making the classes available to the application.

When a Java 2 Platform, Enterprise Edition (J2EE) application is installed on a server, dependency information is specified in its manifest file. WebSphere Application Server reads the dependency information of the application (.ear file) to automatically associate the application with an installed optional package .jar file. WebSphere Application Server adds the .jar files in associated optional packages to the application class path. Classes in the installed optional packages are then available to application classes.

Installed optional packages used by WebSphere Application Server are described in section 8.2 of the J2EE specification, Version 1.4 at [http://java.sun.com/j2ee/j2ee-1\\_4-fr-spec.pdf](http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf).

WebSphere Application Server supports using the manifest file (manifest.mf) in shared library .jar files and application .ear files. WebSphere Application Server does not support the Java 2 Platform Standard Edition (J2SE) Installed Optional Package semantics used in the J2SE specification



(<http://java.sun.com/j2se/1.3/docs/guide/extensions/spec.html>), which primarily serve the applet environment. WebSphere Application Server ignores applet-specific tags within manifest files.

### Sample manifest.mf file

A sample manifest file follows for an application `app1.ear` that refers to a single shared library file `util.jar`:

```
app1.ear:
  META-INF/application.xml
  ejb1.jar:
    META-INF/MANIFEST.MF:
      Extension-List: util
      util-Extension-Name: com/example/util
      util-Specification-Version: 1.4
    META-INF/ejb-jar.xml

util.jar:
  META-INF/MANIFEST.MF:
    Extension-Name: com/example/util
    Specification-Title: example.com's util package
    Specification-Version: 1.4
    Specification-Vendor: example.com
    Implementation-Version: build96
```

The syntax of a manifest entry depends on whether the entry applies to a member with a defining role (the shared library) or a member with a referencing role (a J2EE application or a module within a J2EE application).

### Manifest entry tagging

Main tags used for manifest entries include the following:

#### Extension-List

A required tag with variable syntax. Within the context of the referencing role (application's manifest), this is a space delimited list that identifies and constructs unique `Extension-Name`, `Extension-Specification` tags for each element in the list. Within the context of the defining role (shared library), this tag is not valid.

#### Extension-Name

A required tag that provides a name and links the defining and referencing members. The syntax of the element within the referencing role is to prefix the element with the `<ListElement>` string. For each element in the `Extension-List`, there is a corresponding `<ListElement>-Extension-Name` tag. The defining string literal value for this tag (in the above sample `com/example/util`) is used to match (in an equality test) the corresponding tags between the defining and referencing roles.

#### Specification-Version

A required tag that identifies the specification version and links the defining and referencing members.

#### Implementation-Version

An optional tag that identifies the implementation version and links the defining and referencing members.

Further information on these tags is in the `.jar` file specification at <http://java.sun.com/j2se/1.4.2/docs/guide/jar/jar.html#Manifest%20Specification>.

## Using installed optional packages

You can associate one or more shared libraries with an application using an installed optional package that declares the shared libraries in the application's manifest file. Classes represented by the shared libraries are then loaded in the application's class loader, making the classes available to the application.

Read about installed optional packages in “Installed optional packages” on page 96 and in section 8.2 of the Java 2 Platform, Enterprise Edition (J2EE) specification, Version 1.4 at [http://java.sun.com/j2ee/j2ee-1\\_4-fr-spec.pdf](http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf).

WebSphere Application Server does not support the Java 2 Platform Standard Edition (J2SE) Installed Optional Package semantics used in the J2SE specification (<http://java.sun.com/j2se/1.3/docs/guide/extensions/spec.html>), which primarily serve the applet environment. WebSphere Application Server ignores applet-specific tags within manifest files.

Installed optional packages expand the existing shared library capabilities of an application server. Prior to Version 6, an administrator was required to associate a shared library to an application or server. Installed optional packages enable an administrator to declare a dependency in an application’s manifest file to a shared library, with installed optional package elements listed in the manifest file, and automatically associate the application to the shared library. During application installation, the shared library .jar file is added to the class path of the application class loader.

If you use an installed optional package to associate a shared library with an application, do not associate the same shared library with an application class loader or a server class loader using the administrative console.

1. Assemble the library file, including the manifest information that identifies it as an extension. Two sample manifest files follow. The first sample manifest file has application `app1.ear` refer to a single shared library file `util.jar`:

**app1.ear:**

```
META-INF/application.xml
ejb1.jar:
  META-INF/MANIFEST.MF:
    Extension-List: util
    util-Extension-Name: com/example/util
    util-Specification-Version: 1.4
  META-INF/ejb-jar.xml
```

**util.jar:**

```
META-INF/MANIFEST.MF:
  Extension-Name: com/example/util
  Specification-Title: example.com's util package
  Specification-Version: 1.4
  Specification-Vendor: example.com
  Implementation-Version: build96
```

The second sample manifest file has application `app1.ear` refer to multiple shared library .jar files:

**app1.ear:**

```
META-INF/application.xml
ejb1.jar:
  META-INF/MANIFEST.MF:
    Extension-List: util1 util2 util3
    Util1-Extension-Name: com/example/util1
    Util1-Specification-Version: 1.4
    Util2-Extension-Name: com/example/util2
    Util2-Specification-Version: 1.4
    Util3-Extension-Name: com/example/util3
    Util3-Specification-Version: 1.4
  META-INF/ejb-jar.xml
```

**util1.jar:**

```
META-INF/MANIFEST.MF:
  Extension-Name: com/example/util1
  Specification-Title: example.com's util package
  Specification-Version: 1.4
  Specification-Vendor: example.com
  Implementation-Version: build96
```

```
util2.jar:
META-INF/MANIFEST.MF:
  Extension-Name: com/example/util2
  Specification-Title: example.com's util package
  Specification-Version: 1.4
  Specification-Vendor: example.com
  Implementation-Version: build96
```

```
util3.jar:
META-INF/MANIFEST.MF:
  Extension-Name: com/example/util3
  Specification-Title: example.com's util package
  Specification-Version: 1.4
  Specification-Vendor: example.com
  Implementation-Version: build96
```

2. Create a shared library that represents the library file assembled in step 1. This installs the library file as a WebSphere Application Server shared library.
3. Assemble the application, declaring in the application manifest file dependencies to the library files named the manifest created for step 1.
4. Install the application on the server.

During application installation, the shared library .jar files are added to the class path of the application class loader.

## Library reference collection

Use this page to view and manage library references that define how to use global libraries. For example, you can use this page to associate shared library files with a deployed application.

To view this administrative console page, click **Applications > Enterprise Applications > *application\_name* > Libraries**.

If no shared libraries are defined, after you click **Add** a message is displayed stating that you must define a shared library before you can create a library reference. A shared library is a container-wide library file that can be used by deployed applications. To define a shared library, click **Environment > Shared Libraries** and specify the scope of the container. Then, click **New** and specify a name and one or more paths for the shared library. After you define a shared library, return to this page, click **Add**, and create a library reference.

### Library name

Specifies a name for the library reference.

### Library reference settings

Use this page to define library references, which specify how to use global libraries.

To view this administrative console page, click **Applications > Enterprise Applications > *application\_name* > Libraries > *library\_reference\_name***. A shared library must be defined to view this page.

A shared library is a container-wide library file that can be used by deployed applications. To define a shared library, click **Environment > Shared Libraries** and specify the scope of the container. Then, click **New** and specify a name and one or more paths for the shared library.

#### **Library name:**

Specifies the name of the shared library to use for the library reference.

<b>Data type</b>	String
------------------	--------

---

## Environment: Resources for learning

Use the following links to find relevant supplemental information about configuring the WebSphere Application Server environment. 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.

### Programming instructions and examples

- WebSphere Application Server education at <http://www.ibm.com/software/websphere/technical>.

### Administration

- Listing of all IBM WebSphere Application Server Redbooks at <http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere>.

---

## Chapter 10. Working with server configuration files

Application server configuration documents define the available application servers, their configurations, and their contents.

You should periodically save changes to your administrative configuration. You can change the default locations of configuration files, as needed.

1. Edit configuration files. The master repository is comprised of .xml configuration files. You can edit configuration files using the administrative console, scripting, wsadmin commands, programming, or by editing a configuration file directly.
2. Save changes made to configuration files. Using the console, you can save changes as follows:
  - a. Click **Save** on the taskbar of the administrative console.
  - b. On the Save page, click **Save**.
3. Handle temporary configuration files resulting from a session timing out.
4. Change the location of temporary configuration files.
5. Change the location of backed-up configuration files.
6. Change the location of temporary workspace files.
7. Back up and restore configurations.

---

### Configuration documents

WebSphere Application Server stores configuration data for servers in several documents in a cascading hierarchy of directories. The configuration documents describe the available application servers, their configurations, and their contents. Most configuration documents have XML content.

#### Hierarchy of directories of documents

The cascading hierarchy of directories and the documents' structure support multinode replication to synchronize the activities of all servers in a cell. In a Network Deployment environment, changes made to configuration documents in the cell repository, are automatically replicated to the same configuration documents that are stored on nodes throughout the cell.

At the top of the hierarchy is the **cells** directory. It holds a subdirectory for each cell. The names of the cell subdirectories match the names of the cells. For example, a cell named *cell1* has its configuration documents in the subdirectory *cell1*.

On the Network Deployment node, the subdirectories under the cell contain the entire set of documents for every node and server throughout the cell. On other nodes, the set of documents is limited to what applies to that specific node. If a configuration document only applies to *node1*, then that document exists in the configuration on *node1* and in the Network Deployment configuration, but not on any other node in the cell.

Each cell subdirectory has the following files and subdirectories:

- The *cell.xml* file, which provides configuration data for the cell
- Files such as *security.xml*, *virtualhosts.xml*, *resources.xml*, and *variables.xml*, which provide configuration data that applies across every node in the cell
- The **clusters** subdirectory, which holds a subdirectory for each cluster defined in the cell. The names of the subdirectories under clusters match the names of the clusters.

Each cluster subdirectory holds a *cluster.xml* file, which provides configuration data specifically for that cluster.

- The **nodes** subdirectory, which holds a subdirectory for each node in the cell. The names of the nodes subdirectories match the names of the nodes.

Each node subdirectory holds files such as `variables.xml` and `resources.xml`, which provide configuration data that applies across the node. Note that these files have the same name as those in the containing cell's directory. The configurations specified in these node documents override the configurations specified in cell documents having the same name. For example, if a particular variable is in both cell- and node-level `variables.xml` files, all servers on the node use the variable definition in the node document and ignore the definition in the cell document.

Each node subdirectory holds a subdirectory for each server defined on the node. The names of the subdirectories match the names of the servers. Each server subdirectory holds a `server.xml` file, which provides configuration data specific to that server. Server subdirectories might hold files such as `security.xml`, `resources.xml` and `variables.xml`, which provide configuration data that applies only to the server. The configurations specified in these server documents override the configurations specified in containing cell and node documents having the same name.

- The **applications** subdirectory, which holds a subdirectory for each application deployed in the cell. The names of the applications subdirectories match the names of the deployed applications.

Each deployed application subdirectory holds a `deployment.xml` file that contains configuration data on the application deployment. Each subdirectory also holds a **META-INF** subdirectory that holds a J2EE application deployment descriptor file as well as IBM deployment extensions files and bindings files.

Deployed application subdirectories also hold subdirectories for all `.war` and entity bean `.jar` files in the application. Binary files such as `.jar` files are also part of the configuration structure.

An example file structure is as follows:

```
cells
  cell1
    cell.xml resources.xml virtualhosts.xml variables.xml security.xml
    nodes
      nodeX
        node.xml variables.xml resources.xml serverindex.xml
        serverA
          server.xml variables.xml
        nodeAgent
          server.xml variables.xml
      nodeY
        node.xml variables.xml resources.xml serverindex.xml
    applications
      sampleApp1
        deployment.xml
        META-INF
          application.xml ibm-application-ext.xml ibm-application-bnd.xml
      sampleApp2
        deployment.xml
        META-INF
          application.xml ibm-application-ext.xml ibm-application-bnd.xml
```

## Changing configuration documents

You can use one of the administrative tools (console, wsadmin, Java APIs) to modify configuration documents or edit them directly. It is preferable to use the administrative console because it validates changes made to configurations. "Configuration document descriptions" states whether you can edit a document using the administrative tools or must edit it directly.

---

## Configuration document descriptions

Most configuration documents have XML content. The table below describes the documents and states whether you can edit them using an administrative tool or must edit them directly.

If possible, edit a configuration document using the administrative console because it validates any changes that you make to configurations. You can also use one of the other administrative tools (wsadmin or Java APIs) to modify configuration documents. Using the administrative console or wsadmin scripting to update configurations is less error prone and likely quicker and easier than other methods.

However, you cannot edit some files using the administrative tools. Configuration files that you must edit manually have an X in the **Manual editing required** column in the table below.

## Document descriptions

(Locations split for publishing)

Configuration file	Locations	Purpose	Manual editing required
admin-authz.xml	config/cells/ <i>cell_name/</i>	Define a role for administrative operation authorization.	X
app.policy	config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i>	Define security permissions for application code.	X
cell.xml	config/cells/ <i>cell_name/</i>	Identify a cell.	
cluster.xml	config/cells/ <i>cell_name/</i> clusters/ <i>cluster_name/</i>	Identify a cluster and its members and weights.  This file is only available with the Network Deployment product.	
deployment.xml	config/cells/ <i>cell_name/</i> applications/ <i>application_name/</i>	Configure application deployment settings such as target servers and application-specific server configuration.	
filter.policy	config/cells/ <i>cell_name/</i>	Specify security permissions to be filtered out of other policy files.	X
integral-jms-authorizations.xml	config/cells/ <i>cell_name/</i>	Provide security configuration data for the integrated messaging system.	X
library.policy	config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i>	Define security permissions for shared library code.	X
multibroker.xml	config/cells/ <i>cell_name/</i>	Configure a data replication message broker.	
namestore.xml	config/cells/ <i>cell_name/</i>	Provide persistent name binding data.	X
naming-authz.xml	config/cells/ <i>cell_name/</i>	Define roles for a naming operation authorization.	X
node.xml	config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i>	Identify a node.	
pmirm.xml	config/cells/ <i>cell_name/</i>	Configure PMI request metrics.	X

resources.xml	config/cells/ cell_name/ config/cells/ cell_name/ nodes/node_name/ config/cells/ cell_name/ nodes/node_name/ servers/ server_name/	Define operating environment resources, including JDBC, JMS, JavaMail, URL, JCA resource providers and factories.	
security.xml	config/cells/ cell_name/	Configure security, including all user ID and password data.	
server.xml	config/cells/ cell_name/ nodes/ node_name/ servers/ server_name/	Identify a server and its components.	
serverindex.xml	config/cells/ cell_name/ nodes/ node_name/	Specify communication ports used on a specific node.	
spi.policy	config/cells/ cell_name/ nodes/ node_name/	Define security permissions for service provider libraries such as resource providers.	X
variables.xml	config/cells/ cell_name/ config/cells/ cell_name/ nodes/ node_name/ config/cells/ cell_name/ nodes/node_name/ servers/ server_name/	Configure variables used to parameterize any part of the configuration settings.	
virtualhosts.xml	config/cells/ cell_name/	Configure a virtual host and its MIME types.	

---

## Object names

When you create a new object using the administrative console or a wsadmin command, you often must specify a string for a name attribute. Most characters are allowed in the name string. However, the name string cannot contain the following characters. The name string also cannot contain leading and trailing spaces.

/	forward slash
\	backslash
*	asterisk
,	comma
:	colon



;	semi-colon
=	equal sign
+	plus sign
?	question mark
	vertical bar
<	left angle bracket
>	right angle bracket
&	ampersand (and sign)
%	percent sign
'	single quote mark
"	double quote mark
]]>	No specific name exists for this character combination.
.	period (not valid if first character; valid if a later character)

---

## Configuration repositories

A configuration repository stores configuration data. By default, configuration repositories reside in the *config* subdirectory of the product installation root directory.

---

## Handling temporary configuration files resulting from session timeout

If the console is not used for 15 minutes or more, the session times out. The same thing happens if you close the browser window without saving the configuration file. Changes to the file are saved to a temporary file when the session times out, after 15 minutes.

When a session times out, the configuration file in use is saved under the *userid/timeout* directory under the *ServletContext*'s temp area. This is the value of the *javax.servlet.context.tempdir* attribute of the *ServletContext*. By default, it is: *install\_root/temp/hostname/Administration/admin/admin.war*

You can change the temp area by specifying it as a value for the *tempDir* init-param of the action servlet in the deployment descriptor (*web.xml*) of the administrative application.

The next time you log on to the console, you are prompted to load the saved configuration file. If you decide to load the saved file:

1. If a file with the same name exists in the *install\_root/config* directory, that file is moved to the *userid/backup* directory in the temp area.
2. The saved file is moved to the *install\_root/config* directory.
3. The file is then loaded.

If you decide not to load the saved file, it is deleted from the *userid/timeout* directory in the temp area.

The configuration file is also saved automatically when the same user ID logs into the non-secured console again, effectively starting a different session. This process is equivalent to forcing the existing user ID out of session, similar to a session timing out.

---

## Changing the location of temporary configuration files

The configuration repository uses copies of configuration files and temporary files while processing repository requests. It also uses a backup directory while managing the configuration. You can change the default locations of these files from the configuration directory to a directory of your choice using system variables or the administrative console.

The default location for the configuration temporary directory is `CONFIG_ROOT/temp`. Change the location by doing either of the following:

- Set the system variable `was.repository.temp` to the location you want for the repository temporary directory. Set the system variable when launching a Java process using the `-D` option. For example, to set the default location of the repository temporary directory, use the following option:  
`-Dwas.repository.temp=%CONFIG_ROOT%/temp`
- Use the administrative console to change the location of the temporary repository file location for each server configuration. For example, on the Network Deployment product, to change the setting for a deployment manager, do the following:
  1. Click **System Administration > Deployment Manager** in the navigation tree of the administrative console. Then, click **Administration Services, Repository Service, and Custom Properties**.
  2. On the Properties page, click **New**.
  3. On the settings page for a property, define a property for the temporary file location. The key for this property is `was.repository.temp`. The value can include WebSphere Application Server variables such as `${WAS_TEMP_DIR}/config`. Then, click **OK**.

