

IBM WebSphere Application Server for z/OS V5.0.1



Environment

Note

Before using this information, be sure to read the general information under “Notices” on page 45.

Compilation date: September 22, 2003

© Copyright International Business Machines Corporation 2002. 2003. All rights reserved.

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

Contents

How to send your comments v

Chapter 1. Welcome to Environment . . . 1

Chapter 2. Configuring Web server
plug-ins 3

plugin-cfg.xml file 4

Web server plug-ins 13

Installing the WebSphere HTTP Plug-in for z/OS. . . 13

Private headers 15

Installing a distributed platform Web server plug-in 16

Supported distributed platform Web server

plug-in configurations 17

Checking your IBM HTTP Server version 18

Manually editing the plug-in configuration 18

Situations requiring manual editing of the

plug-in configuration 19

Regenerating Web server plug-in configurations . . . 21

Installing a Global Security Kit for a distributed

platform Web server plug-in. 23

Gskit install images files 26

Plug-ins: Resources for learning 26

Chapter 3. Configuring the cell-wide
environment 29

Virtual hosts 29

Why and when to use virtual hosting 29

The default virtual host (default_host) 30

How requests map to virtual host aliases 30

Configuring virtual hosts. 31

Virtual host collection 32

Variables 36

Configuring variables 36

WebSphere variables collection 37

Shared library files 38

Managing shared libraries 39

Shared library collection 40

Library reference collection 41

Environment: Resources for learning 42

Notices 45

Trademarks and service marks 47

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. 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. For tips on tuning your Web server see Web server tuning parameters.

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.

A WebSphere Web server plug-in can be used to forward requests for Web applications from a supported Web server to a WebSphere Application Server for z/OS Web container. A WebSphere Application Server internal HTTP transport provides the connection between the Web server plug-in and the Web container.

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:

A Web server 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.

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.

IMW0093E Method POST is disabled on this server

2. If you encounter problems starting your Web server, check the `http_plugin.log` plug-in 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. Install the plug-in to a specific location.
4. Install either the WebSphere HTTP Plug-in for z/OS, or a WebSphere plug-in for a Web server that is running on a distributed platform..
5. Check the version of your IBM HTTP Server installation.

6. Manually edit the plug-in configuration file.
7. 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.
8. Tune your Web server with Web server tuning parameters.

Related tasks

Configuring transports

Changing the HTTP plug-in 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 *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. 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 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 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 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.

ASDisableNagle (zero or one attribute for each Config)

Specifies whether the user wants to disable nagle algorithm for the connection between the plug-in and the application server. By default, nagle algorithm is enabled.

The value can be true or false.

IISDisableNagle (zero or one attribute for each Config)

The nagle algorithm is enabled by default on Microsoft Internet Informations Services (IIS). This attribute can be used to disable the nagle algorithm.

The value can be true or false.

ResponseChunkSize (zero or one attribute for each Config)

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

The ResponseChunkSize attribute allows the users to specify the maximum chunk size to use when reading the response body. For example, `<Config ResponseChunkSize="N">`, where N equals the chunk size in kilobytes.

If the content length of the response body is unknown, a buffer size of N kilobytes is allocated and the body is read in N kilobyte size chunks, until the entire body is read. If the content length is known, then a buffer size of either content length or N (whichever is less) is used to read the response body.

The default chunk size is 64k.

Log (zero or one element for each Config)

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 config, then, in some cases, log messages are written to the Web server error log.

This section resembles the following in the file:

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

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.

Note: The time the information was written to the log file and the process ID will be appended to the file name specified on this element.

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 spacefile system 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>
```

Note: If you are using the WebSphere HTTP Plug-in for z/OS, the Property Name=keyring and the Property Name=stashfile elements included here will be ignored if they are included in the plugin-cfg.xml file for that plug-in. The WebSphere HTTP Plug-in for z/OS uses the SSL setup specified in the hosting HTTP Server's httpd.conf file and does not look for these elements in the plugin-cfg.xml file.

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 **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 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 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 successful 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.

ExtendedHandshake (zero or one attribute for each Server)

The ExtendedHandshake attribute is used when a proxy firewall is between the plug-in and the application server. In such a case, the plug-in is not failing over, as expected.

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.

The plug-in performs some handshaking with the application server to ensure that it is started before sending the request. This enables the plug-in to failover in the event the application server is down.

The value can be true or false.

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.

By default, the MaxConnections is set to zero. If the value is zero, then, there is no limit to the number of pending connections to the application servers.

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 password, 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.

Note: password is the only name that can be specified for the WebSphere HTTP Plug-in for z/OS. keyring, and stashfile, if specified, will be ignored.

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 specify 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. When you assemble your application, if you specify **File Serving Enabled** then only a wildcard URI is generated for the Web application, regardless of any explicit servlet mappings. If you specify **Serve servlets by classname** then a URI having <Uri Name="Web_application_URI/servlet/*"> is generated.

AffinityCookie (zero or one attribute for each Uri)

The name of the cookie the plug-in should use when trying to determine if the inbound request has session affinity to a particular clone. The default value is **JSESSIONID**. (See the description of the CloneID attribute for additional session affinity information.)

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.

RequestMetrics (zero or one element for each Config)

This element is used to determine if request metrics is enabled, and how to filter the requests based on the Internet protocol (IP) and Uniform Resource Identifiers (URI) filters when request metrics is enabled.

This section resembles the following in the file:

