IBM WebSphere Application Server, Version 5

IBM

Environment

Compilation date: November 13, 2002

Contents

Trademarks and service marks

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

- Everyplace
- iSeries
- IBM
- Redbooks
- ViaVoice
- WebSphere
- zSeries

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

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

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

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

Chapter 1. Welcome to Environment

The environment of the product applies to the configuring of Web server plug-ins and of objects that you want to be 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).

If you specify a Web server when installing the WebSphere Application Server product, the installation program changes the Web server configuration file automatically to establish a plug-in.

You do not need a Web server plug-in or Web server to start the application server or the administrative console. In a test or development environment, you can use the internal HTTP transport instead of the Web server plug-in and Web server. But, for performance reasons, you must use a Web server plug-in and Web server in a production environment.

The WebSphere Application Server documentation provides (information on plug-ins) but does not provide information on administering Web servers. To learn how to administer your Web server, refer to documentation for your Web server.

Cell-wide settings

The configuration data for Version 5.0 of WebSphere Application Server is stored in files, and those files exist in one of several directories in the configuration repository tree. The directory in which a configuration file exists determines its scope, or how broadly or narrowly that data applies. Files in an individual server directory apply to that specific server only. Files in a node level directory apply to every server on that node. And files in the cell directory apply to every server on every node within the entire cell.

Cell-wide settings are configuration data that is stored in files in the cell directory and those files are replicated to every node in the cell. There are several different configuration settings that apply to the entire cell. These include the definition of (virtual hosts), (shared libraries), and any (variables) that you want to be consistent throughout the entire cell.

Chapter 2. Configuring Web server plug-ins

A WebSphere application server works with a Web server to handle requests for Web applications. The Web server and application server communicate using a WebSphere HTTP plug-in for the Web server.

The installation program for WebSphere Application Server modifies the Web server configuration file automatically to establish a plug-in, provided that you specify a Web server during installation.

Plug-ins are the preferred method of communication between the Web server and the application server. A plug-in offers the following advantages:

- XML-based configuration file
- Standard protocol recognized by firewall products
- Security using HTTPS, replacing proprietary OSE over SSL

A Web server plug-in and Web server are not required in order to start the application server or the administrative console. In a test or development environment, you can use the internal HTTP transport instead of the Web server plug-in and Web server. But, for performance reasons, you must use a Web server plug-in and Web server in a production environment. An HTTP transport facilitates the connection between the Web server plug-in and a Web container of an application server.

Steps for this task

1. Administer your Web server. Refer to your Web server documentation for information on administering your Web server.

   Ensure that Web servers are 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 httpd.conf for 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 IBM HTTP Server:

   HTTP method POST is not supported by this URL.

2. **(Optional)** If you encounter problems starting your Web server, check the http_plugin.log file in the WebSphere logs directory for information on what portion of the plug-in configuration file contained the 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. A frequent reason for a Web server not starting is an improper entry in a plug-in configuration file.

3. **(Optional)** Install the plug-in to a specific location.

4. **(Optional)** Check the version of your IBM HTTP Server installation.

5. **(Optional)** Manually edit the plug-in configuration.

6. Regenerate the plug-in configuration. After changing configurations to plug-ins, transports or virtual hosts, you must regenerate your Web server plug-in for the changes to take effect.

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.

You have the option to install WebSphere Application Server plug-ins for your Web servers when the application server is installed.

The plug-ins use a configuration file to determine whether a request should be handled by the Web server or the application server.

Installing plug-ins to specific locations

Sometimes, you might want to install the WebSphere Application Server plug-ins for Web servers to locations other than their default locations.

Installing Web server plug-ins to non-default locations

Depending on the operating system, when you run the WebSphere Application Server installation program and select to install a Web server plug-in, the plug-in might install silently if the Web server is installed in a standard location for that Web server brand. In such a case, the WebSphere Application Server installation program does not prompt you for the configuration file location.

If you have installed two Web servers on the same machine, the plug-in operates properly for the server installed at the "standard" location, but the plug-in does not operate properly for the server that is in the non-standard location.

For example, suppose IBM HTTP Server "installation 1" is installed in the standard directory in which the Web server is installed on Solaris, /opt/IBMHTTPD. When the plug-in for this first Web server is installed, it silently modifies the httpd.conf file in /opt/IBMHTTPD/conf, which is appropriate.

But now suppose that you installed IBM HTTP Server "installation 2" in /opt/IHS2. When the Web server plug-in for this second Web server is installed, it too is installed in httpd.conf in /opt/IBMHTTPD/conf.

One way around this is to manually edit the httpd.conf file of the second IBM HTTP Server installation, rather than installing the plug-in. You could copy the modified configuration file from the first Web server and make it the configuration file of the second server. Of course, this approach assumes that you have not customized the configuration file of the second server, and find it acceptable for its configuration to match that of the first server.