The system property set using the first option takes precedence over the configuration property set using the second option.

---

## Changing the location of backed-up configuration files

During administrative processes like adding a node to a cell or updating a file, configuration files are backed up to a backup location. The default location for the backup configuration directory is `CONFIG_ROOT/backup`. Change the location by doing either of the following:

- Set the system variable `was.repository.backup` to the location you want as the repository backup directory. Set the system variable when launching a Java process using the `-D` option. For example, to set the default location of the repository backup directory, use the following option:  
`-Dwas.repository.backup=%CONFIG_ROOT%/backup`
- Use the administrative console to change the location of the repository backup directory for each server configuration. For example, on the Network Deployment product, do the following to change the setting for a deployment manager:
  1. Click **System Administration > Deployment Manager** in the navigation tree of the administrative console. Then, click **Administration Services, Repository Service, and Custom Properties**.
  2. On the Properties page, click **New**.
  3. On the settings page for a property, define a property for the backup file location. The key for this property is `was.repository.backup`. The value can include WebSphere Application Server variables such as `${WAS_TEMP_DIR}/backup`. Then, click **OK**.

The system property set using the first option takes precedence over the configuration property set using the second option.

---

## Changing the location of temporary workspace files

The administrative console workspace allows client applications to navigate the configuration. Each workspace has its own repository location defined either in the system property or the property passed to a workspace manager when creating the workspace: `workspace.user.root` or `workspace.root`, which is calculated as `%workspace.root%/user_ID/workspace/wstemp`.

The default workspace root is calculated based on the user installation root: `%user.install.root%/wstemp`. You can change the default location of temporary workspace files by doing the following:

- Distributed platforms: Change the setting for the system variable `workspace.user.root` or `workspace.root` so its value is no longer set to the default location. Set the system variable when launching a Java process using the `-D` option. For example, to set the default location the full path of the root of all users' directories, use the following option:

## Backing up and restoring administrative configurations

WebSphere Application Server represents its administrative configurations as XML files. You should back up configuration files on a regular basis.

1. Run the backupConfig command to back up configuration files.
2. Run the restoreConfig command to restore configuration files. Specify backup files that do not contain invalid or inconsistent configurations.

---

## Transformation of configuration files

The WebSphere Application Server master configuration repository stores configuration files for all the nodes in the cell. When you upgrade the deployment manager from one release of WebSphere Application Server to another, the configuration files that are stored in the master repository for the nodes on the old release are converted into the format of the new release.

With this conversion, the deployment manager can process the configuration files uniformly. However, nodes on an old release cannot readily use configuration files that are in the format of the new release. WebSphere Application Server addresses the problem when it synchronizes the configuration files from the master repository to a node on an old release. The configuration files are first transformed into the old release format before they ship to the node. WebSphere Application Server performs the following transformations on configuration documents:

- Changes the XML name space from the format of the new release to the format of the old release
- Strips out attributes of cell-level documents that are applicable to the new release only
- Strips out new resource definitions that are not understood by old release nodes

---

## Server configuration files: Resources for learning

Use the following links to find relevant supplemental information about administering WebSphere Application Server configuration files. 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.

View links to additional information:

### Administration

- IBM WebSphere Application Server Redbooks at <http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere>.

This site contains a listing of all WebSphere Application Server Redbooks.

- IBM developerWorks WebSphere at <http://www.software.ibm.com/wsdd>.

This site is the home of technical information for developers working with WebSphere products. You can download WebSphere software, take a fast path to developerWorks 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.

- WebSphere Application Server Support page at <http://www.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 11. Administering application servers

An application server configuration provides settings that control how an application server provides services for running enterprise applications and their components.

This section describes how to create and configure application servers in an existing application server environment.

A WebSphere Application Server administrator can configure one or more application servers and perform tasks such as the following:

1. Create application servers.
2. Manage application servers.
3. Configure transport chains.
4. Develop custom services.
5. Define processes for the application server. As part of defining processes, you can define:
6. Use the Java virtual machine.

After preparing a server, deploy an application or component on the server. See “Preparing to host applications” on page 159 for a sample procedure that you might follow in configuring the application server runtime and resources.

---

### Application servers

Application servers extend a Web server’s capabilities to handle Web application requests, typically using Java technology. An application server makes it possible for a server to generate a dynamic, customized response to a client request.

For example, suppose--

1. A user at a Web browser on the public Internet visits a company Web site. The user requests to use an application that provides access to data in a database.
2. The user request flows to the Web server.
3. The Web server determines that the request involves an application containing resources not handled directly by the Web server (such as servlets). It forwards the request to a WebSphere Application Server product.
4. The WebSphere Application Server product forwards the request to one of its application servers on which the application is running.
5. The invoked application then processes the user request. For example:
  - An application servlet prepares the user request for processing by an enterprise bean that performs the database access.
  - The application produces a dynamic Web page containing the results of the user query.
6. The application server collaborates with the Web server to return the results to the user at the Web browser.

The WebSphere Application Server product provides multiple application servers that can be either separately configured processes or nearly identical clones.

---

### Creating application servers

For the Express and Base products, you must use scripting to create a new application server (see the *Administering applications and their environment* PDF). The server you create cannot be managed using the administrative console. Also note that the only server you can manage using the administrative console is the default server (server1).

With WebSphere Application Server Version 6.0, you can now upgrade a portion of the nodes in a cell, while leaving others at the older release level. This means that, for a period of time, you may be managing servers that are at the current release and servers that are running the newer release in the same cell. In this mixed environment, there are restrictions on what you can do with servers at the older release level. They are:

- You can only create new server definitions on nodes that are running WebSphere Application Server Version 6.0.
- When you create a new server definition, you must use a server configuration template, and that template must be created from a WebSphere Application Server Version 6.0 server instance. You cannot create (or use) a template from a WebSphere Application Server Version 5.x server instance.

There are no restrictions on what you can do with the servers running on the newer release level.

The steps below describe how to use the Create New Application Server page.

1. Create the new application server using the wsadmin **createApplicationServer** command. For information, see the *Administering applications and their environment* PDF.
2. To use multiple language encoding support in the administrative console, configure an application server with UTF-8 encoding enabled.

The new application server appears in the list of servers on the Application Servers page.

Note that the application server created has many default values specified for it. An application server has many properties that can be set and creating an application server on the Create New Application Server page specifies values for only a few of the important properties. To view all of the properties of your application server and to customize your application server further, click on the name of your application server on the Application Servers page and change the settings for your application server as needed.

## Configuring application servers for UTF-8 encoding

To use multiple language encoding support in the administrative console, you must configure an application server with UTF-8 encoding enabled.

1. Create an application server or use an existing application server.
2. On the Application Server page, click on the name of the server you want enabled for UTF-8.
3. On the settings page for the selected application server, under Server Infrastructure, click **Java and Process Management > Process Definition**.
4. On the Process Definition page, click **Java Virtual Machine**.
5. On the Java Virtual Machine page, specify `-Dclient.encoding.override=UTF-8` for **Generic JVM Arguments** and click **OK**.
6. Click **Save** on the console task bar.
7. Restart the application server.

Note that the `autoRequestEncoding` option does not work with UTF-8 encoding enabled. The default behavior for WebSphere Application Server is, first, to check if `charset` is set on content type header. If it is, then the product uses content type header for character encoding; if it is not, then the product uses character encoding set on server using the system property `default.client.encoding`. If `charset` is not present and the system property is not set, then the product uses ISO-8859-1. Enabling `autoRequestEncoding` on a Web module changes the default behavior: if `charset` is not present on an incoming request header, the product checks the `Accept-Language` header of the incoming request and does encoding using the first language found in that header. If there is no `charset` on content type header and no `Accept language` header, then the product uses character encoding set on server using the system property `default.client.encoding`. As with the default behavior, if `charset` is not present and the system property is not set, then the product uses ISO-8859-1.

---

## Managing application servers

To view information about an application server, use the Application Servers panel on the administrative console.

You must use scripting to create a new application server (see the *Administering applications and their environment* PDF). The server you create can not be managed using the administrative console. Also note that the only server you can manage using the administrative console is the default server (server1).

1. Access the Application Servers page. Click **Servers > Application Servers** in the console navigation tree.
2. View information about application servers.

The Application Servers page lists application servers in the cells holding the application servers.

To view additional information about a particular application server or to further configure an application server, click on the application server name under **Name**. This accesses the settings page for an application server.

To view product information for an application server:

- a. Verify that the application server is running.
- b. Display the **Runtime** tab on the settings page for an application server.
- c. Click **Product Information**.

The Product Information page displayed lists the WebSphere Application Server products installed for the application server, the version and build levels for the products, the build dates, and any interim fixes applied to the application server.

**Note:** You can also get this information by using the **versionInfo** command. For more information, see the *Installing your application serving environment* PDF.

3. Create an application server using the wsadmin **createApplicationServer** command. For information, see the *Administering applications and their environment* PDF.
4. Monitor the running of application servers.

## Server collection

Use this page to view information about and manage application servers, generic servers, Java Message Service (JMS) servers, and Web servers.

### Application Servers

The Application Servers page lists the application servers in the cell. You can use this page to create new application servers, create application server templates, or delete existing application servers. You can also use this page to start and stop these application servers.

To view this administrative console page, click **Application Servers**.

### Generic Servers

The Generic Servers page lists the generic servers in the cell. You can use this page to create new generic servers, create generic server templates, or delete existing generic servers. You can also use this page to start and stop these generic servers.

The Network Deployment product also shows the status of the generic servers. The status indicates whether a server is running, stopped, or encountering problems.

You can use this page to add or delete application servers.



To view this administrative console page, click **Generic Servers**.

## Java Message Service (JMS) Servers

The JMS Servers page lists the JMS servers in the cell. You can use this page to start and stop these JMS servers.

Each JMS server provides the functions of the JMS provider for a node in your administrative domain. There can be at most one JMS server on each node in the administration domain, and any application server within the domain can access JMS resources served by any JMS server on any node in the domain.

**Note:** JMS servers apply only to WebSphere Application Server Version 5.x nodes. You cannot create a JMS server on a node that is running WebSphere Application Server 6.0, but the existing Version 5.x JMS servers will continue to be displayed, and you can modify their properties. You can also delete Version 5.x JMS servers.

To view this administrative console page, click **JMS Servers**.

## Web Servers

The Web Servers page lists the Web servers in your administrative domain. You can use this page to generate and propagate a Web server plug-in configuration file, create new Web servers, create new Web server templates, or delete existing Web servers. You can also use this page to start and stop these Web servers.

To view this administrative console page, click **Web Servers**.

### Name

Specifies a logical name for the server. For WebSphere Application Server, server names must be unique within a node.

### Node

Specifies the name of the node holding the server.

### Version

Specifies the version of the WebSphere Application Server product on which the server runs.

### Status

Indicates whether the server is started or stopped. (Network Deployment only)

Note that if the status is *Unavailable*, the node agent is not running in that node and you must restart the node agent before you can start the server.

## Application server settings

An application server is a server which provides services required to run enterprise applications. Use this page to view or change the settings of an application server instance.

To view this administrative console page, click **Servers > Application Servers >server\_name**.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information. The **Runtime** tab is available only when the server is running.

### Name:



Specifies a logical name for the server. Server names must be unique within a node. However, for multiple nodes within a cluster, you may have different servers with the same server name as long as the server and node pair are unique.

For example, a server named *server1* in a node named *node1* in the same cluster with a server named *server1* in a node named *node2* is allowed. Configuring two servers named *server1* in the same node is not allowed. WebSphere Application Server uses the server name for administrative actions, such as referencing the server in scripting.

<b>Data type</b>	String
<b>Default</b>	server1

### ***Run in development mode:***

Enabling this option may reduce the startup time of an application server. This may include JVM settings such as disabling bytecode verification and reducing JIT compilation costs. Do not enable this setting on production servers. This setting is only available on application servers running WebSphere Application Server Version 6.0 and later.

Specifies that you want to use the JVM settings **-Xverify** and **-Xquickstart** on startup. After selecting this option, save the configuration and restart the server to activate development mode.

The default setting for this option is `false`, which indicates that the server will not be started in development mode. Setting this option to `true` specifies that the server will be started in development mode (with settings that will speed server startup time).

<b>Data type</b>	Boolean
<b>Default</b>	false

### ***Parallel start:***

Select this field to start the server on multiple threads. This might shorten the startup time.

Specifies that you want the server components, services, and applications to start in parallel rather than sequentially.

The default setting for this option is `true`, which indicates that the server be started using multiple threads. Setting this option to `false` specifies that the server will not be started in using multiple threads (which may lengthen startup time).

Note that the order in which the applications start depends on the weights you assigned to each them. Applications that have the same weight are started in parallel. You set an application's weight with the *Starting weight* option on the **Applications > Enterprise Applications > application\_name** page of the Administrative Console. For more information about the *Starting weight* option, see the *Installing your application serving environment* PDF.

<b>Data type</b>	Boolean
<b>Default</b>	true

### ***Class loader policy:***

Select whether there is a single class loader to load all applications or a different class loader for each application.

### ***Class loading mode:***

Specifies whether the class loader should search in the parent class loader or in the application class loader first to load a class. The standard for Developer Kit class loaders and WebSphere Application Server class loaders is Parent first.

If you select Parent last, your application can override classes contained in the parent class loader, but this action can potentially result in ClassCastException or linkage errors if you have mixed use of overridden classes and non-overridden classes.

**Process Id:**

The native operating system's process ID for this server.

The process ID property is read only. The system automatically generates the value.

**Cell name:**

The name of the cell in which this server is running.

The Cell name property is read only.

**Node name:**

The name of the node in which this server is running.

The Node name property is read only.

**State:**

The run-time execution state for this server.

The State property is read only.

**Ports collection:**

Use this page to view and manage communication ports used by run-time components running within a process. Communication ports provide host and port specifications for a server.

To view this administrative console page, click **Servers > Application Servers > *server\_name* Communications > Ports**.

Note that this page displays only when you are working with ports for application servers.

*Port Name:*

Specifies the name of a port. Each name must be unique within the server.

*Host:*

Specifies the IP address, domain name server (DNS) host name with domain name suffix, or just the DNS host name, used by a client to request a resource (such as the naming service, or administrative service).

*Port:*

Specifies the port for which the service is configured to accept client requests. The port value is used in conjunction with the host name.

*Transport Details:*

Provides a link to the transport chains associated with this port. If no transport chains are associated with this port, the string "No associated transports" appears in this column.

*Ports settings:*

Use this to view and change the configuration for a communication port used by run-time components running within a process. A communication port provides host and port specifications for a server.

For base WebSphere Application Server, you can view this administrative console page, by clicking **Servers > Application Servers > *server\_name* > Ports > *port\_name***

*Port Name:*

Specifies the name of the port. The name must be unique within the server.

Note that this field displays only when you are defining a port for an application server. You can select a radio button to:

**Well-known Port**

select a previously defined port from the drop down list

**User-defined Port**

create a port with a new name by entering the name in the text box

**Data type** String

*Host:*

Specifies the IP address, domain name server (DNS) host name with domain name suffix, or just the DNS host name, used by a client to request a resource (such as the naming service, administrative service, or JMS broker).

For example, if the host name is *myhost*, the fully qualified DNS name can be *myhost.myco.com* and the IP address can be *155.123.88.201*.

Host names on the ports can be resolvable names or IP addresses. The server will bind to the specific host name or IP address that is supplied. That port will only be accessible through the IP address that is resolved from the given host name or IP address. The IP address may be of the IPv4 (Internet Protocol Version 4) format for all platforms, and IPv6 (Internet Protocol Version 6) format on specific operating systems where the server supports IPv6.

**Data type** String  
**Default** \* (asterisk)

*Port:*

Specifies the port for which the service is configured to accept client requests. The port value is used in conjunction with the host name.

Port numbers in the server can be reused among multiple ports as long as they have host names that resolve to unique IP addresses and there is not a port with the same port number and a wildcard ( \* ) host name. A port number is valid in the range of 0 and 65535. 0 specifies that the server should bind to any ephemeral port available.

**Data type** Integer  
**Default** None

Range	1-65536
-------	---------

***Custom property collection:***

Use this page to view and manage arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties.

The administrative console contains several Custom Properties pages that work similarly. To view one of these administrative pages, click a **Custom Properties** link.

*Name:*

Specifies the name (or key) for the property.

Do not start your property names with `was.` because this prefix is reserved for properties that are predefined in WebSphere Application Server.

*Value:*

Specifies the value paired with the specified name.

*Description:*

Provides information about the name-value pair.

*Custom property settings:*

Use this page to configure arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available in the administrative console.

For base WebSphere Application Server you can view this administrative console page by, clicking **Servers > Application Servers > *server\_name***. Then, under Server Infrastructure, click **Administration > Custom Properties**

*Name:*

Specifies the name (or key) for the property.

Do not start your property names with `was.` because this prefix is reserved for properties that are predefined in WebSphere Application Server.

**Data type** String

*Value:*

Specifies the value paired with the specified name.

**Data type** String

*Description:*

Provides information about the name and value pair.

**Data type** String

***Server component collection:***

Use this page to view information about and manage server component types such as application servers, messaging servers, or name servers.

To view this administrative console page, click **Servers > Application Servers >server\_name**. Then, under Server Infrastructure, click **Administration > Server Components**.

*Type:*

Specifies the type of internal server.

*Server component settings:*

Use this page to view or configure a server component instance.

To view this administrative console, click **Servers > Application Servers >server\_name**. Then, under Server Infrastructure, click **Administration > Server Components >server\_component\_name**.

*Name:*

Specifies the name of the component.

**Data type** String

*Initial State:*

Specifies the desired state of the component when the server process starts. The options are: *Started* and *Stopped*. The default is *Started*.