```
<RequestMetrics armEnabled="false" newBehavior="false"
  rmEnabled="false" traceLevel="PERF_DEBUG">
```

armEnabled (zero or one attribute for RequestMetrics)

This attribute is currently ignored by the plug-in. It is for the future usage.

newBehavior (zero or one attribute for RequestMetrics)

This attribute is currently ignored by the plug-in. It is for the future usage.

rmEnabled (exactly one attribute for RequestMetrics)

This attribute indicates whether or not the request metrics is enabled in the plug-in. When it is true, the plug-in request metrics will look at the filters and log the request trace record in the plug-in log file. This is performed if a request passes the filters.

traceLevel (exactly one attribute for RequestMetrics)

When rmEnabled is true, this attribute indicates how much information is logged. When it is NONE, there is no request logging. When it is not NONE, the request response time (and other request information) is logged when the request is done.

filters (zero, one, or two attributes for RequestMetrics)

When rmEnabled is true, the filters control which requests are traced.

enable (exactly one attribute for each filter)

When enable is true, the type of filter is on and requests must pass the filter.

type (exactly one attribute for each filter)

There are two types of filters: SOURCE_IP (for example., client IP address) and URI. For the SOURCE_IP filter type, requests are filtered based on a known IP address. You can specify a mask for an IP address using the asterisk (*). If the asterisk is used, the asterisk must always be the last character of the mask, for example 127.0.0.*, 127.0.*, 127*. For performance reasons, the pattern matches character by character, until either an asterisk is found in the filter, a mismatch occurs, or the filters are found as an exact match.

For the URI filter type, requests are filtered based on the URI of the incoming HTTP request. The rules for pattern matching are the same as matching SOURCE_IP address filters.

If both URI and client IP address filters are enabled, Request Metrics requires a match for both filter types. If neither is enabled, all requests are considered a match.

filterValues (one or multiple attribute for each filter)

The filterValues show the detailed filter information.

value (exactly one attribute for each filterValue)

Specifies the filter value for the corresponding filter type. This could be either a client IP address or a URI.

Related tasks

“Installing a distributed platform Web server plug-in” on page 16

“Installing the WebSphere HTTP Plug-in for z/OS”

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.

You can install WebSphere Application Server plug-ins for your Web servers after you finish installing the application server.

Related tasks

Chapter 2, “Configuring Web server plug-ins,” on page 3

Installing the WebSphere HTTP Plug-in for z/OS

The WebSphere HTTP Plug-in for z/OS ships as part of the WebSphere Application Server for z/OS product. To use this plug-in, you must have an IBM HTTP Server for z/OS or OS/390 Version 5.3 configured as part of a z/OS system.

After the application server, the HTTP Server and the plug-in are properly configured:

- WebSphere Application Server for z/OS can use this plug-in to perform regular plug-in functions.
 - Requests can be routed from a browser, through the HTTP Server and plug-in, to one of the application server J2EE server instances defined in the ServerGroup element in the `plugin-cfg.xml` file. (New requests are sent to randomly selected server instances, but once a session is established, requests get routed back to the server instance assigned to the original request.)
 - Private headers can be used as a mechanism for forwarding proxy information from this plug-in to a WebSphere Application Server for z/OS Web container. (This information is not otherwise included with HTTP requests.)
1. Make sure a Version 5.3 IBM HTTP Server for z/OS and OS/390 is installed on a z/OS or OS/390 system.
 2. Use FTP or another file transfer mechanism to download, in binary format, the WebSphere HTTP Plug-in for z/OS from your WebSphere Application Server for z/OS system to that HTTP Server and place it in the `webservers_install_root/bin/` directory. The `ihs390WAS50Plugin_http.so` DLL is located in the application server's `/install_root/bin/` directory.

- Using an authorized z/OS user ID, issue the following commands from an OMVS command line prompt to turn on the \hat{p} bit in the HFS where the WebSphere HTTP Plug-in for z/OS is now located:

```
chmod 777 ihs390WASPlugin_http.so
extattr +p ihs390WASPlugin_http.so
```

- Add ServerInit, Service, and ServerTerm directives to the httpd.conf configuration file of the HTTP Server:
 - The following ServerInit and ServerTerm directives indicate the entry points to the plug-in's initialization and exit routines. These routines exist as entry points `init_exit`, and `term_exit`, respectively, within the `ihs390WAS50Plugin_http.so` DLL file.

```
ServerInit /webserver_install_root/bin/
           ihs390WAS50Plugin_http.so:init_exit fully_qualified_path
ServerTerm /webserver_install_root/bin/ihs390WAS50Plugin_http.so:term_exit
```

webserver_install_root is the name of the root directory where the Web server is installed, and *fully_qualified_path* is the fully qualified path to the `plugin-cfg.xml` file.

- The following Service directive for each application that will be using the Web server plug-in. This directive indicates the entry point to the plug-in's request routine. The request routine exists as the entry point `service_exit` within the `ihs390WAS50Plugin_http.so` Dynamic Link Library (DLL) file.

```
Service /webapp_contextroot/* /webserver_install_root/bin/
        ihs390WAS50Plugin_http.so:service_exit
```

webapp_contextroot is the context root of the Web application, and *webserver_install_root* is the name of the root directory where the Web server is installed.

Notes:

- In this discussion, the ServerInit and Service directives are split for printing purposes. In the actual `httpd.conf` file, enter each of these directives on a single line.
 - The HTTP Server interprets a blank in a directive specification as a delimiter and a number sign (#) as the beginning of a comment that should be ignored. Therefore, if you need to use a blank or number sign in a directive, you must include a backslash (\) before the blank or number sign to enable the HTTP Server to correctly process the directive.
 - If a servlet sets an HTTP response code by any means, such as using methods `lastModified()` or `setStatus()`, and the client does not receive the expected response code, add the following directive to the HTTP Server configuration file:

```
ServiceSync On
```
 - If you want to use Secure-socket layer (SSL), make sure that the HTTP Server is configured for SSL. The SSL connection of the Web server plug-in uses the SSL session established by the HTTP Server. (See *z/OS HTTP Server Planning, Installing, and Using, Version 5.3, SC34-4826*, for a description of how to configure the HTTP Server for SSL.)
- Configure the plug-in. Use the `GenPluginCfg.sh` script to create your `plugin-cfg.xml` file. The resulting file will be in ASCII format. You must convert this file to EBCDIC before executing it on your z/OS system.
 - Using the administrative console, make sure the virtual host is configured with an alias for the port number used by the z/OS V5.3 HTTP Server. Specify the same port on a `<Transport Hostname>` element in the `plugin-cfg.xml` file.

7. If you intend to use private headers, using the administration console, set the TrustedProxy property for the transport to true. This setting enables the transport to trust private headers received from any supported Web server plug-in.

You can set the TrustedProxy property to true from either the Web container **Custom Properties** page or the HTTP Transport **Custom Properties** page. If you set it on the Web container **Custom Properties** page, all transports will support private headers.

Note: If you try to use private headers without adding the TrustedProxy property, they will be ignored. If the private headers are ignored, the application server might not locate the requested application.

After you add this setting, the transport(s) trusts all private headers it receives. Therefore, you must ensure that all paths to the HTTP or HTTPS transport are trusted.

8. Stop the application server and the HTTP Server and start them again.

The configuration is complete. To activate the configuration, stop and restart both the application server and the HTTP Server. If the WebSphere HTTP Plug-in for z/OS comes up when the HTTP Server starts again, you receive the following messages:

```
WebSphere HTTP Plug-in for z/OS Version 5.00 Service Level 0 is starting
WebSphere HTTP Plug-in for z/OS initializing with configuration file :
    fully_qualified_path_to_the_plugin-cfg.xml_file
WebSphere HTTP Plug-in for z/OS initialization went OK :-)
```

Related tasks

Modifying the default Web container configuration

Configuring virtual hosts

“Regenerating Web server plug-in configurations” on page 21

Related reference

Example: Setting custom properties for an HTTP transport

“plugin-cfg.xml file” on page 4

Private headers

Web server plug-ins enable the Web server to communicate requests for dynamic content, such as servlets, to the application server.

Once you configure a WebSphere plug-in for Web servers, in addition to regular plug-in functions, you can use private headers as a mechanism for forwarding proxy information from these plug-ins to the application server on a z/OS platform. This information is not otherwise included with the HTTP requests.

Private headers are implemented as a set of HTTP request header name and value pairs that the plug-ins add to the HTTP request header as it is forwarded by the Web server. The application server Web container removes this information from the header and processes it.

Private headers can include such information as the remote (client) user, the remote (client) host name, or an SSL client certificate. They conform to a naming standard so that there is no namespace collision with the architected HTTP header fields (hence the name “private”).

For example, authentication information, such as a client certificate, is normally requested by the Web server once during the establishment of an HTTP session. It

is not required again for individual requests within that session. However, a client certificate must accompany each request forwarded to the application server on a z/OS platform so that the application server can use it as needed.

Similarly, the Web server examines the TCP/IP socket connection for information about the host address of the client. The application server cannot do this examination because its socket connection is with the plug-in, not the actual client. Therefore, one of the private headers is the host address of the actual client.

Installing a distributed platform Web server plug-in

Web server plug-ins for distributed platform Web servers are no longer shipped with the WebSphere Application Server for z/OS product. Use the installation CD for WebSphere Application Server or WebSphere Application Server Network Deployment V5 for distributed platforms to install these Web server plug-ins.

Note: If you do not have a copy of Version 5 of the WebSphere Application Server or WebSphere Application Server Network Deployment for distributed platforms product, you must request a not-for-resale copy from IBM.

Once you have the application server, the Web server and the plug-in properly configured, requests can be routed from a browser, through that Web server and plug-in, to one of the application server J2EE server instances defined in the ServerGroup element in the plugin-cfg.xml file. New requests are sent to randomly selected server instances, but once a session is established, requests will get routed back to the correct HTTP or HTTPS internal transport the J2EE server Web container assigned to the original request.

WebSphere plug-ins for Web servers, in addition to regular plug-in functions, enable you to use private headers as a mechanism for forwarding proxy information from these plug-ins to the WebSphere Application Server for z/OS Web container. This information is not otherwise included with the HTTP(S) requests.

1. Make sure a supported distributed platform Web server is installed on your workstation. If you need to install a Web server, follow the instructions provided with that Web server.

Note: If you have WebSphere Application Server Network Deployment V5 or later for distributed platforms installed on your workstation, you can use the IBM HTTP Server that is provided with that product, and the Web server plug-in that is appropriate for that Web server.

2. Install the appropriate Web server plug-in. Using the installation CD for WebSphere Application Server or WebSphere Application Server Network Deployment V5 for distributed platforms, select to perform a custom installation, and then under "Web Server Plugins" select the appropriate Web server plug-in for your system.
3. Re-configure the Web server.

Once the plug-in files are installed on the Web server, update the Web server configuration file on the workstation with the location of the plug-in and plug-in configuration file. For Windows systems, also add the location of the common file, plugin_common.dll, to the PATH statement.

Specific instructions for configuring plug-ins within a Web server configuration file are contained in the documentation for each Web server.

4. Go to the WebSphere Application Server for z/OS administrative console and make sure the virtual host contains an alias for the port number used by the Web server. Specify this same port on a <Virtual Hostname> element in the plug-in plugin-cfg.xml file.
5. Configure the plug-in. Issue the GenPluginCfg.sh script from an OMVS command line prompt to create your plugin-cfg.xml file. The resulting file will be in ASCII format, which is the required format.
6. If you intend to use private headers, using the WebSphere Application Server for z/OS administrative console, set the TrustedProxy custom property for the internal transport to true. This setting enables the HTTP internal transport to trust private headers received from any WebSphere plug-in for Web servers. You can set the TrustedProxy property to true from either the Web container **Custom Properties** page or the Http Transport **Custom Properties** page. If you set it on the Web container **Custom Properties** page, all transports will support private headers.

Note: If you try to use private headers without adding the TrustedProxy property, they will be ignored. If the private headers are ignored, the application server might not locate the requested application.

After you add this setting, the HTTP or HTTPS internal transport trusts all private headers it receives. Therefore, you must ensure that all paths to the HTTP or HTTPS transport are trusted.

7. If you want to use Secure-socket layer (SSL) with this configuration, install the appropriate GSKIT installation image file on your workstation using the native install process for that platform.
8. The configuration is complete. To activate the configuration, stop and restart both the the WebSphere Application Server for z/OS and the Web server that is running on your workstation.

Related concepts

“Private headers” on page 15

Related tasks

Modifying the default Web container configuration

Configuring virtual hosts

“Installing a Global Security Kit for a distributed platform Web server plug-in” on page 23

“Manually editing the plug-in configuration” on page 18

Related reference

Example: Setting custom properties for an HTTP transport

“Plug-ins: Resources for learning” on page 26

Supported distributed platform Web server plug-in configurations

Each of the supported distributed platform WebSphere Web server plug-ins run on a number of operating systems. The following table lists the supported distributed platform Web servers and plug-in executable files for each operating system.

Operating system	Web server	Plug-in executable file
Windows 2000 and Windows NT	IBM HTTP Server for distributed platforms	mod_ibm_app_server_http.dll
	Lotus Domino	libdomino5_http.dll

	Apache	app_server_http.dll
	iPlanet (Netscape)	libns41_http.dll
	Microsoft Internet Information Server (IIS)	iisWASPlugin_http.dll
IBM AIX	IBM HTTP Server for distributed platforms	mod_ibm_app_server_http.so
	Lotus Domino	libdomino5_http.a
	Apache	mod_app_server_http.so
	iPlanet (Netscape)	libns41_http.so
HP-UX	IBM HTTP Server for distributed platforms	mod_ibm_app_server_http.sl
	Lotus Domino	libdomino5_http.sl
	Apache	mod_app_server_http.sl
	iPlanet (Netscape)	libns41_http.sl
Solaris Operating Environment	IBM HTTP Server for distributed platforms	mod_ibm_app_server_http.so
	Lotus Domino	libdomino5_http.a
	Apache	/mod_app_server_http.so
	iPlanet (Netscape)	libns41_http.so
Linux	IBM HTTP Server for distributed platforms	mod_ibm_app_server_http.so
	Apache	mod_app_server_http.so

Related tasks

“Installing a distributed platform Web server plug-in” on page 16
 Configuring high-speed external caching through the Web server

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/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
```