Another way is to install both Web servers in some non-standard place, such as /opt/IHS1 and /opt/IHS2. When the WebSphere Application Server installation program cannot find either httpd.conf file, you are prompted for the one to edit. Specify the location of one of the Web server files. Then repeat the plug-in installation, specifying the other location this time.

Installing WebSphere Application Server plug-ins on non-default IIS servers

When the product installs the WebSphere Application Server plug-in for Microsoft Internet Information Server (IIS), it assumes the user wants to attach the plug-in to the default IIS server. The following instructions explain how to attach the plug-in to an IIS server other than the default.

The procedure might be necessary for a user implementing multiple Web sites that separate the pages they serve along some logical boundary, such as security level. Follow the steps to allow a newly defined site (using an IIS instance other than the default) to exercise servlets in conjunction with an existing site or sites.

The instructions apply to IIS Versions 4.x and 5.x.

Steps for this task
1. Use the Internet Service Manager (from Microsoft IIS) to create a new site with a name, port number, and base subdirectory.
2. Go to the WebSphere Application Server administrative console and (configure the virtual host to contain an alias for the port number used by the site).
3. Create a virtual directory for the new site.
   a. Open the Internet Service Manager for IIS.
   b. Select the new site in the tree view.
   c. Right-click to display a menu. Select **New > Virtual Directory**.
   d. Ensure that the values are set appropriately:
      • The name of the virtual directory should be set to SEPLUGINS (using all capital letters).
      • The physical path should be set to the WebSphere Application Server bin directory (such as c:\WebSphere\AppServer\bin on Windows NT).
        Note, the directory must have the EXECUTE permission set, but setting any other permissions (or allowing it to inherit other permissions) is a security risk.
4. Start the new site.
5. Issue a browser request to verify that the configuration works, such as:
   ```
   http://mymachine:8080/servlet/snoop
   ```
6. The administrator must ensure that the SEPLUGINS retains its EXECUTE permission.
   It is possible for virtual directories under a site to inherit properties from the site. Ensure that the SEPLUGINS virtual directory does not inherit permissions from changes to the site, if those changes involve withdrawing the EXECUTE permission.

Checking your IBM HTTP Server version

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

Steps for this task
1. Change directory to the installation root of the Web server.
   For example, this is /opt/IBMHTTPD on a Solaris machine.
2. Locate the file named version.signature.
3. At a system command prompt in that directory, enter:
   ```
   more version.signature
   ```

Results

The version of IBM HTTP Server is shown.

Manually editing the plug-in configuration

If your use of the WebSphere Application Server product requires a plug-in configuration more advanced than what the default plug-in configuration provides, you can edit the plug-in configuration file manually.

Steps for this task

1. Find the plug-in configuration files. The working or active versions of the HTTP plugin-cfg.xml file reside in the directory *install_root*/config.
2. Manually edit the plug-in configuration files. Open an editor on a file, change the plug-in settings as needed, and save the file.

   CAUTION: If the plug-in configuration is regenerated, your edits to a HTTP plug-in configuration file are overwritten.

   This does not mean, however, that you should never manually edit a configuration file. Regenerating the plug-in configuration cannot guarantee a correct configuration file for advanced configuration scenarios. If you see strange behavior after triggering a regeneration of the plug-in configuration, consider manually editing the configuration.

plugin-cfg.xml file

The plugin-cfg.xml file includes the following elements and attributes:

Config (exactly one)

This element starts the WebSphere HTTP plug-in configuration file. It contains all other elements and attributes of the configuration.

This section resembles the following in the file:

```
<Config IgnoreDNSFailures="true" RefreshInterval="240">
```

IgnoreDNSFailures (zero or one attribute for each Config)
Specifies whether the plug-in ignores DNS failures within a configuration when starting. 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. The server is marked *unavailble* 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. The default value is `false`, meaning DNS failures cause the Web server not to start.

Refresh interval (zero or one attribute for each Config)
The time interval (in seconds) at which the plug-in should check the configuration file to see if updates or changes have occured. 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 is preferable. In production, a higher value then 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 config reloads successfully. 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.

Log (zero or one element for each Config)

The log describes the location and level of log messages that should be written by the plug-in. If a Log is not specified within the Config, then in some cases log messages will be written to the Web server error log.

This section resembles the following in the file, though the section is split here for publication:

```
<Log LogLevel="Error"
     Name="/opt/WebSphere/AppServer50/logs/http_plugin.log"/>
```

Name (exactly one attribute for each Log)

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

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

LogLevel (zero or one attribute for each Log)

The level of detail of the log messages that the plug-in should write to the log. Values for this attribute are one of the following:

- Trace
- Warn
- Error

If a LogLevel is not specified with the Log, then the default value is Error.

Trace allows you to see the steps in the request process in detail. Warn and Error means that only information about abnormal request processing will be logged.

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

ServerCluster (one or more elements for each Config)

A group of servers that are generally configured to service the same types of requests.

In the simplest case, the cluster contains only one server definition. In the case in which more than one server is defined, the plug-in will load balance across the defined servers using either a Round Robin or a Random algorithm. The default is Round Robin.