**Data type** String  
**Default** Started

***Thread pool collection:***

Use this page to select or create a group of threads that an application server uses. Requests are sent to the server through any of the HTTP transports. A thread pool enables components of the server to reuse threads to eliminate the need to create new threads at run time. Creating new threads expends time and resources.

To view this administrative console page, click **Servers > Application Servers >server\_name > Thread Pools**. (You can reach this page through more than one navigational route.)

*Thread pool settings:*

Use this page to configure a group of threads that an application server uses. Requests are sent to the server through any of the HTTP transport channels or HTTP transports. A thread pool enables components of the server to reuse threads to eliminate the need to create new threads at run time. Creating new threads expends time and resources.

To view this administrative console page, click **Servers > Application Servers >server\_name > Thread Pools**, then select the thread pool. (You can reach this page through more than one navigational route.)

#### *Minimum size:*

Specifies the minimum number of threads to allow in the pool.

<b>Data type</b>	Integer
<b>Default</b>	10

#### *Maximum size:*

Specifies the maximum number of threads to allow in the pool.

If your Tivoli Performance Viewer shows the Percent Maxed metric to remain consistently in the double digits, consider increasing the Maximum size. The Percent Maxed metric indicates the amount of time that the configured threads are used. If there are several simultaneous clients connecting to the server-side ORB, increase the size to support up to 1000 clients.

<b>Data type</b>	Integer
<b>Default</b>	50
<b>Recommended</b>	50 (25 on Linux systems)

#### *Thread inactivity timeout:*

Specifies the number of milliseconds of inactivity that should elapse before a thread is reclaimed. A value of 0 indicates not to wait and a negative value (less than 0) means to wait forever.

**Note:** The administrative console does not allow you to set the inactivity timeout to a negative number. To do this you must modify the value directly in the *server.xml* file.

<b>Data type</b>	Integer
<b>Units</b>	Milliseconds
<b>Default</b>	3500

#### *Allow thread allocation beyond maximum thread size:*

Specifies whether the number of threads can increase beyond the maximum size configured for the thread pool.

<b>Data type</b>	Boolean
<b>Default</b>	Not enabled (false)

## **Generic server settings**

Use this page to view or change the settings of a generic server.

A generic server is a server that is managed in the WebSphere Application Server administrative domain, although it is not a server that is supplied by the WebSphere Application Server product. The generic server can be any server or process that is necessary to support the Application Server environment, including a Java server, a C or C++ server or process, or a Remote Method Invocation (RMI) server.

To view this administrative console page, click **Servers > Generic Servers > *server\_name***.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information. The **Runtime** tab is available only when the server is running.

#### **Name:**

Specifies a logical name for the generic server.

It is highly recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular WebSphere Application Servers. This will enable you to quickly determine whether to use the Terminate or Stop button in the administrative console to stop a specific application server.

You must use the Terminate button to stop a generic application server.

<b>Data type</b>	String
<b>Default</b>	

## Starting servers

Starting a server starts a new server process based on the process definition settings of the current server configuration.

If you need to restart a server, follow the directions in this article for starting servers. The procedure that applies to starting servers also applies to restarting servers.

**Note:** If you created a new server definition using a base WebSphere Application Server, you cannot start, stop, or manage the new server using the original base Application Server.

There are several options for starting an Application Server:

- **Windows** If you are using Windows, you can use the **Start** menu to start the Application Server. If you are using the Express version of the product, click **Start > Programs > IBM WebSphere > Express v6.0 > Start the server**. You can check that the server has successfully started by checking the startServer.log file. If the server has successfully started, the last two lines of the startServer.log file reads:

```
Server launched. Waiting for initialization status.  
Server server1 open for e-business; process id is 1932.
```

The startServer.log file is located in the <drive>:\Program Files\IBM\WebSphere\AppServer\profiles\profile\_name\logs\server1 directory if you have installed your server with the default settings. The server name and process id vary depending on your settings.

- On distributed platforms, use the **startServer** command to start an Application Server from the command line.
- On AIX, you can use the command line to start the server. Use the **startServer** command from the /usr/WebSphere/AppServer/bin directory, as shown below. To start a server that is associated with a non-default profile, issue the **startServer** command from the /opt/WebSphere/AppServer/profiles/profile\_name/bin directory.

```
# ./startServer.sh server1
```

You can check that the server has successfully started by checking the startServer.log file. If the server has successfully started, the last two lines of the startServer.log file reads:

```
Server launched. Waiting for initialization status.  
Server server1 open for e-business; process id is 1932.
```

On AIX, the startServer.log file is located in the /usr/IBM/WebSphere/AppServer/profiles/profile\_name/logs/server1/ directory.

- Start an Application Server for tracing and debugging.  
To start the Application Server with standard Java debugging enabled:

1. Click **Servers > Application Servers** from the administrative console navigation tree. Then, click the Application Server whose processes you want to trace and debug, then **Java and Process Management > Process Definition > Java Virtual Machine**.
2. On the Java Virtual Machine page, place a checkmark in the check box for the **Debug Mode** setting to enable the standard Java debugger. If needed, set debug arguments. Then, click **OK**.
3. Save the changes to a configuration file.
4. Stop the Application Server.
5. Start the Application Server again as described previously.

Once the server is started, you can install your applications.

## Running application servers from a non-root user

By default, each base WebSphere Application Server server on a Linux and UNIX platform uses the root user ID to run all application server processes. However, you can run all application server processes under the same non-root user and user group. This task describes how to run an application server process from a non-root user.

If global security is enabled, the user registry must not be Local OS. Using the Local OS user registry requires the application server to run as root. Refer to the *Securing applications and their environment* PDF for details.

Run your application servers as non-root when you no longer want to use root authority. For security or administrative reasons, you may want to change to non-root user IDs. Perform this task at any time to change the permissions of an application server. You must restart the application server in order for the changes to take effect. .

**Note:** If you are using the Tivoli Access Manager (TAM) to perform authentication or authorization for WebSphere Application Server, it is important to be aware of potential permissions problems. For more information, see the *Securing applications and their environment* PDF.

For the following steps, assume that:

- was1 is the user to run the application server
- wasgroup is the primary user group for user was1
- wasnode is the node name
- server1 is the application server
- /opt/WebSphere/AppServer is the installation root
- nodeProfile1 is the profile name.

**Note:** For information about creating a profile, see the *Administering applications and their environment* PDF.

To configure an application server to run as non-root, complete the following steps.

1. Log on to the application server system as the root user.
2. Create the user ID was1 with a primary user group of wasgroup. The user ID, was1, is an example. You can name the user something else.
3. Log off and back on as root.
4. Start server1 as root. Run the startServer.sh script from the /bin directory of the installation root:
 

```
startServer.sh server1
```
5. Specify user and group ID values for the **Run As User** and **Run As Group** settings for a server:
  - a. Start the administrative console.
  - b. Go to the Process execution page of the administrative console. You must define all three properties in the following table. Click **Servers > Application Servers > server1 > Server Infrastructure > Java and Process Management > Process Execution** and change all of the



following values:

Property	Value
Run As User	was1
Run As Group	wasgroup
UMASK	022

- c. Click **OK**.
- d. Save the configuration.
6. Stop the application server. Use the `stopServer.sh` script from the `/bin` directory of the installation root:  

```
stopServer.sh server1
```
7. Change file permissions as the root user. The following example assumes that the installation root directory for WebSphere Application Server is `/opt/WebSphere/AppServer`:  

```
chgrp wasgroup /opt/WebSphere
chgrp wasgroup /opt/WebSphere/AppServer
chgrp -R wasgroup /opt/WebSphere/AppServer/cloudscape
chgrp -R wasgroup /opt/WebSphere/AppServer/profiles/nodeProfile1
chmod g+wr /opt/WebSphere
chmod g+wr /opt/WebSphere/AppServer
chmod -R g+wr /opt/WebSphere/AppServer/cloudscape
chmod -R g+wr /opt/WebSphere/AppServer/profiles/nodeProfile1
```
8. Log on to the application server system as `was1`.
9. Start `server1` as `was1`. Run the `startServer.sh` script from the `/bin` directory of the installation root:  

```
startServer.sh server1
```
10. If creating another server with a different user ID, follow this procedure again for the new user ID and server name.  
The two user IDs must share the same group, `wasgroup`.

You can start an application server from a non-root user.

## Detecting and handling problems with run-time components

You must monitor the status of run-time components to ensure that, once started, they remain operational as needed.

1. Regularly examine the status of run-time components. Browse messages displayed under **Websphere Runtime Messages** in the status area at the bottom of the console. The run-time event messages marked with a red **X** provide detailed information on event processing. You can also use the Logging and Tracing page of the administrative console to monitor the status of run-time components. Click **Troubleshooting > Logs and Trace** in the console navigation tree to access the page.
2. If an application stops running when it should be operational, examine the application's status on an Applications page and try restarting the application.
3. If the run-time components do not restart, reexamine the messages and read information on problem determination to help you to restart the components.

## Stopping servers

Stopping an application server stops a server process based on the process definition settings in the current application server configuration.

- **Windows** In Windows, you can use the Start menu to stop your application server. Click **Start > Programs > IBM WebSphere > Express v6.0 > Stop the server**. When the server stops successfully, the `stopServer.log` file contains the following in the last two lines:

Server stop request issued. Waiting for stop status.  
Server server1 stop completed.

The server name varies depending on your settings.

- Use the **stopServer** command to stop an application server from the command line.  
A warning message appears if you are stopping the application server that is running the administrative console application.
- Stop the application server from the command line. In distributed environments, you can use the **stopServer** command to stop a single server. In AIX, use the **stopServer** or the **stopManager** command from the `/usr/WebSphere/AppServer/bin` directory:

```
# ./stopServer.sh server1  
# ./stopManager.sh
```

---

## Creating generic servers

There are two types of generic application servers:

- Non-Java applications or processes.
- Java applications or processes

You can use the wsadmin tool or the **Generic servers** panel of the administrative console to create either type.

**Note:** For the Base WebSphere Application Server product, although you can use the administrative console to create a generic application server definition, you cannot use it to start, stop or, in any way, control or manage that application server. The Base product administrative console can only be used to create server definitions and, if necessary, adjust the server definitions that it creates. To manage Base generic application servers, use the wsadmin tool.

- **Create a non-Java application as a generic server.** The following steps describe how to use the administrative console to create a non-Java application as a generic application server.
  1. Select **Servers > Generic servers**
  2. Click **New**. You can then specify the name of the generic server you are creating.
  3. Type in a name for the generic server. The name must be unique within the application server. It is highly recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular Websphere Application Servers. This will enable you to quickly determine whether to use the **Terminate** or **Stop** button in the administrative console to stop specific application server. You must use the **Terminate** button to stop a generic application server.
  4. Select a template to use in creating the new server. You can use a default application server template for your new server or use an existing application server as a template. The new application server will inherit all properties of the template server. If you create the new server using an existing application server do not enable the option to map applications from the existing server to the new server. This option does not apply for a generic server.
  5. Click **Next**
  6. Click **Finish**. The generic server now appears as an option on the **Generic servers** panel in the administrative console.
  7. On the **Generic servers** panel, click on the name of the generic server.
  8. Under **Additional Properties** click **Process Definition**.
  9. In the **Executable name** field under **General Properties**, enter the name of the non-WebSphere Application Server program that is to be launched when you start this generic server. Executable target type and Executable target properties are not used for non-Java applications. Executable target type and Executable target properties are only used for Java applications
  10. Click **OK**.

- **Create a Java application as a generic server:** The following steps describe how to use the administrative console to create a Java application as a generic application server.
  1. Select **Servers > Generic servers**
  2. Click **New**. You can then specify the name of the generic server you are creating.
  3. Type in a name for the generic server. The name must be unique within the application server. It is highly recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular Websphere Application Servers. This will enable you to quickly determine whether to use the **Terminate** or **Stop** button in the administrative console to stop specific application server. You must use the **Terminate** button to stop a generic application server.
  4. Click **Next**
  5. Click **Finish**. The generic server now appears as an option on the **Applications Server** panel in the administrative console.
  6. Click **Finish**. The generic server now appears as an option on the **Generic servers** panel in the administrative console.
  7. On the **Generic servers** panel, click on the name of the generic server.
  8. Under **Additional Properties** click **Process Definition**.
  9. In the **Executable name** field under **General Properties**, enter the path for Websphere Application Server's default JVM (`{JAVA_HOME}/bin/java`), which will be used to run the Java application when you start this generic server.
  10. In the **Executable target type** field under **General Properties**, select whether a Java class name, **JAVA\_CLASS**, or the name of an executable JAR file, **EXECUTABLE\_JAR**, will be used as the executable target of this Java process. The default for Websphere Application Server is **JAVA\_CLASS**.
  11. In the **Executable target** field under **General Properties**, enter the name of the executable target. (Depending on the executable target type, this will be either a Java class containing a `main()` method, or the name of an executable JAR file.) The default for Websphere Application Server is **com.ibm.ws.runtime.WsServer**.
  12. Click **OK**.

**Note:** If the generic server is to run an application server other than the WebSphere Application Server, leave the **Executable name** field set to the default value and specify the Java class containing the main function for your application serve in the **Executable target** field.

You can now start and terminate the generic server whenever you want to start or terminate the non-WebSphere Application Server server or process associated with this server.

## Starting and terminating generic servers

This topic describes how to start and terminate generic servers.

If you created a generic server on a Base WebSphere Application Server, you cannot start, terminate, or monitor this server with the Base Application Server administrative console. You must use the `wsadmin` tool to manage Base generic servers.

### Starting generic servers

There are two ways to start a generic server in a Network Deployment environment:

- Use the administrative console:
  1. From the administrative console navigation tree, select **Servers > Application Servers**.
  2. Select the check box beside the name of the generic server, and then click **Start**.
  3. View the **Status** value and any messages or logs to see whether the generic server starts.
- Use the MBean `NodeAgent launchProcess` operation of the `wsadmin` tool.

## Terminating generic servers

There are two ways to terminate a generic server in a Network Deployment environment:

- Use the administrative console:
  1. From the administrative console navigation tree, select **Servers > Application Servers**.
  2. Select the check box beside the name of the generic server, and then click **Terminate**.
  3. View the **Status** value and any messages or logs to see whether the generic server terminates.

**Note:** The **Stop** and **Stop Immediate** buttons on the administrative console do not work for generic servers.

- Use the MBean terminate launchProcess operation of the wsadmin tool.

---

## Configuring transport chains

You need to configure transport chains to provide networking services to such functions as the service integration bus component of IBM service integration technologies, WebSphere Secure Caching Proxy, and the high availability manager core group bridge service.

A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP or HTTP. Network ports can be shared among all of the channels within a chain. The channel framework function automatically distributes a request arriving on that port to the correct I/O protocol channel for processing. To define these channels:

1. Create a transport chain: You can either use the administrative console or wsadmin commands to create a transport chain. If you want to use the administrative console:
  - a. Ensure that a port is available for the new transport chain.
  - b. In the administrative console, click **Servers > Application servers > server\_name**, and then click on one of the following:
    - Under **Web container settings**, click **Web container transport chains**.
    - Under **Server messaging**, click either **Messaging engine inbound transports** or **WebSphere MQ link inbound transports**.
  - c. Click **New**. The Create New Transport Chain wizard initializes. During the transport chain creation process, you are asked to:
    - Specify a name for the new chain.
    - Select a transport chain template
    - Select a port, if one is available to which the new transport chain will be bound. If a port is not available or you want to define a new port, specify a port name, the host name or IP address for that port, and a valid port number.

When you click **Finish**, the new transport chain is added to the list of defined transport chains on the **Transport chain** panel.

2. Click on the transport chain's name to view the configuration settings that are in effect for the transport channels contained in this chain. To change any of these settings:
  - a. Click on the channel that requires changes to its settings.
  - b. Make your changes to the configuration settings. Some of the settings, such as the port number are determined by what is specified for the transport chain when it is created and cannot be changed.
  - c. Click on **Custom properties** to set any custom properties that have been defined for your system.
3. When you have made all of your changes, click **OK**.
4. Stop the application server and start it again. You must stop the application server and start it again before the configuration changes you made take affect.

## Transport chains

Transport chains represent a network protocol stack that is used for I/O operations within an application server environment. Transport chains are part of the channel framework function that provides a common networking service for all components, including the service integration bus component of IBM service integration technologies, WebSphere Secure Caching Proxy, and the high availability manager core group bridge service.

A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP, DCS or HTTP. Network ports can be shared among all of the channels within a chain. The channel framework function automatically distributes a request arriving on that port to the correct I/O protocol channel for processing.

The transport chain configuration settings determine which I/O protocols are supported for that chain. Following are some of the more common types of channels. Custom channels that support requirements unique to a particular customer or environment can also be added to a transport chain.

### TCP channel

Used to provide client applications with persistent connections within a Local Area Network (LAN). When configuring a TCP channel, you can specify a list of IP addresses that are allowed to make inbound connections and a list of IP addresses that are not allowed to make inbound connections. You can also specify the thread pool that this channel uses, which allows you to segregate work by the port that the application server is listening on.

### HTTP channel

Used to enable communication with remote servers. It implements the HTTP 1.0 and 1.1 standards and is used by other channels, such as the Web container channel, to server HTTP requests and to send HTTP specific information to servlets expecting this type of information.

### HTTP Tunnel channel

Used to provide client applications with persistent HTTP connections to remote hosts that are either blocked by firewalls or require an HTTP proxy server (including authentication) or both. An HTTP Tunnel channel enables the exchange of application data in the body of an HTTP request or response that is sent to or received from a remote server. An HTTP Tunnel channel also enables client-side applications to poll the remote host and to use HTTP requests to either send data from the client or to receive data from an application server. In either case, neither the client nor the application server is aware that HTTP is being used to exchange the data.

### Web container channel

Used to create a bridge in the transport chain between an HTTP inbound channel and a servlet and JavaServer Pages (JSP) engine.

### DCS channel

Used by the core group bridge service, the data replication service (DRS), and the high availability manager to transfer data, objects, or events among application servers.