The version of IBM HTTP Server is shown.

Related tasks

Chapter 2, “Configuring Web server plug-ins,” on page 3

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.

1. Find the plug-in configuration files. By default, the working or active versions of the `plugin-cfg.xml` file reside in the directory `install_root/config/cellsinstall_root/config/plugin-cfg.xml`.

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.

A `plugin-cfg.xml` created using either the Update Web Server Plugin Configuration page in the administrative console, or the `GenPluginCfg.sh` script is in ASCII format. To edit a `plugin-cfg.xml` file that resides on a z/OS system, you must either:

- a. FTP the file, in binary format, to a workstation, edit it, and then FTP it, in binary format, back to your z/OS system, or
- b. Use an ASCII-to-EBCDIC conversion tool on your z/OS system to convert the `plugin-cfg.xml` file to EBCDIC format, and then issue an `oedit` command from an OMVS command line prompt to edit the file. You must be sure to convert the `plugin-cfg.xml` file back to ASCII format before attempting to execute it on your z/OS system.

CAUTION: If the plug-in configuration is regenerated, your manual edits to the 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.

3. If you edited the `plugin-cfg.xml` files on federated nodes, turn off automatic synchronization of nodes on the File Synchronization Service page to prevent the edited files from being overwritten. Because a `plugin-cfg.xml` file on a Network Deployment machine is stored in the configuration repository, the `plugin-cfg.xml` file is overwritten on all federated nodes in your network whenever the Network Deployment machine updates the configuration repository on these nodes. If you need to regenerate the `plugin-cfg.xml` file on the Network Deployment machine:
 - a. Save your edited `plugin-cfg.xml` files on the federated nodes.
 - b. Force node synchronization of each federated node.
 - c. Replace the updated `plugin-cfg.xml` file with the saved copy or merge the customized pieces of the saved copy into the new `plugin-cfg.xml` file.

Related reference

“`plugin-cfg.xml` file” on page 4

Situations requiring manual editing of the plug-in configuration

IBM WebSphere Application Server for z/OS integrates the WebSphere Application Server product and the Network Deployment product into a single package. When you see WebSphere Application Server product or Network Deployment product in reference to WebSphere Application Server for z/OS, the term product equates to function.

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` file if:

- The plug-in was generated on a Win32 system and needs to be copied to a remote UNIX system with an HTTP Server 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 an HTTP Server 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 HTTP Server 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 HTTP Server are all installed on the same node:
 - During installation of the base product, you must copy the `plugin-cfg.xml` file to the `install_root/AppServer/config/cells` directory because that is where the plug-in and the HTTP Server will look for it. (The `plugin-cfg.xml` file is initially located in the `install_root/DeploymentManager/config/cells` directory.)
 - In the `plugin-cfg.xml` file, you must change references to the `install_root/DeploymentManager/config/cells` path to `install_root/AppServer/config/cells`. The references to the `install_root/DeploymentManager/config/cells` path exist because the deployment manager runs the `plugin-update` procedure.