This section resembles the following in the file:

```
<ServerCluster name="Servers">
<ClusterAddress name="ClusterAddr">
<Transport Hostname="192.168.1.2" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.2" Port="9443" Protocol="HTTPS">
<Property name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</ClusterAddress>
<Server name="Server1">
<Transport Hostname="192.168.1.3" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.3" Port="9443" Protocol="HTTPS">
<Property name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Server>
<Server name="Server2">
```

```
<Transport Hostname="192.168.1.4" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.4" Port="9443" Protocol="HTTPS">
<Property name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Server>
<Server name="Server3">
<Transport Hostname="192.168.1.5" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.5" Port="9443" Protocol="HTTPS">
<Property name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Server>
<PrimaryServers>
<Server name="Server1"/>
<Server name="Server2"/>
</PrimaryServers>
<BackupServers>
<Server name="Server3"/>
</BackupServers>
</ServerCluster>
```

Name (exactly one attribute for each ServerCluster)
> The logical or administrative name to be used for this group of
> servers.

LoadBalance (zero or one attribute for each ServerCluster)
> The default load balancing type is Round Robin.
>
> The Round Robin implementation has a random starting point.
> This means that the first server picked will be done so randomly
> and then round robin will be used from that point forward. This is
> so that in multiple process based Web servers all of the processes
> don't start up by sending the first request to the same application
> server.

RetryInterval (zero or one attribute for each ServerCluster)
> An integer specifying the length of time that should elapse from
> the time that a server is marked down to the time that the plug-in
> will retry a connection. The default is 60 seconds.

RemoveSpecialHeaders (zero or one attribute for each ServerCluster)
> The plug-in adds special headers to the request before it is
> forwarded to the application server. These headers store
> information about the request that will need to be used by the
> application. By default the plug-in will remove these headers from
> incoming requests before adding the headers it is supposed to add.
>
> The value can be `true` or `false`. Setting the attribute to false
> introduces a potential security exposure by not removing headers
> from incoming requests.

CloneSeparatorChange (zero or one attribute for each ServerCluster)
> Some pervasive devices cannot handle the colon character (:) used
> to separate clone IDs in conjunction with session affinity. This
> attribute for the server group tells the plug-in to expect the plus
> character (+) as the clone separator. You must change application
> server configurations so that an application server separates clone
> IDs with the plus character as well.
>
> The value can be `true` or `false`.

PostSizeLimit (zero or one attribute for each ServerCluster)
> 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. If a request is received that is greater than this size, the plug-in fails the request. The default value is 10 million bytes.

Server (one or more elements for each ServerCluster)
A WebSphere Application Server instance that is configured to handle requests routed to it given the routing rules of the plug-in configuration. The Server should correspond to an application server running on either the local machine or a remote machine.

Name (exactly one attribute for each Server)
The administrative or logical name for the server.

CloneID (zero or one attribute for each Server)
If this unique ID is present in the HTTP Cookie header of a request (or the URL if using URL rewriting), the plug-in routes the request to this particular server, provided all other routing rules are met. If a CloneID is not specified in the Server, then Session Affinity is not enabled for this server.

This attribute is used in conjunction with Session Affinity. When this attribute is set, the plug-in checks the incoming cookie header or URL for jsessionid. If the jsessionid is found then the plug-in looks for one or more clone IDs. If clone IDs are found and a match is made to this attribute then the request is sent to this server rather than load balanced across the cluster.

If you are not using Session Affinity then it is best to remove these clone IDs from the configuration because there is added request processing in the plug-in when these are set. If clone IDs are not in the plug-in then it is assumed that session affinity is not on and the request is load balanced across the cluster.

WaitForContinue (zero or one attribute for each Server)
Specifies whether to use the HTTP 1.1 100 Continue support before sending the request content to the application server. Possible attribute values are `true` or `false`. The default value is false; the plug-in does not wait for the 100 Continue response from the application server before sending the request content because it is a performance hit.

Enable this function (set to true) when configuring the plug-in to work with certain types of proxy firewalls.

LoadBalanceWeight (zero or one attribute for each Server)
The weight associated with this server when the plug-in does weighted round robin load balancing. The algorithm for this attribute decrements all weights within the server cluster until all weights reach zero. Once a particular server's weight reaches zero, no more requests are routed to that server until all servers in the cluster have a weight of zero. After all servers reach zero, the weights for all servers in the cluster are reset and the algorithm starts over.

ConnectTimeout (zero or one attribute for each Server)
The ConnectTimeout attribute of a Server element allows

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 if the port is available or unavailable.

If no ConnectTimeout value is specified, the plug-in performs a blocking connect in which the plug-in sits until an operating system times out (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 sucessful connection. If a connection does not occur after that time interval, the plug-in marks the server *unavailable* and fails over to one of the other servers defined in the cluster.

Transport (one or more elements for each Server)
The transport for reading and writing requests to a particular WebSphere application server instance. The transport provides the information needed to determine the location of the application server to which the request will be sent. If the Server has multiple transports defined to use the same protocol, the first one will be used.