### MQ channel

Used in combination with other channels, such as a TCP channel, within the confines of WebSphere MQ support to facilitate communications between a WebSphere System Integration Bus and a WebSphere MQ client or queue manager.

### JFAP channel

Used by the Java Message Service (JMS) server to create connections to JMS resources on a service integration bus.

### SSL channel

Used to associate an SSL configuration repertoire with the transport chain. This channel is only available when Secure Sockets Layer (SSL) support is enabled for the transport chain. An SSL configuration repertoire is defined in the administrative console, under security, on the **SSL configuration repertoires > SSL configuration repertoires** page.

## HTTP transport channel custom property

If you are using an HTTP transport channel, you can add the following custom property to the configuration settings for that HTTP transport channel.

To add a custom property:

1. In the administrative console, click **Application servers** > *server\_name* **Web container settings** > **Web container transport chains** > *chain\_name* > **HTTP Inbound Channel** > **Custom Properties** > **New**
2. Under **General Properties** specify the name of the custom property in the Name field and a value for this property in the Value field. You can also specify a description of this property in the Description field.
3. Click **Apply** or **OK**.
4. Click **Save** to save your configuration changes.
5. Restart the server.

Following is a list of custom properties provided with the application server. These properties are not shown on the settings page for an HTTP transport channel.

### **inProcessLogFilenamePrefix**

Use to specify a prefix for the filename of the network log file. Normally, when inprocess optimization is enabled, requests through the inprocess path are logged based on the logging attributes set up for the Web container's network channel chain. You can use this property to add a prefix to the filename of the network log file. This new filename is then used as the filename for the log file for inprocess requests. Requests sent through the inprocess path are logged to this file instead of to the network log file. For example, if the log file for a network transport chain is named `.../httpaccess.log`, and this property is set to `local` for the HTTP channel in that chain, the filename of the log file for inprocess requests to the host associated with that chain is `.../localhttpaccess.log`.

**Data type** String

## HTTP Tunnel transport channel custom property

If you are using an HTTP Tunnel transport channel, you can add the following custom property to the configuration settings for that HTTP Tunnel transport channel.

To add a custom property:

1. In the administrative console, click **Servers** > **Application servers** > *server\_name* > **Ports**. Click on **View associated transports** for the HTTP Tunnel port to whose configuration settings you want to add this custom property.
2. Click **New**.
3. Under **General Properties** specify the name of the custom property in the Name field and a value for this property in the Value field. You can also specify a description of this property in the Description field.
4. Click **Apply** or **OK**.
5. Click **Save** to save your configuration changes.
6. Restart the server.

Following is a description of the custom property that is provided with the application server. This property is not shown on the settings page for an HTTP Tunnel transport channel.

### **pluginConfigurable**

Indicates whether or not the configuration settings for the HTTP Tunnel transport channel are



included in the plugin-cfg.xml file for the Web server associated with the application server that is using this channel. Configuration settings for each of the Web container transport channels defined for an application server are automatically included in the plugin-cfg.xml file for the Web server associated with that application server.

<b>Data type</b>	Boolean
<b>Default</b>	False

## Troubleshooting transport chain problems

### TCP transport channel fails to bind to a specific host/port combination

If a TCP transport channel fails to bind to a specific port, one of the following situations might have occurred:

- You are trying to bind the channel to a port that is already bound to another application, such as another instance of a WebSphere Application Server.
- You are trying to bind to a port that is in a transitional state waiting for closure. This socket must transition to closed before you restart the server. The port might be in TIME\_WAIT, FIN\_WAIT\_2, or CLOSE\_WAIT state. Issue the netstat -a command from a command prompt window to display the state of the port to which you are trying to bind. If you need to change the amount of elapse time that must occur before TCP/IP can release a closed connection and reuse its resources, see the *Troubleshooting and support PDF*.

## Configuring HTTP transports

**Important:** On a distributed platform, HTTP transport support is deprecated. Therefore, the administrative console page used to configure an HTTP transport is not available unless you migrated an HTTP transport from your V5 environment. You must define an HTTP transport channel instead of an HTTP transport to handle your HTTP requests.

An HTTP transport is the request queue between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application reside. To define the characteristics of the connections between that plug-in and the Web container, you must specify:

- How the transport is to handle a set of connections. For example, you must specify the number of concurrent requests that is to be allowed.
- Whether to secure the connections with SSL.
- The Host and IP information for the transport participants.

1. Change the configuration for an existing HTTP transport.
  - a. Ensure that virtual host aliases include port values for the transport you are changing.
  - b. Go to the HTTP Transports page and click on the transport under **Host** whose configuration you want to change.

Remember, on a distributed platform, HTTP transport support is deprecated. Therefore you cannot view this administrative console page unless you are a V5.x user who, during the V6 migration process, indicated that you want to continue using the HTTP transports that are defined for your V5 environment.

- c. On the settings page for an HTTP transport, which might have the page title DefaultSSLSettings, change the specified values as needed, then click **OK**.
  - d. Custom properties page, add and set any custom properties you want to use.
2. Stop the WebSphere Application Server and start it again. You must stop the WebSphere Application Server and start it again before the configuration changes you made take affect.

If the Web server is located on a machine remote from the Application Server, copy the `plugin-cfg.xml` file to the remote Web server and replace the file that is there. See the *Installing your application serving environment* PDF for information about copying the `plugin-cfg.xml` and binary plug-in module to a remote Web server and configuring the Web server to use the files.

## HTTP transport collection

Use this page to view or manage HTTP transports. Transports provide request queues between WebSphere Application Server plug-ins for Web servers and Web containers in which the Web modules of applications reside. When you request an application in a Web browser, the request is passed to the Web server, then along the transport to the Web container.

On a distributed platform, if, when migrating from WebSphere Application Server Version 5.x, you indicate that you want to continue using an HTTP transport to handle your HTTP requests, your Version 5.x transports are migrated for you. If you are not migrating from Version 5.x, you must set up an HTTP transport channel to handle your HTTP requests.

To view the HTTP Transport administrative console page, click **Servers > Application Servers > *server\_name* > Web Container Settings > Web Container > HTTP Transports.**

the HTTP Transport panel on the administrative console

### **Host:**

Specifies the host IP address to bind for transport. If the application server is on a local machine, the host name might be `localhost`.

### **Port:**

Specifies the port to bind for transport. The port number can be any port that currently is not in use on the system. The port number must be unique for each application server instance on a given machine.

For distributed platforms, there is no limit to the number of HTTP ports that are allowed per process.

### **SSL Enabled:**

Specifies whether to protect connections between the WebSphere plug-in and application server with Secure Sockets Layer (SSL). The default is not to use SSL.

## HTTP transport settings

Use this page to view and configure an HTTP transport. The name of the page might be that of an SSL setting such as `DefaultSSLSettings`.

On a distributed platform, if, when migrating from Version 5, you indicate that you want to continue using an HTTP transport to handle your HTTP requests, your Version 5 transports are migrated for you. If you are not migrating from WebSphere Application Server Version 5.x, you must set up an HTTP transport channel to handle your HTTP requests.

To view the HTTP Transport panel on the administrative console, click **Servers > Application Servers > *server\_name* > Web Container Settings > Web Container > HTTP Transports > *host\_name*.**

### **Host:**

Specifies the host IP address to bind for transport.



If the application server is on a local machine, the host name might be localhost.

**Data type** String

### **Port:**

Specifies the port to bind for transport. Specify a port number between 1 and 65535. The port number must be unique for each application server on a given machine.

**Data type** Integer  
**Range** 1 to 65535

### **SSL Enabled:**

Specifies whether to protect connections between the WebSphere Application Server plug-in and application server with Secure Sockets Layer (SSL). The default is not to use SSL.

**Data type** Boolean  
**Default** false

### **SSL:**

Specifies the Secure Sockets Layer (SSL) settings type for connections between the WebSphere Application Server plug-in and application server. The options include one or more SSL settings defined in the Security Center; for example, DefaultSSLSettings, ORBSSLSettings, or LDAPSSLSettings.

**Data type** String  
**Default** An SSL setting defined in the Security Center

## **Transports**

A transport is the request queue between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application reside. When a user at a Web browser requests an application, the request is passed to the Web server, then along the transport to the Web container.

Transports define the characteristics of the connections between a Web server and an application server, across which requests for applications are routed. Specifically, they define the connection between the Web server plug-in and the Web container of the application server.

Administering transports is closely related to administering WebSphere Application Server plug-ins for Web servers. Indeed, without a plug-in configuration, a transport configuration is of little use.

On a distributed platform, when migrating from WebSphere Application Server Version 5.x, you indicate that you want to continue using an HTTP transport to handle your HTTP requests, your Version 5.x transports are migrated for you. If you are not migrating from Version 5.x, you must set up an HTTP transport channel to handle your HTTP requests.

### **The internal transport**

The internal HTTP transport allows HTTP requests to be routed to the application server directly through a Web server plug-in. Logging is provided for debug purposes.

Prior to WebSphere Application Server Version 5.0.2, the HTTP transport functionality existed only as a means of accepting HTTP requests forwarded by an HTTP plug-in that was connected to a Web server. In

WebSphere Application Server Version 5.0.2, HTTP transport functionality is now a supported internal Web server. By default, the internal HTTP transport listens for HTTP requests on port 9080 and for HTTPS requests on port 9443.

For example, use the URL `http://localhost:9080/snoop` to send requests to the snoop servlet on the local machine over HTTP and `https://localhost:9443/snoop` to send requests to the snoop servlet on the local machine over HTTPS.

The transport configuration is a part of the Web container configuration. You can configure the internal transport to use ports other than 9080 and 9443. However, you must also adjust your virtual host alias and what you type into the Web browser.

## HTTP transport custom properties

Use this page to set custom properties for an HTTP transport.

On a distributed platform, HTTP transport support is deprecated. Therefore you cannot create a new HTTP transport. Instead you must create an HTTP transport channel to handle your HTTP requests. If you are a WebSphere Application Server Version 5.x user who has migrated to Version 6, and during the migration process you indicated that you want to continue using an HTTP transport to handle your HTTP requests, your Version 5.x transports are still available for your use.

If you are using HTTP transports, you can set the following custom properties on either the Web Container or HTTP Transport **Custom Properties** panel on the administrative console. When set on the Web container **Custom Properties** page, all transports inherit the properties. Setting the same properties on a transport overrides like settings defined for a Web container.

To specify values for these custom properties for a specific transport on the HTTP Transport **Custom Properties** page:

1. In the console navigation tree, click **Servers > Application Servers > server\_name > Web Container settings > Web Container > HTTP Transport**

To specify a custom property:

1. Click on the **HOST** whose properties you want to set.
2. Under **Additional Properties** select **Custom Properties**.
3. On the Custom Properties page, click **New**.
4. On the settings page, enter the property you want to configure in the **Name** field and the value you want to set it to in the **Value** field.
5. Click **Apply** or **OK**.
6. Click **Save** on the console task bar to save your configuration changes.
7. Restart the server.

Following is a list of custom properties provided with the Application Server. These properties are not shown on the settings page for an HTTP transport.

### **ConnectionIOTimeout:**

Use the `ConnectionIOTimeout` property to specify the maximum number of seconds to wait when trying to read or process data during a request.

<b>Data type</b>	Integer
<b>Default</b>	For distributed platforms: 5 seconds

### **ConnectionKeepAliveTimeout:**

Use the `ConnectionKeepAliveTimeout` property to specify the maximum number of seconds to wait for the next request on a keep alive connection.

**Data type** Integer

### ***MaxConnectBacklog:***

Use the `MaxConnectBacklog` property to specify the maximum number of outstanding connect requests that the operating system will buffer while it waits for the application server to accept the connections. If a client attempts to connect when this operating system buffer is full, the connect request will be rejected.

Set this value to the number of concurrent connections that you would like to allow. Keep in mind that a single client browser might need to open multiple concurrent connections (perhaps 4 or 5); however, also keep in mind that increasing this value consumes more kernel resources. The value of this property is specific to each transport.

**Data type** Integer  
**Default** 511

### ***MaxKeepAliveRequests:***

Use the `MaxKeepAliveRequests` property to specify the maximum number of requests which can be processed on a single keep alive connection. This parameter can help prevent denial of service attacks when a client tries to hold on to a keep-alive connection. The Web server plug-in keeps connections open to the application server as long as it can, providing optimum performance.

**Data type** Integer  
**Default** For distributed platforms: 100 requests

### ***KeepAliveEnabled:***

This property is only valid in a distributed environment. Use the `KeepAliveEnabled` property to specify whether or not to keep connections alive

**Data type** String  
**Default** true

### ***Trusted:***

This property is only valid in a distributed environment. Use the `Trusted` property to indicate that the application server can use the private headers that the Web server plug-in adds to requests.

**Data type** String  
**Default** false

## **Configuring error logging for internal Web server HTTP transport**

To debug potential problems with using the HTTP transport as an internal Web server, you can use the following error logging capabilities.

1. Turn error logging on. To turn error logging on, add the following custom property to the HTTP Transport's configuration settings, and set the value to **false**:

Property name: `ErrorLogDisable`  
Value: True/False  
Default: Error log is disabled by default

When you are ready to turn error logging off, set the value of the **ErrorLogDisable** property back to **true**.

2. To specify your own error log file, add the following property to the transport section of the server.xml file:

Property name: ErrorLog  
Value: <filename>  
Default: logs/<server instance>/http.log

The error log property is used to specify where to place the error log. For example:<properties xmi:id="WebContainer\_Property\_6" name="ErrorLog" value="logs/<server instance>/http.log"/>

**Note:** The error log should appear in each instance of the server.

If you are going to be using error logging for multiple HTTP transports in a single HTTP server, make sure you specify a unique filename for the error log file associated with each HTTP transport.

3. Add the LogLevel property to the transport section of the server.xml file to specify the level of messages to log in the error log file.

Property name: LogLevel  
Value: <level> (Levels include: debug, info, warn, error, crit)  
Default: warn (warn includes error and crit; debug includes all levels)  
Scope: Virtual/Global

Log levels specify the type of message that appears in the error log. The *warn*, *error*, and *crit* messages are logged by default.

4. Restart the server.

If you have enabled error logging and encounter an error, there should be an error log message in the error log file you specified.

## Configuring access logging for internal Web server HTTP transport

To debug potential problems with using the HTTP transport as an internal Web server, you can use the following access logging capabilities.

1. Turn access logging on. To turn access logging on, add the following custom property to the HTTP Transport's configuration settings, and set the value to **false**:

Property name: AccessLogDisable  
Values: True/False  
Default: Access log is disabled by default

When you are ready to turn access logging off, set the value of the **AccessLogDisable** property back to **true**.

2. To specify your own access log file, add the following property to the transport section of the server.xml file:

Property name: AccessLog  
Value: <filename>  
Default Value: logs/<server instance>/http\_access.log

The default access log file is logs/<server\_instance>/http\_access.log. Access log entries should have the format:

<hostname or IP> <user agent> [<local time> -<status code>] <thread id> <http request> <status code> <bytecount>

**Note:** If you are going to be using access logging for multiple HTTP transports in a single HTTP server, make sure you specify a unique filename for the access log file associated with each HTTP transport.

3. Restart the server.

If you have enabled access logging, there will be an access log in the location you specified.

## Transport chains collection

Use this page to view or manage transport chains. Transport chains enable communication through transports, or protocol stacks, which are usually socket based.

A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP or HTTP. Network ports can be shared among all of the channels within a chain. The Channel Framework function automatically distributes a request arriving on that port to the correct I/O protocol channel for processing.

The **Transport chains** page lists the transport chains defined for the selected application server. Transport chains represent network protocol stacks operating within this application server.

To view this administrative console page, click **Servers > Application servers > *server\_name* > Ports**. Click on **View associated transports** for the port whose transport chains you want to view.

### Name

Specifies a unique identifier for the transport chain. For WebSphere Application Server, transport name must be unique within a WebSphere Application Server configuration. Click on the name of a transport chain to change its configuration settings.

### Enabled

When set to true, the transport chain is activated at application server startup.

### Host

Specifies the host IP address to bind for transport. If the application server is on a local machine, the host name might be localhost.

### Port

Specifies the port to bind for transport. The port number can be any port that currently is not in use on the system, might be localhost or the wildcard character \* (an asterisk). The port number must be unique for each application server instance on a given machine

### SSL Enabled

When set to true, users are notified if there is a channel that enables Secure Sockets Layer (SSL) in the listed chain. When SSL is enabled, all traffic going through this transport is encrypted and digitally secured.

## Transport chain settings

This page lists the types of transport channels configured for the selected transport chain. A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP, HTTP, or DCS.

To view this administrative console page, click **Servers > Application servers > *server\_name* > Ports**. Click on **View associated transports** for the port whose transport chains you want view and then click on the name of a specific chain.

### Name

Specifies the name of the selected transport chain.

You can edit this field to rename this transport chain. However, remember that the name must be unique within a WebSphere Application Server configuration.

### Enabled

When checked, this transport chain is activated at application server startup.

## Transport channels

Lists the transport channels configured for this transport chain and their configuration settings. To change a transport channel's configuration settings, click on the name of that transport channel.

## HTTP tunnel transport channel settings

Use this page to view and configure an HTTP tunnel transport channels. Inbound connections sent through this channel are tunneled over HTTP, allowing intermediates to view this data as the body of an HTTP message instead of in its natural format. This type of channel is often used to circumvent firewalls with protocol restrictions.

To view this administrative console page, click **Servers > Application servers > *server\_name* > Ports** . Click on **View associated transports** for the port associated with the HTTP Tunnel transport channel whose settings you want to look at.

### Transport channel name

Specifies the name of the HTTP tunnel transport channel.

This name must be unique across all channels in a WebSphere Application Server environment. For example DCS and TCP transport channels cannot have the same name if they reside within the same system.

### Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

<b>Data type</b>	Positive integer
<b>Default</b>	10

## HTTP transport channel settings

Use this page to view and configure an HTTP transport channel. This type of transport channel handles HTTP requests from a remote client.