Note: `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.

- In a Network Deployment environment, if the single-server (base) product, Network Deployment product, plug-in, and HTTP Server are all installed on the same node, you must change references in the `plugin-cfg.xml` file to the `install_root/DeploymentManager/config/cells` path to `install_root/AppServer/config/cells`. The references to the `install_root/DeploymentManager/config/cells` path exist because the deployment manager runs the `plugin-update` procedure.
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\...`
- When the deployment manager is installed on a machine that is remote from the base WebSphere Application Server installation, one of the following solutions must be implemented in order for the `plugin-cfg.xml` file to retain the `install_root/AppServer` directory structures, and not assume those of the `install_root/Deployment Manager`, after a regeneration of the plug-in and full synchronization. The `plugin-cfg.xml` file is located in the `install_root/AppServer/config` directory.
 - **Command line:**

At a command prompt, change to the Deployment Manager/bin directory and type `GenPluginCfg -destination.root<install_root>` on the machine where Deployment Manager is installed. This creates or updates the `plugin-cfg.xml` file. This will change all of the directories in the `plugin-cfg.xml` file to the `install_root/AppServer` directories. For example, run the `GenPluginCfg -destination.root "E:\WebSphere\AppServer"` command at the deployment mgr/bin directory.

– **plugin-cfg.xml file:**

Edit the `plugin-cfg.xml` file in the `deployment manager/config/cells` directory to point to the correct directory structure for the log file, keyring, and stashfile. Perform a full sync so the `plugin-cfg.xml` file is replicated in all the WebSphere Application Server nodes. The Deployment Manager `plugin-cfg.xml` file can point to the application server directories without any conflict.

Related tasks

“Manually editing the plug-in configuration” on page 18

Related reference

“plugin-cfg.xml file” on page 4

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, you can either use the Update Web Server Plugin Configuration page in the administrative console, or run the following script:

```
install_root/bin/GenPluginCfg.sh|bat
GenPluginCfg.sh -cell.name cell_name -node.name node_name
                 -server.name server_name | a2e
```

Note:

1. This command must be entered on a single line, it is split here for display purposes.
2. The `a2e` parameter uses the `a2e` tool to convert output messages from ASCII to EBCDIC format. These messages, which indicate that the script is running and the location of the `plugin-cfg.xml` file, can then be read at the MVS console and in the output log file.

John Pfuntner’s (jpfuntne@us.ibm.com) ASCII to EBCDIC conversion tool, `a2e`, is available as freeware at <http://www.ibm.com/servers/eserver/zseries/zos/unix/bpxa1ty2.html>.

3. Both methods for regenerating the plug-in configuration will create a `plugin-cfg.xml` file in ASCII format. You must convert this file to

EBCDIC before attempting to execute it on your z/OS system. Once the file is converted to EBCDIC, you will be able to view or edit the content of this file on your z/OS system.

Note: Both methods for regenerating the plug-in configuration will create a `plugin-cfg.xml` file in ASCII format, which is the proper format for execution.

To use the Update Web Server Plug-in Configuration page in the administrative console:

1. Go to the Update Web Server Plugin page. Click **Environment > Update Web Server Plugin** in the console navigation tree.
2. Click **OK**.
3. You might need to stop the server and then start the server again before the changes to the plug-in configuration go into affect.

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 diskfile system 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.

Note:

When the deployment manager is installed on a machine that is remote from the base WebSphere Application Server installation, one of the following solutions must be implemented in order for the `plugin-cfg.xml` file to retain the `install_root/AppServer` directory structures, and not assume those of the `install_root/Deployment Manager`, after a regeneration of the plug-in and full synchronization. The `plugin-cfg.xml` file is located in the `install_root/AppServer/config` directory.

• **Command line:**

At a command prompt, change to the Deployment Manager/bin directory and type `GenPluginCfg -destination.root<install_root>` on the machine where Deployment Manager is installed. This creates or updates the `plugin-cfg.xml` file. This will change all of the directories in the

plugin-cfg.xml file to the *install_root*/AppServer directories. For example, run the GenPluginCfg -destination.root "E:\WebSphere\AppServer" command at the deployment mgr/bin directory.

- **plugin-cfg.xml file:**

Edit the deployment manager/config/cells plugin-cfg.xml file to point to the correct directory structure for the log file, keyring, and stashfile. Perform a full sync so the plugin-cfg.xml file is replicated in all the WebSphere Application Server nodes. The Deployment Manager plugin-cfg.xml file can point to the application server directories without any conflict.

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.

Related tasks

Chapter 2, "Configuring Web server plug-ins," on page 3

Installing a Global Security Kit for a distributed platform Web server plug-in

In addition to the plug-in files, there is another software package provided with WebSphere Application Server for z/OS that must be installed on the Web server that is running on your workstation. This package is called the Global Security Kit (GSKit). It helps connect this Web server to the WebSphere Application Server for z/OS product. This package is required if the Secure Sockets Layer (SSL) Transport (also known as HTTPS) is used.

There is one GSKit installation image per platform. GSKit is the same for all Web servers running on that platform.

Steps for this task:

1. On your workstation, add the GSKit installation directory to the Web server's PATH statement.
2. Using FTP or another file transfer mechanism, download, in binary format, the appropriate GSKit installation image file from the WebSphere Application Server for z/OS *install_root*/DownloadPlugins directory.

When downloading these files, remove the "_bin" suffix from the file name. It is included as a reminder that these files are in binary format. For example, when downloading the gskkm.rte_bin file for the AIX GSKit installation image residing on AIX, use gskkm.rte as the file name.