It is possible to configure the Server to have one non-secure transport and one that uses SSL. In this configuration, a match of the incoming request protocol will be performed to determine the appropriate transport to use to send the request to the application server.

Hostname (exactly one attribute for each Transport)
The host name or IP address of the machine on which the WebSphere application server instance is running.

Port (exactly one attribute for each Transport)
The port on which the WebSphere application server instance is listening.

Protocol (exactly one attribute for each Transport)
The protocol to use when communicating over this transport — either HTTP or HTTPS.

Property (zero, one, or more elements for each Transport)
When the Protocol of the Transport is set to HTTPS, use this element to supply the various initialization parameters, such as keyring and stashfile.

Name (exactly one attribute for each Property)
The name of the Property being defined. Supported names recognized by the transport are keyring, stashfile, and password.

Value (exactly one attribute for each Property)
The value of the Property being defined.

ClusterAddress (zero or one element for each ServerCluster)
A ClusterAddress is like a Server element in that you can specify the same attributes and elements as for a Server element. The difference is that you can only define one of them within a

ServerCluster. Use a ClusterAddress when you do not want the plug-in to perform any type of load balancing because you already have some type of load balancer in between the plug-in and the application server.

If a request comes in that does not have affinity established, the plug-in routes it to the ClusterAddress, if defined. If affinity has been established, then the plug-in routes the request directly to the clone, bypassing the ClusterAddress entirely. If no ClusterAddress is defined for the ServerCluster, then the plug-in load balances across the PrimaryServers list.

PrimaryServers (zero or one element for each ServerCluster)
Lists defined servers to which the plug-in routes requests for this cluster. If a list of PrimaryServers is not specified, the plug-in routes requests to servers defined for the ServerCluster.

BackupServers (zero or one element for each ServerCluster)
Lists servers to which requests should be sent to if all servers specified in the PrimaryServers list are unavailable. The plug-in does not load balance across the BackupServers list but traverses the list in order until no servers are left in the list or until a request is successfully sent and a response received from an application server.

VirtualHostGroup (zero, one, or more elements for each Config)
A group of virtual host names that will be specified in the HTTP Host header. Enables you to group virtual host definitions together that are configured to handle similar types of requests.

This section resembles the following in the file:
```
<VirtualHostGroup name="Hosts">
<VirtualHost name="www.x.com"/>
<VirtualHost name="www.x.com:443"/>
<VirtualHost name="*:8080"/>
<VirtualHost name="www.x.com:*"/>
<VirtualHost name="*:*"/>
</VirtualHostGroup>
```

Name (exactly one attribute for each VirtualHostGroup)
The logical or administrative name to be used for this group of virtual hosts.

VirtualHost (one or more elements for each VirtualHost)
The name used for a virtual or real machine used to determine if incoming requests should be handled by WebSphere Application Server or not. Use this element to specify host names that will be in the HTTP Host header which should be seen for requests that need to be handled by the application server. You can specify specific host names and ports that incoming requests will have or specify a * for either the host name, port, or both.

Name (exactly one attribute for each VirtualHost)
The actual name that should be specified in the HTTP Host header in order to match successfully with this VirtualHost.

The value is a host name or IP address and port combination, separated by a colon.

You can configure the plug-in to route requests to the application server based on the incoming HTTP Host header and port for the request. The Name attribute specifies what those combinations are.

You can use a wildcard for this attribute. The only acceptable solutions are either a * for the host name, a * for the port, or a * for both. A * for both means that any request will match this rule. If no port is specified in the definition the default HTTP port of 80 is assumed.

UriGroup (zero, one or more elements for each Config)
A group of URIs that will be specified on the HTTP request line. The same application server must be able to handle the URIs. The route will compare the incoming URI with the URIs in the group to determine if the application server will handle the request.

This section resembles the following in the file:

```
<UriGroup name="Uris">
<Uri name="/servlet/snoop"/>
<Uri name="/webapp/*"/>
<Uri name="*.jsp"/>
</UriGroup>
```

Name (exactly one attribute for each UriGroup)
The logical or administrative name for this group of URIs.

Uri (one or more elements for each UriGroup)
The virtual path to the resource that will be serviced by WebSphere Application Server. Each URI specifies the incoming URLs that need to be handled by the application server. You can use a wildcard in these definitions.

Name (exactly one attribute for each Uri)
The actual string that should be specified in the HTTP request line in order to match successfully with this URI. You can use a wildcard within the URI definition. You can specify rules such as *.jsp or /servlet/* to be handled by WebSphere Application Server.

AffinityCookie (zero or one attribute for each Uri)
The name of the cookie the plug-in should use when trying to determine if the inboud request has affinity to a particular clone. The default value is JSESSIONID.

Route (one or more elements for each Config)
A request routing rule by which the plug-in will determine if an incoming request should be handled by a WebSphere application server.