An HTTP transport channel parses HTTP requests and then finds an appropriate application channel to handle the request and send a response.

To view this administrative console page, click **Servers > Application servers > *server\_name* > Ports** . Click on **View associated transports** for the port associated with the HTTP transport channel whose settings you want to look at.

### Transport channel name

Specifies the name of the HTTP transport channel.

This name must be unique across all channels in a WebSphere Application Server environment. For example, an HTTP transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

### Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

<b>Data type</b>	Positive integer
<b>Default</b>	10

### Maximum persistent requests

Specifies the maximum number of persistent (keep-alive) requests that are allowed on a single HTTP connection. If a value of 0 (zero) is specified, only one request is allowed per connection. If a value of -1 is specified, an unlimited number of requests is allowed per connection.

<b>Data type</b>	Integer
<b>Default</b>	100

### Use Keep-Alive

When selected, the HTTP transport channel, when sending an outgoing HTTP message, uses a persistent connection (keep-alive connection) instead of a connection that closes after one request or response exchange occurs.

**Note:** If a value other than 0 is specified for the maximum persistent requests property, the Use Keep-Alive property setting is ignored.

The default for this property is selected.

### Read timeout

Specifies the amount of time, in seconds, the HTTP transport channel waits for a read request to complete on a socket after the first read request occurs. The read being waited for could be an HTTP body (such as a POST) or part of the headers if they were not all read as part of the first read request on the socket.

<b>Data type</b>	Integer
<b>Default</b>	60 seconds

### Write timeout

Specifies the amount of time, in seconds, that the HTTP transport channel waits on a socket for each portion of response data to be transmitted. This timeout usually only occurs in situations where the writes are lagging behind new requests. This can occur when a client has a low data rate or the server's network interface card (NIC) is saturated with I/O.

<b>Data type</b>	Integer
<b>Default</b>	60 seconds

### Persistent timeout

Specifies the amount of time, in seconds, that the HTTP transport channel allows a socket to remain idle between requests.

<b>Data type</b>	Integer
<b>Default</b>	30 seconds

### Enable NCSA access logging

When selected, the HTTP transport channel performs NCSA access and error logging. Enabling NCSA access and error logging slows server performance.

To configure NCSA access and error logging, click **HTTP error and NCSA access logging** under **Related Items**. Even if HTTP error and NCSA access logging is configured, it is not enabled unless the Enable NCSA access logging property is selected.

The default value for the Enable NCSA access logging property is not selected.

## TCP transport channel settings

Use this page to view and configure an TCP transport channels. This type of transport channel handles inbound TCP/IP requests from a remote client.

To view this administrative console page, click **Servers > Application servers > *server\_name* > Ports > .** Click on **View associated transports** for the port associated with the TCP transport channel whose settings you want to view.

### Transport channel name

Specifies the name of the TCP transport channel.

This name must be unique across all channels in a WebSphere Application Server environment. For example, a TCP transport channel and an HTTP transport channel cannot have the same name if they reside within the same system.

### Port

Specifies the TCP/IP port this transport channel uses to establish connections between a client and an application server. The TCP transport channel binds to the hostnames and ports listed for the Port property. You can specify the wildcard \* (an asterisk), for the hostname if you want this channel to listen to all hosts that are available on this system. However, before specifying the wildcard value, make sure this TCP transport channel does not have to bind to a specific hostname.

### Thread pool

Select from the drop-down list of available thread pools the thread pool you want the TCP transport channel to use when dispatching work.

### Maximum open connections

Specifies the maximum number of connections that can be open at one time.

<b>Data type</b>	Integer between 1 and 20,000 inclusive
<b>Default</b>	20,000

### Inactivity timeout

Specifies the amount of time, in seconds, that the TCP transport channel waits for a read or write request to complete on a socket.

**Note:** The value specified for this property might be overridden by the wait times established for channels above this channel. For example, the wait time established for an HTTP transport channel overrides the value specified for this property for every operation except the initial read on a new socket.

<b>Data type</b>	Integer
<b>Default</b>	60 seconds

### Address exclude list

Lists the IP addresses that are not allowed to make inbound connections. Use a comma to separate the IPv4 or IPv6 or both addresses to which you want to deny access on inbound TCP connection requests.

All four numeric values in an IPv4 address must be represented by a number or the wildcard character \* (an asterisk).

Following are examples of valid IPv4 addresses that can be included in an **Address exclude list**:



```
*.1.255.0
254.*.*.9
1.*.*.*
```

All eight numeric values of an IPv6 address must be represented by a number or the wildcard character \* (an asterisk). No shortened version of the IPv6 address should be used. Even though a shortened version is processed with no error given, it does not function correctly in this list. Each numeric entry should be a 1- 4 digit hexadecimal number.

Following are examples of valid IPv6 addresses that can be included in an **Address exclude list**:

```
0::*:0:007F:0:0001:0001
F:FF:FFF:FFFF:1:01:001:0001
1234::*:4321::*:9F9f::*:0000
```

**Note:** The **Address include list** and **Host name include list** are processed before the **Address exclude list** and the **Host name exclude list**. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it will not be allowed access.

### Address include list

Lists the IP addresses that are allowed to make inbound connections. Use a comma to separate the IPv4 or IPv6 or both addresses to which you want to grant access on inbound TCP connection requests.

All four numeric values in an IPv4 address must be represented by a number or the wildcard character \* (an asterisk).

Following are examples of valid IP addresses that can be included in an **Address include list**:

```
*.1.255.0
254.*.*.9
1.*.*.*
```

All eight numeric values of an IPv6 address must be represented by a number or the wildcard character \* (an asterisk). No shortened version of the IPv6 address should be used. Even though a shortened version is processed with no error given, it does not function correctly in this list. Each numeric entry should be a 1- 4 digit hexadecimal number.

Following are examples of valid IPv6 addresses that can be included in an **Address include list**:

```
0::*:0:007F:0:0001:0001
F:FF:FFF:FFFF:1:01:001:0001
1234::*:4321::*:9F9f::*:0000
```

**Note:** The **Address include list** and **Host name include list** are processed before the **Address exclude list** and the **Host name exclude list**. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it will not be allowed access.

### Host name exclude list

List the host names that are not allowed to make connections. Use a comma to separate the URL addresses to which you want to deny access on inbound TCP connection requests.

A URL address can start with the wildcard character \* (an asterisk) followed by a period; for example, \*.Rest.Of.Address. If a period does not follow the wildcard character, the asterisk will be treated as a normal non-wildcard character. The wildcard character can not appear any where else in the address. For example, ibm\*.com is not a valid hostname.

Following are examples of valid URL addresses that can be included in a **Host name exclude list**:

```
*.ibm.com  
www.ibm.com  
*.com
```

**Note:** The **Address include list** and **Host name include list** are processed before the **Address exclude list** and the **Host name exclude list**. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it is not allowed access.

### Host name include list

Lists the host names that are allowed to make inbound connections. Use a comma to separate the URL addresses to which you want to grant access on inbound TCP connection requests.

A URL address can start with the wildcard character \* (an asterisk) followed by a period; for example, \*.Rest.Of.Address. If a period does not follow the wildcard character, the asterisk will be treated as a normal non-wildcard character. The wildcard character can not appear any where else in the address. For example, ibm\*.com is not a valid hostname.

Following are examples of valid URL addresses that can be included in a **Host name include list**:

```
*.ibm.com  
www.ibm.com  
*.com
```

**Note:** The **Address include list** and **Host name include list** are processed before the **Address exclude list** and the **Host name exclude list**. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it is not allowed access.

## DCS transport channel settings

Use this page to view and configure an DCS transport channels. This type of transport channel handles inbound Distribution and Consistency Services (DCS) messages.

By default, two channel transport chains are defined for an application server that contains a DCS channel:

- The chain named DCS contains a TCP and a DCS channel.
- The chain named DCS-Secure contains a TCP, an SSL, and a DCS channel.

Both of these chains terminate in, or use the same TCP channel instance. This TCP channel is associated with the DCS\_UNICAST\_ADDRESS port and is not used in any other transport chains. One instance of an SSL channel is reserved for use in the DCS-Secure chain. It also is not used in any other transport chains.

To view this administrative console page, click **Servers > Application servers > server\_name > Ports > .** Click on **View associated transports** for the port associated with the DCS transport channel whose settings you want to look at.

### Transport channel name

Specifies the name of the DCS transport channel.

This name must be unique across all channels in a WebSphere Application Server environment. For example DCS and TCP transport channels cannot have the same name if they reside within the same system.

### Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

The discrimination weight of the DCS channel in a DCS-Secure transport chain should always be less than the discrimination weight of the SSL channel that is in that chain. Other SSL channels in other chains might have different discrimination values.

<b>Data type</b>	Positive integer	
<b>Default</b>	1 for the DCS channel	2 for the SSL channel

## Web container transport channel settings

Use this page to view and configure a Web container inbound channel transport. This type of channel transport handles inbound Web container requests from a remote client.

To view this administrative console page, click **Servers > Application servers > *server\_instance* > Web container settings > Web container transport chains > *transport\_chain* > Web Container Inbound Channel** .

### Transport Channel Name

This name must be unique across all channels in a WebSphere Application Server environment. This means that TCP transport channels and HTTP transport channels cannot have the same name if they reside within the same system.

### Discrimination weight

Specifies the priority that this transport chain has in relation to other transport chains if this transport channel is shared amongst several transport chains.

### Write buffer size

Specifies the amount of content in bytes to buffer unless the servlet explicitly calls flush/close on the response/writer output stream.

<b>Data type</b>	bytes
<b>Default</b>	9192 bytes

---

## Custom services

A custom service provides the ability to plug into a WebSphere Application Server application server to define a hook point that runs when the server starts and shuts down.

A developer implements a custom service containing a class that implements a particular interface. The administrator configures the custom service in the administrative console, identifying the class created by the developer. When an application server starts, any custom services defined for the application server are loaded and the server runtime calls their initialize methods.

---

## Developing custom services

The following restrictions apply to the WebSphere Application Server custom services implementation:

- The init and shutdown methods must return control to the runtime.
- No work is dispatched into the server instance until all custom service initialize methods return.
- The init and shutdown methods are called only once on each service, and once for each operating system process that makes up the server instance. File I/O is supported.
- Initialization of process level static data, without leaving the process, is supported.
- Only JDBC RMLT (resource manager local transaction) operations are supported. Every unit of work (UOW) must be completed before the methods return.
- Creation of threads is not supported.
- Creation of sockets and I/O, other than file I/O, is not supported. Running standard J2EE code (client code, servlets, enterprise beans) is not supported.
- The JTA interface is not available. This feature is available in J2EE server processes and distributed generic server processes only.
- While the runtime makes an effort to call shutdown, there is no guarantee that shutdown will be called prior to process termination.

Note that these restrictions apply to the shutdown and init methods equally. Some JNDI operations are available.

Develop a custom service class that implements the `com.ibm.websphere.runtime.CustomService` interface. The properties passed by the application server runtime to the initialize method can include one for an external file containing configuration information for the service (retrieved with `externalConfigURLKey`). In addition, the properties can contain any name-value pairs that are stored for the service, along with the other system administration configuration data for the service. The properties are passed to the initialize method of the service as a `Properties` object.

There is a shutdown method for the interface as well. Both methods of the interface declare that they may create an exception, although no specific exception subclass is defined. If an exception is created, the runtime logs it, disables the custom service, and proceeds with starting the server.

As mentioned above, your custom services class must implement the `CustomService` interface. In addition, your class must implement the initialize and shutdown methods. Suppose the name of the class that implements your custom service is `ServerInit`, your code would declare this class as shown below. The code below assumes that your custom services class needs a configuration file. It shows how to process the input parameter in order to get the configuration file. If your class does not require a configuration file, the code that processes `configProperties` is not needed.

```
public class ServerInit implements CustomService
{
/**
 * The initialize method is called by the application server run-time when the
 * server starts. The Properties object passed to this method must contain all
 * configuration information necessary for this service to initialize properly.
 *
 * @param configProperties java.util.Properties
 */
    static final java.lang.String externalConfigURLKey =
        "com.ibm.websphere.runtime.CustomService.externalConfigURLKey";

    static String ConfigFileName="";

    public void initialize(java.util.Properties configProperties) throws Exception
    {
        if (configProperties.getProperty(externalConfigURLKey) != null)
        {
```

```

        ConfigFileName = configProperties.getProperty(externalConfigURLKey);
    }

    // Implement rest of initialize method
}

/**
 * The shutdown method is called by the application server run-time when the
 * server begins its shutdown processing.
 *
 * @param configProperties java.util.Properties
 */
public void shutdown() throws Exception
{
    // Implement shutdown method
}

```

## Custom service collection

Use this page to view a list of services available to the application server and to see whether the services are enabled. A custom service provides the ability to plug into a WebSphere application server and define code that runs when the server starts or shuts down.

To view this administrative console page, click **Servers > Application servers > server\_name**. Then, under Server Infrastructure, click **Administration > Custom Services**.

## External Configuration URL

Specifies the URL for a custom service configuration file.

If your custom services class requires a configuration file, the value provides a fully-qualified path name to that configuration file. This file name is passed into your custom service class.

## Classname

Specifies the class name of the service implementation. This class must implement the Custom Service interface.

## Display Name

Specifies the name of the service.

## Enable service at server startup

Specifies whether the server attempts to start and initialize the service when its containing process (the server) starts. By default, the service is not enabled when its containing process starts.

## Custom service settings

Use this page to configure a service that runs in an application server.

To view this administrative console page, click **Servers > Application servers > server\_name**. Then, under Server Infrastructure, click **Administration > Custom services > custom\_service\_name**.

### ***Enable service at server startup:***

Specifies whether the server attempts to start and initialize the service when its containing process (the server) starts. By default, the service is not enabled when its containing process starts.

<b>Data type</b>	Boolean
<b>Default</b>	false

### ***External Configuration URL:***

Specifies the URL for a custom service configuration file.

If your custom services class requires a configuration file, specify the fully-qualified path name to that configuration file for the value. This file name is passed into your custom service class.

<b>Data type</b>	String
<b>Units</b>	URL

#### ***Classname:***

Specifies the class name of the service implementation. This class must implement the Custom Service interface.

<b>Data type</b>	String
<b>Units</b>	Java class name

#### ***Display Name:***

Specifies the name of the service.

<b>Data type</b>	String
------------------	--------

#### ***Description:***

Describes the custom service.

<b>Data type</b>	String
------------------	--------

#### ***Classpath:***

Specifies the class path used to locate the classes and JAR files for this service.

<b>Data type</b>	String
<b>Units</b>	Class path

---

## **Process definition**

A process definition specifies the run-time characteristics of an application server process.

A process definition can include characteristics such as JVM settings, standard in, error and output paths, and the user ID and password under which a server runs.

---

## **Defining application server processes**

To enhance the operation of an application server, you can define command-line information for starting or initializing an application server process. Such settings define run-time properties such as the program to run, arguments to run the program, and the working directory.

1. Go to the settings page for a process definition in the administrative console. Click **Servers > Application Servers** in the console navigation tree, click on an application server name and then **Java and Process Management > Process Definition**. Note that you can also define application server processes using the wsadmin tool. For more information, see the *Administering applications and their environment* PDF.

2. On the settings page for a process definition, specify the name of the executable to run, any arguments to pass when the process starts running, and the working directory in which the process will run. Then click **OK**.
3. Specify process execution statements for starting or initializing a UNIX process.
4. Specify monitoring policies to track the performance of a process.
5. Specify process logs to which standard out and standard error streams write. Complete this step if you do not want to use the default file names.
6. Specify name-value pairs for properties needed by the process definition.
7. Stop the application server and then restart the server.
8. Check the application server to ensure that the process definition runs and operates as intended.

## Process definition settings

Use this page to view or change settings for a process definition. For WebSphere Application Server, this page provides command-line information for starting or initializing a process.

To view this administrative console page, click **Servers > Application Servers > *server\_name***. Then under Server Infrastructure click **Java Process Management > Process Definition**.

### Executable Name

This command applies to base WebSphere Application Server only. It specifies the executable name that is invoked to start the process.

**Data type** String

### Executable Arguments

This command applies to base WebSphere Application Server only. It specifies the arguments that are passed to the executable when starting the process.

For example, the executable target program might expect three arguments: *arg1 arg2 arg3*.

**Data type** String  
**Units** Java command-line arguments

### Working Directory

Specifies the file system directory that the process uses as its current working directory.

The process uses this directory to determine the locations of input and output files with relative path names.

Passivated enterprise beans are placed in the current working directory of the application server on which the beans are running. Make sure the working directory is a known directory under the root directory of the WebSphere Application Server product.

**Data type** String

## Process execution settings

Use this page to view or change the process execution settings for a server process that applies to either an application server, a node agent or a deployment manager.

To view this administrative console page for an application server, click **Servers > Application Servers > *server\_name***. Then, under Server Infrastructure, click **Java and Process Management > Process Execution**.

To view this administrative console page for a node agent, click **System Administration > Node agents > *node\_agent\_name***. Then, under Server Infrastructure , click **Java and Process Management > Process Definition > Process Execution**.

To view this administrative console page for a deployment manager, click **System Administration > Deployment manager**. Then, under Server Infrastructure , click **Java and Process Management > Process Definition > Process Execution**.

#### ***Process Priority:***

Specifies the operating system priority for the process. The administrative process that launches the server must have root operating system authority in order to honor this setting.

<b>Data type</b>	Integer
<b>Default</b>	20 for WebSphere Application Server on all operating systems.

#### ***UMASK:***

Specifies the user mask under which the process runs (the file-mode permission mask).

<b>Data type</b>	Integer
------------------	---------

#### ***Run As User:***

Specifies the user that the process runs as.

<b>Data type</b>	String
------------------	--------

#### ***Run As Group:***

Specifies the group that the process is a member of and runs as.

On OS/400, the Run As Group setting is ignored.

<b>Data type</b>	String
------------------	--------

#### ***Run In Process Group:***

Specifies a specific process group for the process. This process group is useful for such things as processor partitioning. A system administrator can assign a process group to run on, for example, 6 of 12 processors. The default (0) is not to assign the process to any specific group.