3. Using the native install process for the operating system you are running on your workstation, install this file onto the Web server. For example, DSMIT should be run on AIX, or the gsk5bas.exe, which invokes InstallShield, should be run on a Windows system.
4. If you intend to use Secure-socket layer SSL:
 - Configure the Web server for SSL support. (See your Web server documentation for a description of how to configure SSL for your specific Web server.)

For a IBM HTTP Server for distributed platforms, you must also add the following lines to the bottom of the Web server httpd.conf file:

```

LoadModule ibm_ssl_module modules/IBModuleSSL128.d11
Listen port_number
Keyfile C:\ssl\http_session\plug-inKeys.kdb
<VirtualHost virtual_host_name:port_number>
    ServerName virtual_host_name
    SSLEnable
    SSLClientAuth none
</VirtualHost>

```

These lines cause the Web server to listen on the specified port.

SSLClientAuth none indicates that you do not want to enable client authentication. If you want to use client authentication, change this line to **SSLClientAuth enable**.

This change causes the HTTP Server to send a request for a certificate to the browser. Your browser might prompt you to choose a certificate to send to the Web server for performing client authentication.

- Configure an HTTPS internal transport to listen on the port the Web server plug-in is using to redirect requests to the WebSphere Application Server for z/OS Web container. Specify this same port on a <Transport Hostname> element in the plug-in plugin-cfg.xml file. Use the administrative console to determine the port on which the internal transport is listening.
- Create an SSL key file for the Web server plug-in.

The content of this file depends on whom you want to allow to communicate directly with the application server over the port number specified for the HTTPS internal transport. It defines the HTTPS server security policy. The following procedure describes how to create an SSL key file with a restrictive security policy in which only the WebSphere plug-ins for the Web server are allowed to connect to the application server HTTPS internal transport:

- a. Create an SSL key file without the default signer certificates.
 - 1) Start IKeyMan.
 - 2) Create a new key database file. Click **Key Database File > New**. Then specify settings:
 - Key database type: **JKS**
 - File Name: **appServerKeys.jks**
 - Location: *your myKeys directory*, such as */install_root/myKeys*

Click **OK**.

- 3) Enter a password (twice for confirmation) and click **OK**.
- 4) Delete all of the signer certificates.
- 5) Click **Signer Certificates > Personal Certificates**.
- 6) Add a new self-signed certificate. Click **New Self-Signed** to add a self-signed certificate. Specify settings:
 - Key Label: **appServerTest**
 - Organization: **IBM**

Click **OK**.

- 7) Extract the certificate from this self-signed certificate so that you can import it into the plug-in SSL key file.
 - Click **Extract Certificate**. Specify settings:
 - Data Type: **Base64-encoded ASCII data**
 - Certificate file name: **appServer.arm**
 - Location: *path_to_myKeys_directory*

Click **OK**.

- 8) Import the plug-in certificate. Click **Personal Certificates > Signer Certificates > Add**. Specify settings:
 - Data Type: **Base64-encoded ASCII data**
 - Certificate file name: **appServer.arm**
 - Location: *path_to_myKeys_directory*

Click **OK**.

- 9) Enter **plug-in** for the label and click **OK**.
 - 10) Click **Key Database File > Exit**.
- b. Add the application server signer certificate to the plug-in SSL key file.
- 1) Start the key management utility.
 - 2) Click **Key Database File > Open**.
 - 3) Select the file */install_root/myKeys/pluginKeys.kdb*.
The *pluginKeys.kdb* file is the key database file, created using the z/OS System SSL *gskkyman* utility, that contains the public keys, private keys, trusted CAs, and certificates for the Web server plug-ins.
 - 4) Enter the associated password and click **OK**.
 - 5) Click **Personal Certificates > Signer Certificates**.
 - 6) Click **Add**. Then specify settings:
 - Data Type: **Base64-encoded ASCII data**
 - Certificate File Name: **appServer.arm**
 - Location: *path_to_myKeys_directory*
 - 7) Click **OK**.
 - 8) Click **Key Database File > Exit**.
- c. Reference the key file in the administrative console.
Reference the appropriate SSL key file in the default SSL settings configuration panel or in the HTTPS SSL settings configuration panel. Using the default SSL settings panel, you would:
- 1) Start the administrative console.
 - 2) Open the Security Center.
 - 3) Specify settings in the default SSL configuration:
 - Key File Name: *install_root/myKeys/appServer.jks*
 - Key File Password: enter your password
 - Key File Format: **JKS**
 - Trust File Name: (empty)
 - Trust File Password: (empty)
 - Client Authentication: **selected**
- d. Modify the Web server plug-in configuration file to indicate that you are using an HTTPS internal transport, and to add the keyring and stashfile properties to the definition of this internal transport.

Example: The Server Group definition for J2EE server BBS1122, with server instances BBS1122A and BBS1122B defined, looks like the following:

```
<ServerGroup Name="Default Host/Default Server">
  <Server CloneID="BBS1122.BBS1122A" Name="INSTANCE_A">
    <Transport Hostname="websphere1.ibm.com" Port="8083"
      Protocol="https"/>
      <Property name="keyring" value="C:\ssl\http_session
        \plug-inKeys.kdb">
      <Property name="stashfile" value="C:\ssl\
        http_session\plug-inpw.sth">
    </Transport>
  </Server>
  <Server CloneID="BBS1122.BBS1122B" Name="INSTANCE_B">
    <Transport Hostname="websphere1.ibm.com" Port="8084"
      Protocol="https"/>
    <Property name="keyring" value="C:\ssl
```

```

                                \http_session\plug-inKeys.kdb">
    <Property name="stashfile" value="C:\ssl
                                \http_session \plug-inpw.sth">
    </Transport>
  </Server>
</ServerGroup>

```

where:

- plug-inKeys.kdb is the key database file created using the z/OS System SSL gskkyman utility, that contains the public keys, private keys, trusted CAs, and certificates for the Web server plug-ins.file containing the keys for the plug-ins.
- plug-inpw.sth is the stash file in which the z/OS System SSL gskkyman utility stored the encrypted database password for these certificates.