The route definition is the central element of the plug-in configuration. It specifies how the plug-in will handle requests based on certain characteristics of the request. The route definition contains the other main elements: a required ServerCluster, and either a VirtualHostGroup, UriGroup, or both.

Using the information that is defined in the VirtualHostGroup and the UriGroup for the route, the plug-in determines if the incoming request to the Web server should be sent on to the ServerCluster defined in this route.

This section resembles the following in the file:

```
<Route VirtualHostGroup="Hosts" UriGroup="Uris" ServerCluster="servers/>
```

VirtualHostGroup (zero or one attribute for each Route)
The group of virtual hosts that should be used in route

determination. The incoming host header and server port are matched to determine if this request should be handled by the application server.

It is possible to omit this from the route definition. If it is not present then every request will match during the virtual host match portion of route determination.

UriGroup (zero or one attribute for each Route)
The group of URIs to use for determining the route. The incoming URI for the request is matched to the defined URIs in this group to determine if this request should be handled by the application server.

It is possible to omit this from the route definition. If it is not present than every request will match during the URI match portion of route determination.

ServerCluster (exactly one attribute for each Route)
The cluster to which to send request that successfully match the route.

The cluster that should be used to handle this request. If both the URI and the virtual host matching is successful for this route then the request is sent to one of the servers defined within this cluster.

Situations requiring manual editing of the plug-in configuration

The following situations require manual editing of the plugin-cfg.xml file:

- If the Web server and plugin-cfg.xml file are installed on a separate remote system, you must change the paths in plugin-cfg.xml if:
  - The plug-in was generated on a Win32 system and needs to be copied to a remote UNIX system with httpServer and a WebSphere Application Server Version 5 plug-in.
  - The plug-in was generated on a UNIX system and needs to be copied to a remote Win32 system with httpServer and a WebSphere Application Server Version 5 plug-in.
  - The plug-in was generated on one UNIX distribution and needs to be copied to a remote UNIX system that is a different distribution. For example, the plug-in was generated on a system having a single-server (base) or Network Deployment installation on AIX in the default path, and the remote httpServer and plug-in are installed on a Solaris or Linux distribution with the plug-in installed in the default location.
- In a Network Deployment environment, if the single-server (base) product, Network Deployment product, plug-in, and HttpServer are all on the same machine, you must change the paths in plugin-cfg.xml if:
  - The plugin-cfg.xml file is located in *install_root*/DeploymentManager/config/ and not *install_root*/AppServer/config/. During installation of the base product, you must copy the plugin-cfg.xml file to *install_root*/AppServer/config/ because the plug-in and modifications to the HttpServer configuration files are looking for plugin-cfg.xml in *install_root*/AppServer/config.
  - The paths in the plugin-cfg.xml file specify *install_root*/DeploymentManager/... These paths exist because the deployment manager runs the plugin-update procedure. You must change the paths to

install_root/AppServer/config. *install_root*/AppServer is the default path for the base product, if the base product is installed in a different location, change the paths to that location.

- If the single-server (base) or Network Deployment product, or the plug-in, are installed in a non-default location, you must change the paths in plugin-cfg.xml. The plug-in generator assumes that the plug-in path is the same as the base product's path structure. For example, if you install the base product in c:\myApps\WebSphere\AppServer50 and install the plug-in on a remote Win32 system in c:\WebSphere\AppServer50, you must generate the plug-in using the administrative console on the base product and then edit the plug-in at c:\myApps\WebSphere\AppServer50\config. You must change all of the c:\myApps\WebSphere\AppServer50\... path structures to c:\WebSphere\AppServer50\...

Regenerating Web server plug-in configurations

At times, you might need to instruct the WebSphere Application Server plug-in to regenerate its configuration. You should regenerate the plug-in configuration after, for example, installing or removing an enterprise application, adding or removing servlets and mappings from a particular application, or changing the configuration for the plug-in, a virtual host or a transport. Failure to regenerate the plug-in after introducing a new application likely results in a *404 File Not Found* error when a user tries to access the new Web application.

CAUTION: Regenerating the plug-in configuration can overwrite manual configuration changes that you might want to preserve instead. Before performing this task, understand its implications as described in the article about configuring Web server plug-ins.

To regenerate the plug-in configuration, use the Update Web Server Plugin Configuration page.

Steps for this task
1. Go to the Update Web Server Plugin page. Click **Environment > Update Web Server Plugin** in the console navigation tree.
2. Click **OK**.

Or, to regenerate the plug-in configuration, run the following script:
`install_root`/bin/GenPluginCfg.sh|bat

You might need to stop the server and then start the server again to regenerate the plug-in configuration.

Results

Regenerating the configuration might take a while to complete. After it finishes, all objects in the administrative cell use their newest settings, of which the Web server is now aware. Whether triggered manually or occurring automatically, plug-in regeneration requires about 30 to 60 seconds to complete, when the application server is on the same physical machine (node) as the Web server. In other cases, it takes more time.