On OS/400, the Run In Process Group setting is ignored.

<b>Data type</b>	Integer
<b>Default</b>	0

## **Process logs settings**

Use this page to view or change settings for specifying the files to which standard out and standard error streams write.

To view this administrative console page, click **Servers > Application Servers > *server\_name***. Then, under Troubleshooting, click **Logging and Tracing > Process Logs**.



### ***Stdout File Name:***

Specifies the file to which the standard output stream is directed. The file name can include a symbolic path name defined in the variable entries.

Use the field on the configuration tab to specify the file name. Use the field on the Runtime tab to select a file for viewing. View the file by clicking **View**.

Direct server output to the administrative console or to the process that launched the server, by either deleting the file name or specifying console on the configuration tab.

<b>Data type</b>	String
<b>Units</b>	File path name

### ***Stderr File Name:***

Specifies the file to which the standard error stream is directed. The file name can include a symbolic path name defined in the variable entries.

Use the field on the configuration tab to specify the file name. Use the field on the runtime tab to select a file for viewing. View the file by clicking **View**.

<b>Data type</b>	String
<b>Units</b>	File path name

## **Monitoring policy settings**

Use this page to view or change settings that control how the node agent monitors and restarts a process.

To view this administrative console page, click **Servers > Application Servers > *server\_name***. Then, under Server Infrastructure, click **Java and Process Management > Process Definition > Monitoring Policy**.

### ***Maximum Startup Attempts:***

Specifies the maximum number of times to attempt to start the application server before giving up.

<b>Data type</b>	Integer
------------------	---------

### ***Ping Interval:***

Specifies the frequency of communication attempts between the parent process, such as the node agent, and the process it has spawned, such as an application server. Adjust this value based on your requirements for restarting failed servers. Decreasing the value detects failures sooner; increasing the value reduces the frequency of pings, reducing system overhead.

<b>Data type</b>	Integer
<b>Range</b>	Set the value greater than or equal to 0 (zero) and less than 2147483.

### ***Ping Timeout:***

When a parent process is spawning a child process, such as when a process manager spawns a server, the parent process pings the child process to see whether the child was spawned successfully. This value specifies the number of seconds that the parent process should wait (after pinging the child process) before assuming that the child process failed.

<b>Data type</b>	Integer
<b>Units</b>	Seconds
<b>Range</b>	Set the value greater than or equal to 0 (zero) and less than 2147483.

### ***Automatic Restart:***

Specifies whether the process should restart automatically if it fails. The default is to restart the process automatically.

<b>Data type</b>	Boolean
<b>Default</b>	true

### ***Node Restart State:***

Specifies the desired state for the process after the node completely shuts down and restarts.

<b>Data type</b>	String
<b>Default</b>	STOPPED
<b>Range</b>	Valid values are STOPPED, RUNNING, or PREVIOUS. If you want the process to return to its current state after the node restarts, use PREVIOUS.

## **Automatically restarting server processes**

There are several server processes related to WebSphere Application Server products that the operating system can monitor and automatically restart when the server processes stop abnormally. This task describes how to set up these *monitored* processes.

To set up this function on a Linux or UNIX-based operating system, you must have root authority to edit the inittab file.

On a Windows operating system, you must belong to the Administrator group and have the following advanced user rights:

- Act as part of the operating system
- Log on as a service

The Installation wizard grants you the user rights if your user ID is part of the administrator group. If you are running on a Microsoft Windows 2000 Operating System, the Installation wizard displays a message that states that although the advanced user rights are now effective, they do not display as effective until the next time you log on to the Windows machine.

You can also add the advanced user rights manually if you are performing a silent installation on a Windows platform. For example, to grant the user rights to your administrator group user ID on a Windows 2000 Server platform, perform the following procedure:

1. Click **Administrative Tools** in the Control Panel.
2. Click **Local Security Policy**.
3. Click **Local Policies**.
4. Click **User Rights Assignments**.

5. Right click **Act as part of the operating system**.
6. Click **Security**.
7. Click **Add**.
8. Click your user ID.
9. Click **Add**.
10. Click **OK**.
11. Click **OK**.
12. Right click **Log on as a service**.
13. Click **Security**.
14. Click **Add**.
15. Click **OK**.
16. Click **OK**.
17. Reboot your machine to make the settings effective.

Consult your Windows help system for more information.

You can use this function to automatically restart base servers. You can restart the *server1* process, for example.

You must manually create a shell script that automatically starts any of the processes previously mentioned, on a Linux and UNIX-based operating system. Each Windows service or UNIX shell script controls a single process, such as a stand-alone WebSphere Application Server instance. Multiple stand-alone Application Server processes require multiple Windows service or UNIX scripts, which you can define.

If you do not install the WebSphere Application Server base product as a Windows service during installation, you can use the do so at a later time. The operating system can then monitor each server process and restart the process if it stops.

1. **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 to select services that the installation wizard can set up:
    - a. Click **Run WebSphere Application Server as a service**.  
 If you select this option, the installation wizard creates the following service during the installation:  
**IBMWAS6Service - node\_name**  
 The **IBMWAS6Service -node\_name** service controls the *node\_name* process.  
 After you complete and verify the installation, use the Windows Services panel to change the **IBMWAS6Service -node\_name** service to an automatic startup type.
      - 1) Right click **IBMWAS6Service -node\_name** and click **Properties**.
      - 2) Click **Automatic** from the **Startup type** list box and click **OK**.
    - b. Click **Run IBM HTTP Server as a service**.  
 Select this option on the machine where you are installing the IBM HTTP Server.  
 If you select this option, the installation wizard creates the following services during the installation:
      - **IBM HTTP Server 2.0.x**
      - **IBM HTTP Administration 2.0.x**
 The installation wizard defines the startup type of these services as **automatic**. It is not necessary for you to change the type from manual to automatic.
    - c. Enter your user ID and password and click **Next**.

In a coexistence environment, you can change the default service names to make them unique. In a same version coexistence scenario for IBM HTTP Server 2.0.x on a Windows platform, you cannot use the default service names created by the installer because they are common.

To work around this problem:

- a. Install the first copy of IBM HTTP Server, either by itself or with WebSphere Application Server and select to install the services.
- b. Customize the service names for the first install by running the following commands from the first install location:

```
apache -k install -n "IHS 2.0(1)"
apache -k install -f conf\admin.conf -n "IHS 2.0 Administration (1)"
```

- c. Edit the AdminAlias directive in the *installLocation 1\conf\admin.conf* file to point to the new service name, such as **IHS 2.0(1)**.
- d. Remove the default service names installed by the first install by running the following commands:

```
apache -k uninstall -n "IBM HTTP Server 2.0"
apache -k uninstall -n "IBM HTTP Administration 2.0"
```

- e. Install the second copy of IBM HTTP Server, either by itself or with WebSphere Application Server. The default service names correspond to the second install.

**Note:** Customized service names must be unique on your system.

2. **After installing**, you can use the WASService.exe utility in the *install\_root\bin* directory to manually define a Windows service for another installation instance or for another configuration instance of the server1 process.

You can use the **net start** and **net stop** commands to control the IBM HTTP Server services on a Windows system. For more information about these commands, see the Windows help file. Access these commands from the Start menu, clicking **Start > Programs > IBM HTTP Server**.

You can also use the **Start the Server** and **Stop the Server** commands to control the IBM WebSphere Application Server process on a Windows system. Access these commands from the Start menu, clicking **Start > Programs > IBM WebSphere > Application Server V6**.

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

For example, you can configure a server1 process as a monitored process. However, if you start the server1 process using the **startServer** command, the operating system does not monitor or restart the server1 process because the operating system did not originally start the process as a monitored process.

Return to Defining application server processes to continue.

## WASService command

The **WASService** command line tool lets you create a Windows service for any WebSphere Application Server Java process.

You can create Windows services for WebSphere Application Server Java processes. Potential Windows services include the following server processes:

- The default server1 process on an application server node
- Application server processes that you create on an application server node

When the installation wizard creates a Windows service, the uninstaller program can remove the Windows service. If you use the **WASService** command to create a service yourself, it is your responsibility to remove the service when it is no longer valid. The uninstaller program does not remove Windows services that you create with the **WASService** command.

**Location of the command file:** The WASService.exe command file is located in the *install\_root*\bin directory.

**Command syntax:**

**WASService.exe command syntax for starting an existing service**

The command syntax is as follows:

```
WASService.exe [-start] "service_name" [optional startServer.bat parameters]
```

**WASService.exe command syntax for creating a service or updating an existing service**

The command syntax is as follows:

```
WASService.exe -add "service_name"  
                -serverName server  
                -profilePath server_profile_directory  
  
                [-wasHome install_root]  
                [-configRoot configuration_repository_directory]  
                [-startArgs additional_start_arguments]  
                [-stopArgs additional_stop_arguments]  
                [-userid user_id -password password]  
                [-logFile service_log_file]  
                [-logRoot server_log_directory]  
                [-restart true | -restart false]  
                [-startType automatic | manual | disabled]
```

**WASService.exe command syntax for deleting a service**

The command syntax is as follows:

```
WASService.exe -remove "service_name"
```

**WASService.exe command syntax for stopping a running service**

The command syntax is as follows:

```
WASService.exe -stop "service_name" [optional stopServer.bat parameters]
```

**WASService.exe command syntax for retrieving service status**

The command syntax is as follows:

```
WASService.exe -status "service_name"
```

**WASService.exe command syntax for encoding parameters**

The command syntax is as follows:

```
WASService.exe -encodeParams "service_name"
```

**Parameters:** Supported arguments include:

**-add** "service\_name"

Creates a service named *service\_name* or updates an existing Windows service. The syntax is the same for both cases.

**-configRoot** *configuration\_repository\_directory*

Optional parameter that identifies the configuration directory of the installation root directory of a WebSphere Application Server product.

**-encodeParams** *service\_name*

Optional parameter that forces the service to encode the `-startArgs` and `-stopArgs` so that the arguments cannot be determined by editing the registry. Use the parameter when creating a service with the `-add` parameter by adding `-encodeParams` to the command line with no arguments. Or encode the parameters of an existing service:

```
WASService -encodeParams service_name
```

**-logFile** *service\_log\_file*

Optional parameter that identifies a log file that the **WASService** command uses to record its activity.

**-logRoot** *server\_log\_directory*

Required parameter that identifies the server log directory for the profile. The **WASService** command looks for a file named *server\_name*.pid to determine if the server is running.

**-profilePath** *server\_profile\_directory*

Specifies the directory path of the profile that defines the server process.

**-remove** *service\_name*

Deletes the specified service.

**-restart true | false**

Restarts the existing service automatically if the service fails when set to true.

**-serverName** *Server\_name*

Identifies the server that the service controls.

**-start "service\_name" [optional startServer.bat parameters]**

Starts the existing service.

**-startArgs** *additional\_start\_arguments*

Optional parameter that identifies additional parameters.

**-startType automatic | manual | disabled**

Defines the startup type of the new service. An automatic startup type starts automatically when the system starts or when the service is called for the first time. You must start a manual service before the operating system can load it and make it available. You cannot start a disabled service before changing the startup type.

**-status** *service\_name*

Returns the current status of the service, which includes whether the service is running or stopped.

**-stop service\_name [optional stopServer.bat parameters]**

Stops the specified service.

**-stopArgs** *additional\_stop\_arguments*

Optional parameter that identifies additional parameters.

**-userid user\_ID -password password**

Optional parameters that identify a privileged user ID and password that the Windows service will run as.

**-wasHome** *install\_root*

Optional parameter that identifies the installation root directory of the WebSphere Application Server product.

**Default names for Windows services that are created by the wizard:** The name of the Windows service that is created by the Installation wizard is IBM WebSphere Application Server V6 - DefaultNode.

**Viewing the Windows services panel:** To view Windows services, open the Control panel and click **Administrative Tools > Services**. Select a service to view information about it. Right click the service and click **Properties**. Four tabs provide information and functionality. For example, select the **Setup type** field on the **General** tab to change the setup type.

## **Examples:**

### **Creating an Application Server service**

This example creates a service called *IBM WebSphere Application Server V6 - server2* that starts an Application Server process:

```
WASService -add server2
            -servername server2
            -profilePath "C:\Program Files\IBM\WebSphere\AppServer\
                        profiles\CustomProfile"
            -wasHome "C:\Program Files\IBM\WebSphere\AppServer"
            -logfile "C:\Program Files\IBM\WebSphere\AppServer\
                    profiles\CustomProfile\logs\startNode.log"
            -logRoot "C:\Program Files\IBM\WebSphere\AppServer\
                    profiles\CustomProfile\logs"
            -restart true
```

After entering the command, messages that are similar to those in the following example display in the command window:

```
Adding Service: server2
    Config Root: C:\Program Files\IBM\WebSphere\AppServer\
                profiles\CustomProfile\config
    Server Name: server2
    Profile Path: C:\Program Files\IBM\WebSphere\AppServer\
                profiles\CustomProfile
    Was Home: C:\Program Files\IBM\WebSphere\AppServer\
    Start Args:
    Restart: 1
IBM WebSphere Application Server V6 - server2 service successfully added.
```

### **Updating an existing Application Server service**

This example updates an existing service called *IBM WebSphere Application Server V6 - server2* with additional stop arguments, username and password. The user name and password are required by the **stopServer** command to stop the application server with security enabled.

```
WASService -add server2
            -servername server2
            -profilePath "C:\Program Files\IBM\WebSphere\AppServer\
                        profiles\CustomProfile"
            -stopArgs "-username user_name -password password"
            -encodeParams server2
```

**Starting and stopping a server process after creating a Windows service:** If you issue the **startServer server1** command or the **stopServer server1** after creating a Windows service for server1, a message that is similar to the following example displays:

Because server1 is registered to run as a Windows Service, the request to start this server will be completed by starting the associated Windows Service.

### **Stopping a server after enabling security**

If you enable security while a Windows service is running, you cannot stop the server from the command line, even when using the username and password parameters on the **stopServer** command. A message similar to the following example is displayed:

```
Could not stop the IBM WebSphere Application Server V6 -
server_name service on Local Computer. The service
did not return an error. This could be an internal Windows
error or an internal service error. If the problem persists,
contact your system administrator.
```



The problem is due to the service control of the process. You must change the service to use the proper stop-server arguments for a secure server.

Use the `-stopArgs` parameter and the `-encodeParams` parameter to update the service as described in the "Updating an existing application server service" example.

---

## Java virtual machines (JVMs)

The Java virtual machine (JVM) is an interpretive computing engine responsible for running the byte codes in a compiled Java program. The JVM translates the Java byte codes into the native instructions of the host machine. The application server, being a Java process, requires a JVM in order to run, and to support the Java applications running on it. JVM settings are part of an application server configuration.

---

## Using the JVM

As part of configuring an application server, you might define settings that enhance your system's use of the Java virtual machine (JVM).

To view and change the JVM configuration for an application server's process, use the Java Virtual Machine page of the administrative console or use the `wsadmin` tool to change the configuration through scripting.

1. Access the Java Virtual Machine page.
  - a. Click **Servers > Application Servers** in the console navigation tree.
  - b. On the Application Server page, click on the name of the server whose JVM settings you want to configure.
  - c. On the settings page for the selected application server, click **Java and Process Management > Process Definition**.
  - d. On the Process Definition page, click **Java Virtual Machine**.
2. On the Java Virtual Machine page, specify values for the JVM settings as needed and click **OK**.
3. Click **Save** on the console task bar.
4. Restart the application server.

"Configuring application servers for UTF-8 encoding" on page 110 provides an example that involves specifying a value for the **Generic JVM Arguments** property on the Java Virtual Machine page to enable UTF-8 encoding on an application server. Enabling UTF-8 allows multiple language encoding support to be used in the administrative console.

"Configuring JVM sendRedirect calls to use context root" on page 156 provides an example that involves defining a property for the JVM.

## Java virtual machine settings

Use this page to view and change the Java virtual machine (JVM) configuration for the application server's process.

To view this administrative console page, click **Servers > Application Servers > *server\_name* > Process Definition > Java Virtual Machine**.

### Classpath

Specifies the standard class path in which the Java virtual machine code looks for classes.

Enter each classpath entry into a table row. You do not need to add the colon or semicolon at the end of each entry.



<b>Data type</b>	String
<b>Units</b>	Class path

## Boot Classpath

Specifies bootstrap classes and resources for JVM code. This option is only available for JVM instructions that support bootstrap classes and resources. You can separate multiple paths by a colon (:) or semi-colon (;), depending on operating system of the node.

<b>Data type</b>	String
------------------	--------

## Verbose Class Loading

Specifies whether to use verbose debug output for class loading. The default is not to enable verbose class loading.

<b>Data type</b>	Boolean
<b>Default</b>	false

## Verbose Garbage Collection

Specifies whether to use verbose debug output for garbage collection. The default is not to enable verbose garbage collection.

<b>Data type</b>	Boolean
<b>Default</b>	false

## Verbose JNI

Specifies whether to use verbose debug output for native method invocation. The default is not to enable verbose Java Native Interface (JNI) activity.

<b>Data type</b>	Boolean
<b>Default</b>	false

## Initial Heap Size

Specifies the initial heap size available to the JVM code, in megabytes.

Increasing the minimum heap size can improve startup. The number of garbage collection occurrences are reduced and a 10% gain in performance is realized.

Increasing the size of the Java heap improves throughput until the heap no longer resides in physical memory, in general. After the heap begins swapping to disk, Java performance suffers drastically.

<b>Data type</b>	Integer
<b>Default</b>	The default is 50.

## Maximum Heap Size

Specifies the maximum heap size available to the JVM code, in megabytes.

Increasing the heap size can improve startup. By increasing heap size, you can reduce the number of garbage collection occurrences with a 10% gain in performance.

Increasing the size of the Java heap improves throughput until the heap no longer resides in physical memory, in general. After the heap begins swapping to disk, Java performance suffers drastically. Set the maximum heap size low enough to contain the heap within physical memory.