See your Web server documentation for more information about these files.

5. Stop the application server and the Web server and start them again. The configuration is complete. In order to activate the configuration, stop and restart both the the application server and the Web server.

Related tasks

“Installing a distributed platform Web server plug-in” on page 16
Starting the key management utility (iKeyman)

Gskit install images files

The following table lists the directory location of the Global Security Kit (GSKit) installation image files for WebSphere plug-ins for Web servers that are running on a distributed platform. The appropriate file must be downloaded to the workstation on which the your Web server is running.

Operating system	GSKit Installation image file
Windows 2000 and Windows NT	<i>install_root</i> / Download Plugins/ Win32gsk5bas.exe_bin
AIX	<i>install_root</i> / DownloadPlugins/AIX/ gsk/gskkm.rte_bin
HP-UX	<i>install_root</i> / DownloadPlugins/HPUX/ gsk/gsk5bas.tar.Z_bin
Solaris Operating Environment	<i>install_root</i> / DownloadPlugins/Solaris/ gsk/gsk5bas.tar.Z_bin
Linux	<i>install_root</i> / DownloadPlugins/LINUX/ gsk/gsk5bas-5.0-4.79. i386.rpm_bin

Related tasks

“Installing a distributed platform Web server plug-in” on page 16

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:

- “Plug-ins: Resources for learning” on page 26
- “Plug-ins: Resources for learning” on page 26

Programming model and decisions

- Best Practice: WebSphere Plug-in Configuration Regeneration
- WebSphere Application Server V4.0 and V4.0.1 for z/OS and OS/390: Configuring Web Applications
- Best Practice: Configuring Web Applications on WebSphere Application Server for z/OS and OS/390

Programming instructions and examples

- IBM HTTP Server documentation
- WebSphere Application Server education
- Listing of all IBM WebSphere Application Server Redbooks
- Redbook on IBM WebSphere V4.0 Advanced Edition Scalability and Availability

Related tasks

Chapter 2, “Configuring Web server plug-ins,” on page 3

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.

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 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.

Related tasks

"Configuring virtual hosts" on page 31

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 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.

Related tasks

"Configuring virtual hosts" on page 31

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.

Related tasks

"Configuring virtual hosts" on page 31

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.

Related tasks

"Configuring virtual hosts" on page 31

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 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 stand-alone 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 and restart WebSphere Application Server.

Related concepts

"Virtual hosts" on page 29

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.

Related tasks

Exporting applications

Virtual host collection

Use this page to manage virtual hosts.

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

Related concepts

“Virtual hosts” on page 29

Related tasks

“Configuring virtual hosts” on page 31

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console filter settings

Use the Filter settings to specify how to filter entries shown in a collection view.

Administrative console preference settings

Use the Preference settings to specify how you would like information to be displayed on an administrative console page.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

Name

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

A virtual host configuration lets a single host machine resemble multiple host machines. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even when virtual hosts share the same 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*.

Related concepts

“Virtual hosts” on page 29

Related tasks

“Configuring virtual hosts” on page 31

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

“Host alias collection” on page 33

“MIME type collection” on page 34

Name:

Specifies a logical name 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 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**.

Related concepts

“Virtual hosts” on page 29

“How requests map to virtual host aliases” on page 30

Related tasks

“Configuring virtual hosts” on page 31

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console filter settings

Use the Filter settings to specify how to filter entries shown in a collection view.

Administrative console preference settings

Use the Preference settings to specify how you would like information to be displayed on an administrative console page.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

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, the host alias name is 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, 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***.

Related concepts

“How requests map to virtual host aliases” on page 30

Related tasks

“Configuring virtual hosts” on page 31

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

Host Name:

Specifies the IP address, 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 host alias name is myhost, the DNS name is myhost:8080, where 8080 is the port. A URL refers to this DNS as:
http://myhost:8080/servlet/snoop.

For existing instances, the default reflects the value specified at product setup. For new instances, the default can be * to allow any value or no specification.

Data type	String
Default	*

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, 9080 or a similar value.

Data type	Integer
Default	80

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 a virtual host. Virtual host MIME entries apply when you do not specify MIME entries at the Web module level.

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

Related concepts

“Virtual hosts” on page 29

Related tasks

“Configuring virtual hosts” on page 31

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console filter settings

Use the Filter settings to specify how to filter entries shown in a collection view.

Administrative console preference settings

Use the Preference settings to specify how you would like information to be displayed on an administrative console page.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

MIME Type:

Specifies a MIME type, which can be text, image, or application. An example value for MIME type is text/html.

Extensions:

Lists file extensions of files that map the MIME type. Example extensions for a text/html MIME type include .htm, .html, and .txt.

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**.

Related concepts

“Virtual hosts” on page 29

Related tasks

“Configuring virtual hosts” on page 31

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

MIME Type:

Specifies a MIME type, which can be text, image, or application.

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:

Lists file extensions of files that map the MIME type.

Example file extensions for a text/html MIME type include .htm, .html, and .txt. A default value appears 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*.

Related tasks

“Configuring variables”

IBM Toolbox for Java JDBC driver

Configure and use the jt400.jar file

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.

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. To have the configuration take effect, stop the server and then start the server again.

Related concepts

“Variables” on page 36

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 the variable and read the value in the **Description** field.

Related concepts

“Variables” on page 36

Related tasks

“Configuring variables” on page 36

IBM Toolbox for Java JDBC driver

Configure and use the jt400.jar file

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console scope settings

Use Scope settings to filter the contents of an administrative console collection table to a particular cell, node, or server, for example. Changing the value for Scope allows you to see other variables that apply to an object and might change the contents of the collection table.