The delay is important because it determines how soon the new plug-in configuration takes effect. Suppose you add a new served path for a servlet, then

regenerate the plug-in configurations. The regeneration requires 40 seconds, after which a user should be able to access the servlet by the new served path.

For an HTTP plug-in, the length of the delay is determined by the Refresh Interval attribute of the Config element in the plugin-cfg.xml file. The plug-in polls the disk at this interval to see whether the configuration has changed. The default interval is 60 seconds. To regenerate the plug-in configurations requires twice the refresh interval.

In a development environment in which you are frequently changing settings in the administrative console, it is recommended that you set the refresh interval to 3 to 5 seconds.

In a production environment, set a longer refresh interval, perhaps as long as 30 minutes, depending on the frequency of changes.

Usage scenario

After using the administrative console to make configuration changes that involve the served paths of Web applications, manually trigger the regeneration of the plug-in configuration (or manually edit the file if that is what you have been doing), the plug-in configuration regenerates. The Web server is aware of the new Web application configuration. If a user requests the servlet using the path specified by the new Web resource, the request should be successful.

Plug-ins: Resources for learning

Use the following links to find 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.

View links to additional information about:
- Programming model and decisions
- Programming instructions and examples

Programming model and decisions

- **IBM WebSphere Application Server V4.0: HTTP Server Plug-in** (ftp://vadd1:ua33pcww@207.25.253.53/1/wsdd/pdf/presents/WS40ST08.pdf)

- **Best Practice: WebSphere Plug-in Configuration Regeneration** (http://www7b.boulder.ibm.com/webapp/dd/transform.wss?URL =/wsdd/library/bestpractices/plug_in_configuration_regeneration.xml &xslURL=/wsdd/xsl/bestpractice.xsl)

- **WebSphere Application Server V4.0 and V4.0.1 for zOS and OS/390: Configuring Web Applications** (ftp://ftp.software.ibm.com/software/mktsupport/techdocs /was4_configuring_web_apps.pdf)

- 🖥 **Best Practice: Configuring Web Applications on WebSphere Application Server for z/OS and OS/390** (http://www7b.software.ibm.com/webapp/dd /transform.wss?URL=%2Fwsdd%2Flibrary%2Fbestpractices%2Fws_zos390 _config_webapps.xml &xslURL=%2Fwsdd%2Fxsl%2Fbestpractice.xsl)

Programming instructions and examples

- ✔ **IBM HTTP Server documentation** (http://www-3.ibm.com/software/webservers/httpservers/library.html)

- ✔ **IBM WebSphere Administration** (http://www.mcgraw-hill.co.uk/html/0072223154.html)

- ✔ **WebSphere Application Server education** (http://www.ibm.com/software/webservers/learn/)

- ✔ **Listing of all IBM WebSphere Application Server Redbooks** (http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere)

- ✔ **Redbook on IBM WebSphere V4.0 Advanced Edition Scalability and Availability** (http://publib-b.boulder.ibm.com/Redbooks.nsf /9445fa5b416f6e32852569ae006bb65f /ff31072025dcf5de85256aca00781918?OpenDocument&Highlight=0,plug-in)

Chapter 3. Configuring the cell-wide environment

To assist in handling requests among Web applications, Web containers, and application servers, you can configure cell-wide settings for virtual hosts, variables and shared libraries.

Steps for this task

1. Configure virtual hosts.
2. Configure variables.
3. If your deployed applications will use shared library files, define the shared library files needed.

Virtual hosts

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

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 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 it can be created, but not started or stopped. For many users, virtual host creation is unnecessary because the default_host is provided.

Why and when to use virtual hosting

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 whose Internet sites it would like to host on the same machine. The ISP would like to keep the two sites isolated from one another, despite their sharing a machine. The ISP could associate the resources of the first company with VirtualHost1 and the resources of the second company with VirtualHost2.

Now suppose both company's sites offer the same servlet. Each site has its own instances of the servlet, which are unaware of the other site's instances. If the

17

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 will not be routed there.

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, even though VirtualHost2 is refusing to fill requests for the same servlet.

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

When you request a resource, WebSphere Application Server tries to map the request to an alias of a defined virtual host.

Mappings are case insensitive, but the match must be alphabetically 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` and to `http://Www.Myhost.Com/Myservlet`. But it does not map successfully to `http://myhost/myservlet` or to `http://myhost:9876/myservlet`.

You can use wildcard entries for aliases by port and specify that all valid hostname 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 used to issue the request. A message states that the virtual host could not be found.

Configuring virtual hosts

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

Steps for this task
1. Create a virtual host using the Virtual Hosts page of the administrative console. Click **Environment > Virtual Hosts** from the navigation tree of the console, click **New** and, on the settings page for a virtual host 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. Determine whether you need a virtual host alias for each HTTP transport port. 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, with one Web container in each application server.

   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, or for some reason the virtual host does not contain the usual entry for port 9080.

   - You have created multiple application servers (either standalone or in a cluster) that are using the same virtual host. Because each server must be listening on a different HTTP transport port, you need a virtual host alias for each one's transport port.