<b>Data type</b>	Integer
<b>Default</b>	0 for iSeries, 256 for all other platforms. Keep the value low enough to avoid paging or swapping-out-memory-to-disk.

## Run HProf

This setting applies to base WebSphere Application Server only. It specifies whether to use HProf profiler support. To use another profiler, specify the custom profiler settings using the HProf Arguments setting. The default is not to enable HProf profiler support.

If you set the Run HProf property to true, then you must specify command-line profiler arguments as values for the HProf Arguments property.

<b>Data type</b>	Boolean
<b>Default</b>	false

## HProf Arguments

This setting applies to base WebSphere Application Server only. It specifies command-line profiler arguments to pass to the JVM code that starts the application server process. You can specify arguments when HProf profiler support is enabled.

HProf arguments are only required if the Run HProf property is set to true.

<b>Data type</b>	String
------------------	--------

## Debug Mode

Specifies whether to run the JVM in debug mode. The default is not to enable debug mode support.

If you set the Debug Mode property to true, then you must specify command-line debug arguments as values for the Debug Arguments property.

<b>Data type</b>	Boolean
<b>Default</b>	false

## Debug Arguments

Specifies command-line debug arguments to pass to the JVM code that starts the application server process. You can specify arguments when Debug Mode is enabled.

Debug arguments are only required if the Debug Mode property is set to true. If you enable debugging on multiple application servers on the same server, make sure that the servers are using different address arguments, which define the port for debugging. For example, if you enable debugging on two servers and leave the default debug port for each server as address=7777, the servers could fail to start properly.

<b>Data type</b>	String
<b>Units</b>	Java command-line arguments

## Generic JVM Arguments

Specifies command line arguments to pass to the Java virtual machine code that starts the application server process.

The following are optional command line arguments that you can use by entering them into the **General JVM Arguments** field. If you enter more than one argument, separate each argument by a space.

**Note:** If the argument says it is for the IBM Developer Kit only, you cannot use the argument with another JVM, such as the Sun JDK or the HP JDK.

- **-Xquickstart:** You can use **-Xquickstart** for initial compilation at a lower optimization level than in default mode. Later, depending on sampling results, you can recompile to the level of the initial compile in default mode. Use **-Xquickstart** for applications where early moderate speed is more important than long run throughput. In some debug scenarios, test harnesses and short-running tools, you can improve startup time between 15-20%.

The **-Xquickstart** option is not supported on OS/400.

- **-Xverify:none:** When using this value, the class verification stage is skipped during class loading. By using **-Xverify:none** with the just in time (JIT) compiler enabled, startup time is improved by 10-15%.

The **-Xverify:none** option is not supported on OS/400.

- **-Xnoclassgc:** You can use this value to disable class garbage collection, which leads to more class reuse and slightly improved performance. The trade-off is that you won't be collecting the resources owned by these classes. You can monitor garbage collection using the verbose:gc configuration setting, which will output class garbage collection statistics. Examining these statistics will help you understand the trade-off between the reclaimed resources and the amount of garbage collection required to reclaim the resources. However, if the same set of classes are garbage collected repeatedly in your workload, you should disable garbage collection. Class garbage collection is enabled by default.
- **-Xgcthreads:** You can use several garbage collection threads at one time, also known as *parallel garbage collection*. When entering this value in the **Generic JVM Arguments** field, also enter the number of processors that your machine has, for example, `-Xgcthreads=number_of_processors`. On a node with *n* processors, the default number of threads is *n*. You should use parallel garbage collection if your machine has more than one processor. This argument is valid only for the IBM Developer Kit.

The **-Xgcthreads** option is not supported on OS/400.

- **-Xnocompactgc:** This value disables heap compaction which is the most expensive garbage collection operation. Avoid compaction in the IBM Developer Kit. If you disable heap compaction, you eliminate all associated overhead.
- **-Xinitsh:** You can use this value to set the initial heap size where class objects are stored. The method definitions and static fields are also stored with the class objects. Although the system heap size has no upper bound, set the initial size so that you do not incur the cost of expanding the system heap size, which involves calls to the operating system memory manager. You can compute a good initial system heap size by knowing the number of classes loaded in the WebSphere Application Server product, which is about 8,000 classes, and their average size. Having knowledge of the applications helps you include them in the calculation. You can use this argument only with the IBM Developer Kit.
- **-Xgcpolicy:** You can use this value to set the garbage collection policy. If the garbage collection policy (gcpolicy) is set to optavgpause, concurrent marking is used to track application threads starting from the stack before the heap becomes full. The garbage collector pauses become uniform and long pauses are not apparent. The trade-off is reduced throughput because threads might have to do extra work. The default, recommended value is optthruput. Enter the value as `-Xgcpolicy:[optthruput|optavgpause]`. You can use this argument only with the IBM Developer Kit.
- **-XX:** The Sun-based Java Development Kit (JDK) Version 1.4.2 has generation garbage collection, which allows separate memory pools to contain objects with different ages. The garbage collection cycle collects the objects independently from one another depending on age. With additional parameters, you can set the size of the memory pools individually. To achieve better performance, set the size of the pool containing short lived objects so that objects in the pool do not live through more than one garbage collection cycle. The size of new generation pool is determined by the `NewSize` and `MaxNewSize` parameters. Objects that survive the first garbage collection cycle are transferred to another pool. The size of the survivor pool is determined by parameter `SurvivorRatio`. If garbage collection becomes a bottleneck, you can try customizing the generation pool settings. To monitor garbage collection statistics, use the object statistics in Tivoli Performance Viewer or the verbose:gc configuration setting. Enter the following values: `-XX:NewSize (lower bound)`, `-XX:MaxNewSize (upper bound)`, and `-XX:SurvivorRatio=NewRatioSize`. The default values are: `NewSize=2m MaxNewSize=32m SurvivorRatio=2`. However, if you have a JVM with more than 1 GB heap size, you should use the values: `-XX:newSize=640m -XX:MaxNewSize=640m -XX:SurvivorRatio=16`, or set 50 to 60% of total heap size to a new generation pool.

The **-XX** option is not supported on OS/400.

- **-Xminf:** You can use this value to specify the minimum free heap size percentage. The heap grows if the free space is below the specified amount. In reset enabled mode, this option specifies the minimum percentage of free space for the middleware and transient heaps. This is a floating point number, 0 through 1. The default is .3 (30%).
- **-server | -client:** Java HotSpot Technology in the Sun-based Java Development Kit (JDK) Version 1.4.2 introduces an adaptive JVM containing algorithms for optimizing byte code execution over time. The JVM runs in two modes, **-server** and **-client**. If you use the default **-client** mode, there will be a faster startup time and a smaller memory footprint, but lower extended performance. You can enhance performance by using **-server** mode if a sufficient amount of time is allowed for the HotSpot JVM to warm up by performing continuous execution of byte code. In most cases, use **-server** mode, which produces more efficient run-time execution over extended periods. You can monitor the process size and the server startup time to check the difference between **-client** and **-server**.

The **-server | -client** option is not supported on OS/400.

<b>Data type</b>	String
<b>Units</b>	Java command line arguments

### Executable JAR File Name

Specifies a full path name for an executable JAR file that the JVM code uses.

<b>Data type</b>	String
<b>Units</b>	Path name

### Disable JIT

Specifies whether to disable the just in time (JIT) compiler option of the JVM code.

If you disable the JIT compiler, throughput decreases noticeably. Therefore, for performance reasons, keep JIT enabled.

<b>Data type</b>	Boolean
<b>Default</b>	false (JIT enabled)
<b>Recommended</b>	JIT enabled

### Operating System Name

Specifies JVM settings for a given operating system. When started, the process uses the JVM settings for the operating system of the server.

<b>Data type</b>	String
------------------	--------

## Configuring JVM sendRedirect calls to use context root

If the `com.ibm.websphere.sendredirect.compatibility` property is not set and your application servlet code has statements such as `sendRedirect("/home.html")`, your Web browser might display messages such as *Error 404: No target servlet configured for uri: /home.html*. To instruct the server not to use the Web server's document root and to use instead the Web application's context root for `sendRedirect()` calls, configure the JVM by setting the `com.ibm.websphere.sendredirect.compatibility` property to a `true` or `false` value.

1. Access the settings page for a property of the JVM.
  - a. Click **Servers > Application Servers** in the console navigation tree.
  - b. On the Application Server page, click on the name of the server whose JVM settings you want to configure.

- c. On the settings page for the selected application server, under Server Infrastructure, click **Java and Process Management > Process Definition**.
  - d. On the Process Definition page, click **Java Virtual Machine**.
  - e. On the Java Virtual Machine page, click **Custom Properties**.
  - f. On the Custom Properties page, click **New**.
2. On the settings page for a property, specify a name of `com.ibm.websphere.sendredirect.compatibility` and either `true` or `false` for the value, then click **OK**.
  3. Click **Save** on the console task bar.
  4. Stop the application server and then restart the application server.

## Setting custom JVM properties

In the WebSphere Application Server administrative console, you can change the values of the following custom JVM properties:

### **com.ibm.websphere.network.useMultiHome**

#### **For a distributed platform:**

Set this property in a multihomed environment where WebSphere Application Server is restricted to listen only on a specific IP address for Discovery and SOAP messages.

The settings for the **com.ibm.websphere.network.useMultiHome** property are as follows:

- Setting this property to `false` specifies that WebSphere Application Server will listen on all IP addresses on the host for Discovery and SOAP messages.
- Setting this property to `true` specifies that WebSphere Application Server will only listen on the configured host name for Discovery and SOAP messages. If you set this property to `true`, you should have a host name configured on WebSphere Application Server that resolves to a specific IP address.
- Setting this property to `null` specifies that WebSphere Application Server will only listen on the default IP address only.

If you cannot contact the server, check the setting for **com.ibm.websphere.network.useMultihome** to ensure it is correct.

You can change the value through the administrative console. Modify the defaults by setting the value for the server, deployment manager, and node agent. In order for these changes to take place, you must restart the server.

#### **Steps for this task**

1. To set this property, connect to the administrative console and navigate to the indicated page.

Application server	<b>Servers &gt; Application Servers &gt; <i>server1</i> &gt; Process Definition &gt; Control &gt; Java Virtual Machine &gt; Custom Properties</b>
Deployment manager	<b>System Administration &gt; Deployment Manager &gt; Process definition &gt; Control &gt; Java Virtual Machine &gt; Custom Properties</b>
Node agent	<b>System Administration &gt; Node Agent &gt; <i>nodeagent</i> &gt; Process definition &gt; Control &gt; Java Virtual Machine &gt; Custom Properties</b>

2. If the **com.ibm.websphere.network.useMultiHome** property is not present in the list, create a new property name and indicate its value.
3. Restart the server.

### **com.ibm.websphere.deletejspclasses**

Deletes JavaServer Pages classes for all applications after those applications have been deleted or updated. By default, the value of this property is true.

#### **Steps for this task**

1. Connect to the administrative console and navigate to the Java Virtual Machine Custom Properties panel.

#### **For a distributed platform:**

Base configuration	<b>Servers &gt; Application Servers &gt; <i>server1</i></b> . Then, under Server Infrastructure, click <b>Java and Process Management &gt; Process definition &gt; Java Virtual Machine &gt; Custom Properties</b>
ND configuration	<b>System Administration &gt; Node Agents &gt; <i>nodeagent</i></b> . Then, under Server Infrastructure, click <b>Java and Process Management &gt; Process definition &gt; Java Virtual Machine &gt; Custom Properties</b>

2. If the **com.ibm.websphere.deletejspclasses** property is not present in the list, create a new property name.
3. Enter the name and value.

### **com.ibm.websphere.deletejspclasses.delete**

Deletes JavaServer Pages classes for all applications after those applications have been deleted, but not after they have been updated. By default, the value of this property is true.

#### **Steps for this task**

1. Connect to the administrative console and navigate to the Java Virtual Machine Custom Properties panel.

#### **For a distributed platform:**

Base configuration	<b>Servers &gt; Application Servers &gt; <i>server1</i> &gt; Process definition &gt; Java Virtual Machine &gt; Custom Properties</b>
ND configuration	<b>System Administration &gt; Node Agents &gt; <i>nodeagent</i></b> . Then, under Server Infrastructure, click <b>Java and Process Management &gt; Process definition &gt; Java Virtual Machine &gt; Custom Properties</b>

2. If the **com.ibm.websphere.deletejspclasses.delete** property is not present in the list, create a new property name.
3. Enter the name and value.

### **com.ibm.websphere.deletejspclasses.update**

Deletes JavaServer Pages classes for all applications after those applications have been updated, but not after they have been deleted. By default, the value of this property is true.

#### **Steps for this task**

1. Connect to the administrative console and navigate to the Java Virtual Machine Custom Properties panel.

#### **For a distributed platform:**

Base configuration	<b>Servers &gt; Application Servers &gt; <i>server1</i></b> . Then, under Server Infrastructure, click <b>Java and Process Management &gt; Process definition &gt; Java Virtual Machine &gt; Custom Properties</b>
--------------------	--

ND configuration	<b>System Administration &gt; Node Agents &gt; nodeagent.</b> Then, under Server Infrastructure, click <b>Java and Process Management &gt; Process definition &gt; Java Virtual Machine &gt; Custom Properties</b>
------------------	---

2. If the **com.ibm.websphere.deletejspclasses.update** property is not present in the list, create a new property name.
3. Enter the name and value.

## Tuning Java virtual machines

The application server, being a Java process, requires a Java virtual machine (JVM) to run, and to support the Java applications running on it. As part of configuring an application server, you can fine-tune settings that enhance system use of the JVM. In addition to the following tuning parameters, see also “Java memory tuning tips” on page 160.

Use the following JVM parameters, including garbage collection options for IBM Developer Kit 1.4.2, to tune the Java virtual machine. For instructions on view and change the JVM configuration, go to “Using the JVM” on page 152. For information on specifying any of the following settings, go to “Java virtual machine settings” on page 152.

- **Specify any or all of the following generic JVM arguments.** These optional command line arguments are passed to the Java virtual machine code that starts the application server process.
  - Quickstart (-Xquickstart)
  - Avoiding class verification (-Xverify:none)
  - Class garbage collection (-Xnoclassgc)
  - Garbage collection threads (-Xgcthreads)
  - Garbage collection policy (-Xgcpolicy)
  - Sun JDK 1.4.2 Generational Garbage Collection (-XX)

You can find more information about generational garbage collection at <http://java.sun.com/docs/hotspot/gc/index.html>.

  - Sun Java Development Kit 1.4.2 HotSpot JVM warm-up (-server)
  - Heap compaction (-Xnocomcompactgc)
  - Initial system heap size (-Xinitsh)
- **Set the initial heap size.**
- **Set the maximum heap size.**

---

## Preparing to host applications

Rather than use the default application server provided with the product, you can configure a new server and set of resources.

The default application server and a set of default resources are available to help you begin quickly. You can choose instead to configure a new server and set of resources. Here is what you need to do in order to set up a run-time environment to support applications.

1. Create an application server.
2. Create a virtual host.
3. Configure a Web container. See the *Developing and deploying applications* PDF for information on how to perform this task.
4. Configure an EJB container. See the *Developing and deploying applications* PDF for information on how to perform this task.
5. Create resources for data access. See the *Developing and deploying applications* PDF for information on how to perform this task.



6. Create a JDBC provider and data source. See the *Developing and deploying applications* PDF for information on how to perform this task.
7. Create a URL and URL provider. See the *Developing and deploying applications* PDF for information on how to perform this task.
8. Create a JavaMail session. See the *Developing and deploying applications* PDF for information on how to perform this task.
9. Create resources for session support. See the *Developing and deploying applications* PDF for information on how to perform this task.
10. Configure a Session Manager. See the *Developing and deploying applications* PDF for information on how to perform this task.

---

## Java memory tuning tips

Enterprise applications written in the Java language involve complex object relationships and utilize large numbers of objects. Although, the Java language automatically manages memory associated with object life cycles, understanding the application usage patterns for objects is important. In particular, verify the following:

- The application is not over-utilizing objects
- The application is not leaking objects
- The Java heap parameters are set properly to handle a given object usage pattern

Understanding the effect of garbage collection is necessary to apply these management techniques.

### The garbage collection bottleneck

Examining Java garbage collection gives insight to how the application is utilizing memory. Garbage collection is a Java strength. By taking the burden of memory management away from the application writer, Java applications are more robust than applications written in languages that do not provide garbage collection. This robustness applies as long as the application is not abusing objects. Garbage collection normally consumes from 5% to 20% of total execution time of a properly functioning application. If not managed, garbage collection is one of the biggest bottlenecks for an application, especially when running on symmetric multiprocessing (SMP) server machines. The Java virtual machine (JVM) uses a parallel garbage collector to fully exploit an SMP during most garbage collection cycles where the Sun HotSpot 1.3.1 JVM has a single-threaded garbage collector. For more information about garbage collection in a Solaris operating environment see the *Troubleshooting and support* PDF.

### The garbage collection gauge

You can use garbage collection to evaluate application performance health. By monitoring garbage collection during the execution of a fixed workload, you gain insight as to whether the application is over-utilizing objects. Garbage collection can even detect the presence of memory leaks.

You can monitor garbage collection statistics using object statistics in the Tivoli Performance Viewer, or using the **verbose:gc** JVM configuration setting. The **verbose:gc** format is not standardized between different JVMs or release levels. For a description of the IBM verbose:gc output and more information about the IBM garbage collector, see the *Troubleshooting and support* PDF.

For this type of investigation, set the minimum and maximum heap sizes to the same value. Choose a representative, repetitive workload that matches production usage as closely as possible, user errors included.

To ensure meaningful statistics, run the fixed workload until the application state is steady. It usually takes several minutes to reach a steady state.

### Detecting over-utilization of objects



You can use the Tivoli Performance Viewer to check if the application is overusing objects, by observing the counters for the JVM runtime. You have to set the **-XrunpmiJvmpiProfiler** command line option, as well as the JVM module maximum level in order to enable the Java virtual machine profiler interface (JVMPPI) counters. The best result for the average time between garbage collections is at least 5-6 times the average duration of a single garbage collection. If you do not achieve this number, the application is spending more than 15% of its time in garbage collection.