Administrative console filter settings

Use the Filter settings to specify how to filter entries shown in a collection view.

Administrative console preference settings

Use the Preference settings to specify how you would like information to be displayed on an administrative console page.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

Name

Specifies a symbolic name that represents a file path, Web address, or other value.

Value

Specifies the value that the symbolic name represents, such as an absolute file path.

Scope

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

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 > Manage WebSphere Variables > WebSphere_variable_name**.

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

Name:

Specifies a symbolic name that represents a file path, Web address, or other value.

WebSphere Application Server substitutes the symbolic name wherever its value appears in the system.

For example, "CONFIG_ROOT" is the symbolic name representing the configuration directory path "C:\WebSphere\AppServer\Config" for the base WebSphere Application Server product on a Windows system.

Data type String

Value:

Specifies the value that the symbolic name represents, such as absolute file path.

For example, the value might be the absolute file path, "C:\WebSphere\AppServer\Config," in the base WebSphere Application Server product on a Windows system. The corresponding symbolic name might be "CONFIG_ROOT."

Data type String

Description:

Documents the purpose of a 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 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.

Related concepts

Class loaders

Class loaders are part of the Java virtual machine (JVM) code and are responsible for finding and loading class files. Class loaders affect the packaging of applications and the run-time behavior of packaged applications deployed on application servers.

Related tasks

“Managing shared libraries”

Related reference

“Shared library collection” on page 40

“Shared library settings” on page 41

Managing shared libraries

If your deployed applications use shared library files, set variables for the library files and associate the files with specific applications or with an Application Server, which associates the files with all applications on the server. Use the Shared Libraries page to define new shared library files to the system and remove them.

1. Identify library files and their classpaths.
 - a. Click **Environment > Shared Libraries** in the console navigation tree to access the Shared Libraries page.
 - b. 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**.
 - d. On the settings page for a shared library, specify the name, classpath, and any other variables for the library file that are needed.
 - e. Click **Apply**.

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

2. Associate shared library files with an application that must use one or more shared libraries.

Use this step to associate a file with an application or perform the next step to associate a file with an Application Server, which associates the file with every application on the server.

 - a. Click **Applications > Enterprise Applications** in the console navigation tree.
 - b. Click on the installed application that uses the shared libraries.
 - c. Click **Libraries** to access the Library Ref page.
 - d. Click **Add**.
 - e. On the settings page for a library reference, specify variables for the library reference as needed.
 - f. Click **Apply**.

Repeat this step until you define a library reference instance for each library file that your application requires.

3. Click **Servers > Application Servers > *server_name*** to associate a shared library with an Application Server for the run-time environment.

Use this step to associate a file with an Application Server, which associates the file with every application on the server, or perform the previous step to associate a file with an application.

- a. Set the application class-loader (sometimes referred to as classloader) policy and application class-loader mode on the settings page for an application server as described in Class loading.
- b. Click **Libraries** on the settings page for a class loader.
- c. Click **Add** from the Library Ref page.
- d. Specify variables for the library reference as needed on the settings page for a library reference and click **OK**.

Repeat this step until you define an Application Server for each library file that your application needs.

4. Remove a library file from the collection of shared library files by placing a checkmark beside the library you want removed on the Shared Libraries page and clicking **Delete**.

Related concepts

Class loaders

Class loaders are part of the Java virtual machine (JVM) code and are responsible for finding and loading class files. Class loaders affect the packaging of applications and the run-time behavior of packaged applications deployed on application servers.

“Shared library files” on page 38

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.

Related tasks

“Managing shared libraries” on page 39

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console scope settings

Use Scope settings to filter the contents of an administrative console collection table to a particular cell, node, or server, for example. Changing the value for Scope allows you to see other variables that apply to an object and might change the contents of the collection table.

Administrative console filter settings

Use the Filter settings to specify how to filter entries shown in a collection view.

Administrative console preference settings

Use the Preference settings to specify how you would like information to be displayed on an administrative console page.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

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*.

Related tasks

“Managing shared libraries” on page 39

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

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

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**.

Related tasks

“Managing shared libraries” on page 39

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console filter settings

Use the Filter settings to specify how to filter entries shown in a collection view.

Administrative console preference settings

Use the Preference settings to specify how you would like information to be displayed on an administrative console page.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

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.

Related tasks

“Managing shared libraries” on page 39

Class loading

Related reference

Administrative console buttons

The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console page features

This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

Library Name:

Specifies the name of the shared library to use for the library reference.

Data type

String

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:

- “Environment: Resources for learning” on page 42
- “Environment: Resources for learning” on page 42

Programming instructions and examples

- WebSphere Application Server education
- WebSphere Application Server V4.0 and V4.0.1 for zOS and OS/390: Configuring Web Applications
- Best Practice: Configuring Web Applications on WebSphere Application Server for z/OS and OS/390

Administration

- Listing of all IBM WebSphere Application Server Redbooks

Related tasks

Chapter 3, “Configuring the cell-wide environment,” on page 29

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks and service marks

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

- AIX
- CICS
- Cloudscape
- DB2
- DFSMS
- Everyplace
- iSeries
- IBM
- IMS
- Informix
- iSeries
- Language Environment
- MQSeries
- MVS
- OS/390
- RACF
- Redbooks
- RMF
- SecureWay
- SupportPac
- ViaVoice
- VisualAge
- VTAM
- WebSphere
- z/OS
- zSeries

The term CORBA used throughout this book refers to Common Object Request Broker Architecture standards promulgated by the Object Management Group, Inc.

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

The Duke logo is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.

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.