   If you determine that you need one or more virtual host aliases, on the HTTP Transports page, note the **Port** values, such as 9080 or 9082.

3. If necessary, create a virtual host alias for each HTTP transport port.

   From the Virtual Hosts page, click on your virtual host and, on the settings page for a virtual host, click **Host Aliases**. For each virtual host alias that you need, on the Host Aliases page, click **New**; then, on the settings page for a virtual host alias, specify a host name and 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 Types page, click **New**; then, on the settings page for a MIME type, 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).

Virtual host collection

Use this page to manage virtual hosts.

To view this administrative console page, click **Environment > Virtual Hosts**.

Name
Specifies a logical name used for configuring Web applications to a particular host name. The default virtual host is suitable for most simple configurations.

A virtual host configuration 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.

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 used for configuring Web applications to a particular host name. The default virtual host is suitable for most simple configurations.

Data type   String
Default   default_host

Host alias collection

Use this page to manage hostname aliases defined for a virtual host. An alias is the DNS host name and port number used by a client 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, JSP, or HTML page). For example, a host alias name might be `myhost` in a DNS name of `myhost:8080`.

Port: Specifies the port for which the Web server has been configured to accept client requests. For example, a port might be *8080* in a DNS name of `myhost:8080`. A URL would refer to this DNS as follows: 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, DNS host name with domain name suffix, or just the DNS host name, used by a client to request a Web application resource (such as a servlet, JSP file, or HTML page).

For example, the host alias name might be `myhost`. In such a case, the DNS name might be `myhost:8080`, where *8080* is the port. A URL would refer to this DNS as follows: http://*myhost:8080*/servlet/snoop.

For existing instances, the default reflects the value specified when the product was initially set up. For new instances, the default might be *, meaning any value allowed, or there might be no value specified.

Data type   String
Default   *

Port: Specifies the port for which the Web server has been configured to accept client requests. Specify a port value in conjunction with the host name.

The default reflects the value specified when the product was initially set up. The default might be 80, 9080 or some similar value.

Data type   Integer
Default   80

MIME type collection

Use this page to view and configure MIME object types and their file name extensions.

The list shows a collection of MIME type extension mappings defined for a virtual host. If MIME entries are not specified at the Web module level, these MIME entries apply.

To view this administrative console page, click **Environment > Virtual Hosts >** *virtual_host_name* **> MIME Types**.

MIME Type: Specifies a MIME type to which the type's file name extensions map. The type might be text, image, or application. An example value for MIME type is `text/html`.

Extensions: Lists file extensions that map to the MIME type. Example file extensions for a `text/html` MIME type include `.htm`, `.html`, and `.txt`.

MIME type settings: Use this page to configure a 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 to which the type's file name extensions map.

The type might be text, image, or application. An example value for MIME type is `text/html`. A default value is given only if you are viewing the configuration for an existing instance.

Data type   String

Extensions: Lists file extensions that map to the MIME type.

Example file extensions for a `text/html` MIME type include `.htm`, `.html`, and `.txt`. A default value is given only if you are viewing the configuration for an existing instance.

Data type   String

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 be used in place of that name wherever the variable name is located within the configuration files.

Variables have a scope, which is the range of locations in the WebSphere Application Server network where the variable is applicable. A variable with a cell-wide scope applies across the entire WebSphere Application Server cell. A variable with node-level scope applies only on the node for which it is defined. If a node-level variable has the same name as a cell-wide variable, the node-level variable value takes precedence. A server variable only applies to the one server process, and takes precedence over any wider scoped variable with the same name.

When you use variables in configuration values such as file system path settings, the following syntax refers to the variable, using the variable's name:

${*variable_name*}

If the value of a variable contains a reference to another variable, the value of the variable is computed by substituting the value of the referenced variable recursively. For example:

Variable name	Variable value
ROOT_DIR	/
HOME_DIR	${ROOT_DIR}home
USER_DIR	${HOME_DIR}/myuserdir

In this example, the variable reference ${USER_DIR} resolves to the value /home/myuserdir.

Variables are often useful where configured paths cannot contain *and*.

Configuring variables

You might want to define a WebSphere Application Server variable to provide a parameter for a value in the system. After you define the name and value for a variable, the value is used in place of that name wherever the variable name is located within the configuration files. The scope of a variable can be cell-wide, node-wide, or applicable to only one server process.

Steps for this task

1. Click **Environment > Manage WebSphere Variables** in the console navigation tree. On the WebSphere Variables page, click **New**.
2. On the Variable page, specify a name and value for the variable. So other people can understand what the variable is used for, also specify a description for the variable. Then click **OK**.
3. Verify that the variable is shown in the list of variables.
4. Save your configuration.
5. **(Optional)** To have the configuration take effect, stop the server and then start the server again.

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 > Manage WebSphere Variables**.

For information on a variable, click on the variable and read the value entered for the **Description** field.

Name

Specifies the symbolic name representing a physical path or URL root.

Value

Specifies the absolute path that the symbolic name represents.