If the information indicates a garbage collection bottleneck, there are two ways to clear the bottleneck. The most cost-effective way to optimize the application is to implement object caches and pools. Use a Java profiler to determine which objects to target. If you can not optimize the application, adding memory, processors and clones might help. Additional memory allows each clone to maintain a reasonable heap size. Additional processors allow the clones to run in parallel.

## Detecting memory leaks

Memory leaks in the Java language are a dangerous contributor to garbage collection bottlenecks. Memory leaks are more damaging than memory overuse, because a memory leak ultimately leads to system instability. Over time, garbage collection occurs more frequently until the heap is exhausted and the Java code fails with a fatal Out of Memory exception. Memory leaks occur when an unused object has references that are never freed. Memory leaks most commonly occur in collection classes, such as Hashtable because the table always has a reference to the object, even after real references are deleted.

High workload often causes applications to crash immediately after deployment in the production environment. This is especially true for leaking applications where the high workload accelerates the magnification of the leakage and a memory allocation failure occurs.

The goal of memory leak testing is to magnify numbers. Memory leaks are measured in terms of the amount of bytes or kilobytes that cannot be garbage collected. The delicate task is to differentiate these amounts between expected sizes of useful and unusable memory. This task is achieved more easily if the numbers are magnified, resulting in larger gaps and easier identification of inconsistencies. The following list contains important conclusions about memory leaks:

- **Long-running test**

Memory leak problems can manifest only after a period of time, therefore, memory leaks are found easily during long-running tests. Short running tests can lead to false alarms. It is sometimes difficult to know when a memory leak is occurring in the Java language, especially when memory usage has seemingly increased either abruptly or monotonically in a given period of time. The reason it is hard to detect a memory leak is that these kinds of increases can be valid or might be the intention of the developer. You can learn how to differentiate the delayed use of objects from completely unused objects by running applications for a longer period of time. Long-running application testing gives you higher confidence for whether the delayed use of objects is actually occurring.

- **Repetitive test**

In many cases, memory leak problems occur by successive repetitions of the same test case. The goal of memory leak testing is to establish a big gap between unusable memory and used memory in terms of their relative sizes. By repeating the same scenario over and over again, the gap is multiplied in a very progressive way. This testing helps if the number of leaks caused by the execution of a test case is so minimal that it is hardly noticeable in one run.

You can use repetitive tests at the system level or module level. The advantage with modular testing is better control. When a module is designed to keep the private module without creating external side effects such as memory usage, testing for memory leaks is easier. First, the memory usage before running the module is recorded. Then, a fixed set of test cases are run repeatedly. At the end of the test run, the current memory usage is recorded and checked for significant changes. Remember, garbage collection must be suggested when recording the actual memory usage by inserting `System.gc()` in the module where you want garbage collection to occur, or using a profiling tool, to force the event to occur.

- **Concurrency test**

Some memory leak problems can occur only when there are several threads running in the application. Unfortunately, synchronization points are very susceptible to memory leaks because of the added complication in the program logic. Careless programming can lead to kept or unreleased references. The incident of memory leaks is often facilitated or accelerated by increased concurrency in the system. The most common way to increase concurrency is to increase the number of clients in the test driver.

Consider the following points when choosing which test cases to use for memory leak testing:

- A good test case exercises areas of the application where objects are created. Most of the time, knowledge of the application is required. A description of the scenario can suggest creation of data spaces, such as adding a new record, creating an HTTP session, performing a transaction and searching a record.
- Look at areas where collections of objects are used. Typically, memory leaks are composed of objects within the same class. Also, collection classes such as Vector and Hashtable are common places where references to objects are implicitly stored by calling corresponding insertion methods. For example, the get method of a Hashtable object does not remove its reference to the retrieved object.

Tivoli Performance Viewer can help find memory leaks. For best results, repeat experiments with increasing duration, like 1000, 2000, and 4000-page requests. The Tivoli Performance Viewer graph of used memory should have a sawtooth shape. Each drop on the graph corresponds to a garbage collection. There is a memory leak if one of the following occurs:

- The amount of memory used immediately after each garbage collection increases significantly. The sawtooth pattern looks more like a staircase.
- The sawtooth pattern has an irregular shape.

Also, look at the difference between the number of objects allocated and the number of objects freed. If the gap between the two increases over time, there is a memory leak.

Heap consumption indicating a possible leak during a heavy workload (the application server is consistently near 100% CPU utilization), yet appearing to recover during a subsequent lighter or near-idle workload, is an indication of heap fragmentation. Heap fragmentation can occur when the JVM can free sufficient objects to satisfy memory allocation requests during garbage collection cycles, but the JVM does not have the time to compact small free memory areas in the heap to larger contiguous spaces.

Another form of heap fragmentation occurs when small objects (less than 512 bytes) are freed. The objects are freed, but the storage is not recovered, resulting in memory fragmentation until a heap compaction has been run.

To avoid heap fragmentation, turn on the `-Xcompactgc` flag in the JVM advanced settings command line arguments. The `-Xcompactgc` function verifies that each garbage collection cycle eliminates fragmentation. However, compaction is a relatively expensive operation. See the heap compaction command line argument (`-Xnocompactgc`) in “Java virtual machine settings” on page 152 for more information.

## Java heap parameters

The Java heap parameters also influence the behavior of garbage collection. Increasing the heap size supports more object creation. Because a large heap takes longer to fill, the application runs longer before a garbage collection occurs. However, a larger heap also takes longer to compact and causes garbage collection to take longer. Refer to the sections on Java Heap sizes in “Java virtual machine settings” on page 152 for more information.

*For performance analysis, the initial and maximum heap sizes should be equal.*

When tuning a production system where the working set size of the Java application is not understood, a good starting value for the initial heap size is 25% of the maximum heap size. The JVM then tries to adapt the size of the heap to the working set size of the application.

The illustration represents three CPU profiles, each running a fixed workload with varying Java heap settings. In the middle profile, the initial and maximum heap sizes are set to 128MB. Four garbage collections occur. The total time in garbage collection is about 15% of the total run. When the heap parameters are doubled to 256MB, as in the top profile, the length of the work time increases between garbage collections. Only three garbage collections occur, but the length of each garbage collection is also increased. In the third profile, the heap size is reduced to 64MB and exhibits the opposite effect. With a smaller heap size, both the time between garbage collections and the time for each garbage collection are shorter. For all three configurations, the total time in garbage collection is approximately 15%. This example illustrates an important concept about the Java heap and its relationship to object utilization. There is always a cost for garbage collection in Java applications.

Run a series of test experiments that vary the Java heap settings. For example, run experiments with 128MB, 192MB, 256MB, and 320MB. During each experiment, monitor the total memory usage. If you expand the heap too aggressively, paging can occur. Use the **vmstat** command or the Windows NT or Windows 2000 Performance Monitor to check for paging. If paging occurs, reduce the size of the heap or add more memory to the system. When all the runs are finished, compare the following statistics:

- Number of garbage collection calls
- Average duration of a single garbage collection call
- Ratio between the length of a single garbage collection call and the average time between calls

If the application is not over-utilizing objects and has no memory leaks, the state of steady memory utilization is reached. Garbage collection also occurs less frequently and for short duration.

If the heap free space settles at 85% or more, consider decreasing the maximum heap size values because the application server and the application are under-utilizing the memory allocated for heap.

For more information about garbage collection see the *Troubleshooting and support* PDF.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

---

## Configuring multiple network interface card support

Use this task to first prepare your server for multiple network interface card support and enable multiple network interface card logic, including configuring multiple network interface cards to coexist on the same machine.

1. Configure object request broker (ORB) properties to support multiple network interface cards.
  - Set the `com.ibm.CORBA.LocalHost` property to resolve to a valid host name or IP address. The value should not resolve to a local host or loop back address (for example, 127.0.0.1).
  - a. In the administrative console, click **Servers > Application servers > *server\_name* > Java and process management > Process Definition > Java Virtual Machine**.
  - b. Add the following generic JVM argument: Add this argument as a single line. It is split here for printing purposes.

```
-Dcom.ibm.CORBA.LocalHost=xxxx.xx.xx.xx -Dcom.ibm.websphere.network.useMultiHome=false -Dcom.ibm.CORBA.ServerSocketQueueDepth=50 -Dcom.ibm.ws.orb.transport.useMultiHome=false
```
2. Update the host name for all HTTP transport chains. In the administrative console, click **Servers > Application servers > *server\_name* > Web container settings > Web container transport chains**. Update the host name for all of the transport chains listed on this page. The value should not resolve to a local host or loop back address (for example, 127.0.0.1).
3. Change the initial state to **stopped** for each of the listener ports. In the administrative console, click:

- **Servers > Application servers > *server\_name* > Messaging > Message listener service > Listener ports > SamplePtoPListenerPort**
- **Servers > Application servers > *server\_name* > Messaging > Message Listener Service > Listener Ports > SamplePubSubListenerPort**

Update the state to **stopped** for each port.

4. Change the initial state of the JMS server to stopped. In the administrative console, click **Servers > JMS Servers > *JMS\_server* > Initial state > stopped**.
5. Apply these changes to all application servers and the deployment manager.

By completing these steps, you enabled multiple network interface card support.

To configure multiple application servers to coexist on a single machine that is using two network interface cards, perform the following steps:

1. Install the WebSphere Application Server base product for each network interface card. To install on distributed platforms, see the *Installing your application serving environment* PDF for more information.
2. Start the server that is on the first network interface card. Follow the preceding steps in this task to prepare this server for multiple network interface card support and enable multiple network interface card logic. After you complete this step, stop the server.
3. Start the server that is on the other network interface card. Follow the preceding steps in this task to prepare this server for multiple network interface card support and enable multiple network interface card logic. After you complete this step, stop the server.
4. Start the servers on both network interface cards.

By completing these steps, you enabled multiple application servers to coexist on a single machine that has two network interface cards.

---

## Tuning application servers

The WebSphere Application Server contains interrelated components that must be harmoniously tuned to support the custom needs of your end-to-end e-business application.

This group of interrelated components is known as the queuing network. The queuing network helps the system achieve maximum throughput while maintaining the overall stability of the system.

The follow steps describe various tuning tasks that may improve your application server performance. You can choose to implement any of these application server settings.

- **Tune the object request broker.** An Object Request Broker (ORB) manages the interaction between clients and servers, using the Internet InterORB Protocol (IIOP). It supports client requests and responses received from servers in a network-distributed environment. You can tune the ORB with the following parameters:
  - Set **Pass by reference (com.ibm.CORBA.iiop.noLocalCopies)** as described in the *Developing and deploying applications* PDF.
  - Set the **Connection cache minimum (com.ibm.CORBA.MaxOpenConnections)** as described in the *Developing and deploying applications* PDF.
  - Set **Maximum size** as described in the *Developing and deploying applications* PDF.
  - Set **com.ibm.CORBA.ServerSocketQueueDepth** as described in the *Developing and deploying applications* PDF.
  - Set the **com.ibm.CORBA.FragmentSize** as described in the *Developing and deploying applications* PDF.

The the *Developing and deploying applications* PDF offer tips on using these parameters to tune the ORB.

- **Tune the XML parser definitions.**

- **Description:** Facilitates server startup by adding XML parser definitions to the `jaxp.properties` and `xerces.properties` files in the `${install_root}/jre/lib` directory. The `XMLParserConfiguration` value might change as new versions of Xerces are provided.
- **How to view or set:** Insert the following lines in both files:
 

```
javax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl
javax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.
    DocumentBuilderFactoryImpl
org.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.parsers.
    StandardParserConfiguration
```
- **Default value:** None
- **Recommended value:** None
- **Tune the dynamic cache service.** Using the dynamic cache service can improve performance. See the *Developing and deploying applications* PDF for information about using the dynamic cache service and how it can affect your application server performance.
- **Tune the Web container.** The WebSphere Application Server Web container manages all HTTP requests to servlets, JSPs and Web services. Requests flow through a transport chain to the Web container. The transport chain defines the important tuning parameters for performance for the Web container. There is a transport chain for each TCP port that WebSphere Application Server is listening on for HTTP requests. For example, the default HTTP port 9080 is defined in Web container inbound channel chain. Use the following parameters to tune the Web container:
  - HTTP requests are processed by a pool of server threads. The minimum and maximum thread pool size for the Web container can be configured for optimal performance. Generally, 5 to 10 threads per server CPU will provide the best throughput. The number of threads configured does not represent the number of requests WebSphere can process concurrently. Requests are queued in the transport chain when all threads are busy. To specify the thread pool settings:
    1. Click **Servers > Application Servers > *server\_name* Web Container Settings > Web Container > Web container transport chains.**
    2. Select the normal inbound chain for serving requests. This will usually be named `WCInboundDefault`, on port 9080.
    3. Click **TCP Inbound Channel (TCP\_2).**
    4. Set **Thread Pools** under Related Items.
    5. Select **WebContainer.**
    6. Enter values for **Minimum Size** and **Maximum Size.**
  - The HTTP 1.1 protocol provides a "keep-alive" feature to enable the TCP connection between HTTP clients and the server to remain open between requests. By default WebSphere Application Server will close a given client connection after a number of requests or a timeout period. After a connection is closed, it will be recreated if the client issues another request. Early closure of connections can reduce performance. Enter a value for the maximum number of persistent requests to (keep-alive) to specify the number of requests that are allowed on a single HTTP connection. Enter a value for persistent timeouts to specify the amount of time, in seconds, that the HTTP transport channel allows a socket to remain idle between requests. To specify values for Maximum persistent requests and Persistent timeout:
    1. Click **Servers > Application Servers > *server\_name* Web Container Settings > Web Container > Web container transport chains.**
    2. Select the normal inbound chain for serving requests. This will usually be named `WCInboundDefault`, on port 9080.
    3. Click **HTTP Inbound Channel (HTTP\_2).**
    4. Enter values for **Maximum persistent requests** and **Persistent timeout.**
- **Tune the EJB container.** An EJB container is automatically created when you create an application server. After the EJB container is deployed, you can use the following parameters to make adjustments that improve performance.
  - Set the **Cleanup interval** and the **Cache size** as described in the *Developing and deploying applications* PDF.
  - **Break CMP enterprise beans into several enterprise bean modules** while Assembling EJB modules. See the *Developing and deploying applications* PDF for more information on how to perform this task.



- **Tune the session management.** The installed default settings for session management are optimal for performance. See the *Developing and deploying applications* PDF for more information about tuning session management.
- **Tune the data sources.** A data source is used to access data from the database. The following parameters reveal how the number of physical connections within a connection pool can change performance.
  - Set the **Maximum connection pool** and **Minimum connection pool** as described in the *Developing and deploying applications* PDF.
  - Set the **Statement cache size** as described in the *Developing and deploying applications* PDF.

---

## Web services client to Web container optimized communication

To improve performance, there is an optimized communication path between a Web services client application and a Web container that are located in the same application server process. Requests from the Web services client that are normally sent to the Web container using a network connection are delivered directly to the Web container using an optimized local path. The local path is available because the Web services client application and the Web container are running in the same process.

This direct communication eliminates the need for clients and web containers that are in the same process to communicate over the network. For example, a Web services client might be running in an application server. Instead of accessing the network to communicate with the Web container, the Web services client can communicate with the Web container using the optimized local path. This optimized local path improves the performance of the application server by allowing Web services clients and Web containers to communicate without using network transports.

In a clustered environment, there is typically an HTTP server (such as IBM HTTP server) that handles incoming client requests, distributing them to the correct application server in the cluster. The HTTP server uses information about the requested application and the defined virtual hosts to determine which application server receives the request. The Web services client also uses the defined virtual host information to determine whether the request can be served by the local Web container. You must define unique values for the host and port on each application server. You cannot define the values of host and port as wild cards denoted by the asterisk symbol (\*) when you enable the optimized communication between the Web services application and the Web container. Using wild cards indicate that the local Web container can handle Web services requests for all destinations.

The optimized local communication path is disabled by default. You can enable the local communication path with the `enableInProcessConnections` custom property. Before configuring this custom property, make sure that you are not using wild cards for host names in your Web container end points. Set this property to **true** in the Web container to enable the optimized local communication path. When disabled, the Web services client and the Web container communicate using network transports.

For information about how to configure the `enableInProcessConnections` custom property, see the *Developing and deploying applications* PDF.

When the optimized local communication path is enabled, logging of requests through the local path uses the same log attributes as the network channel chain for the Web container. To use a different log file for in process requests than the log file for network requests, use a custom property on the HTTP Inbound Channel in the transport chain. Use the `inProcessLogFilenamePrefix` custom property to specify a string that is added to the beginning of the network log file name to create a file name that is unique. Requests through the local process path are logged to this specified file. For example, if the log filename is `../httpaccess.log` for a network chain, and the `inProcessLogFilenamePrefix` custom property is set to "local" on the HTTP channel in that transport chain, the local log file name for requests to the host associated with that chain is `/localhttpaccess.log`.

---

## Application servers: Resources for learning

Use the following links to find relevant supplemental information about configuring application servers. 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.

View links to additional information about:

- Programming instructions and examples
- Programming specifications
- Administration

### Programming instructions and examples

- WebSphere Application Server education at <http://www.ibm.com/software/webservers/learn/>.

### Programming specifications

- The Java™ Virtual Machine Specification, Second Edition at <http://java.sun.com/docs/books/vmspec/>.
- Sun's technology forum for the Java™ Virtual Machine Specification at <http://forum.java.sun.com/forum.jsp?forum=37>

### Administration

- Listing of all IBM WebSphere Application Server Redbooks at <http://publib-b.boulder.ibm.com/Redbooks.snf/Portals/WebSphere>.





---

## Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
500 Columbus Avenue  
Thornwood, New York 10594 USA



---

## **Trademarks and service marks**

For trademark attribution, visit the IBM Terms of Use Web site (<http://www.ibm.com/legal/us/>).