Scope

Specifies whether the variable applies across a cell, node, or server.

Variable settings

Use this page to define the name and value of a WebSphere substitution variable.

To view this administrative console page, click **Environment > Manage WebSphere Variables >** *WebSphere_variable_name*.

Name: Specifies the symbolic name representing a physical path or URL root.

This name is substituted into path names configured in the system. For example, "CONFIG_ROOT" could be the symbolic name representing the path "C:\WebSphere\AppServer\Config" on a Windows system.

Data type   String

Value: Specifies the absolute path that the symbolic name represents.

For example, the absolute path "C:\WebSphere\AppServer\Config" on a Windows system could be represented by the symbolic name "CONFIG_ROOT."

Data type   String

Description: Provides additional documentation of the purpose for your variable.

Data type   String

Shared library files

Shared library files in WebSphere Application Server consist of a symbolic name, a Java classpath, 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 classloader. 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 classloader.

A separate classloader is used for shared libraries that are associated with an application server. This classloader is the parent of the application classloader, and the WebSphere Application Server extensions classloader is its parent. Shared libraries that are associated with an application are loaded by the application classloader.

Managing shared libraries

If your deployed applications will use library files, set variables for those library files and then associate the files to the applications. You use the Shared Libraries page to define new shared library files to the system and remove them.

Steps for this task

1. Identify library files and their classpaths.

a. Click **Environment > Shared Libraries** in the console navigation tree to access the Shared Libraries page.

b. **(Optional)** Change the scope of the collection table to see what shared libraries are in a cell, node, or server. Select the cell, a node, or a server and click **Apply**.

c. Click **New** and, on the settings page for a shared library, specify the name, classpath and any other variables for the library file that you determine are needed. Then, click **Apply**.

d. Repeat step 3 until you define a shared library instance for each library file that your applications need.

2. After you install an application that needs to use one or more shared libraries, associate the shared libraries to the application.

a. Click **Applications > Enterprise Applications** in the console navigation tree. Click on the installed application that will use the shared libraries. Then, click **Libraries** to access the Library Ref page.

b. Click **Add** and, on the settings page for a library reference, specify variables for the library reference as needed. Then, click **Apply**.

c. Repeat step 2 until you define a library reference instance for each library file that your application needs.

3. **(Optional)** Remove a library file from the collection of shared library files. On the Shared Libraries page, place a checkmark beside the library you want removed from the collection and click **Delete**.

Shared library collection

Use this page to define a list of shared library files for use by deployed applications.

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 specify a library file usable by 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 used to locate native library files for the shared library support; for example, .dll or .so files

Data type	String
Units	Class path

Library reference collection

Use this page to view and manage library references that define how global libraries are used. You can use this page to, for example, associate shared library files with a deployed application.

To view this administrative console page, click **Applications > Enterprise Applications >** *application_name* **> Libraries**.

Library Name

Specifies a name for the library reference.

Library reference settings

Use this page to define library references, which specify how global libraries are used.

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.

Library Name: Specifies the name of the shared library to use for the library reference.

Data type	String

Shared Classloader: Specifies whether the classloader for the library will be shared.

Data type	Boolean
Default	true

Environment: Resources for learning

Use the following links to find relevant supplemental information about configuring the WebSphere Application Server cell-wide 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.

View links to additional information about:
- Planning, business scenarios, and IT architecture
- Programming model and decisions
- Programming instructions and examples
- Administration

Planning, business scenarios, and IT architecture

- **WebSphere Application Server V5.0: Architecture and Overview**
  (http://developerworks.cybercentral.com/ibm0502/amt/ibmpresentations
  /683_1.pdf)

Programming model and decisions

- **Redbook on WebSphere Application Server V4 for Linux,
  Implementation and Deployment Guide** (http://publib-
  b.boulder.ibm.com/Redbooks.nsf/RedpaperAbstracts /REDP0405.html)

Programming instructions and examples

- **WebSphere Application Server V4.0 and V4.0.1 for zOS and OS/390:
  Configuring Web Applications**
  (ftp://ftp.software.ibm.com/software/mktsupport/techdocs
  /was4_configuring_web_apps.pdf)

- **Best Practice: Configuring Web Applications on WebSphere Application
  Server for z/OS and OS/390** (http://www7b.software.ibm.com/webapp
  /dd/transform.wss?URL=%2Fwsdd%2Flibrary%2Fbestpractices%2
  Fws_zos390_config_webapps.xml&xslURL=%2Fwsdd%2Fxsl%2 Fbestpractice.xsl)

- **WebSphere Application Server education**
  (http://www.ibm.com/software/webservers/learn/)

Administration

- **IBM WebSphere Administration** (http://www.mcgraw-
  hill.co.uk/html/0072223154.html)

- **Listing of all IBM WebSphere Application Server Redbooks**
  (http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals /WebSphere)

- **IBM WebSphere V4.0 Advanced Edition Handbook**
  (http://www.redbooks.ibm.com/abstracts/sg246176.html)